Cape Peninsula
University of Technology

DEVELOPMENT OF METHODS FOR MODELLING, PARAMETER AND STATE ESTIMATION FOR NONLINEAR PROCESSES

by

NTUTHUKO MARCUS DUBE

Thesis submitted in fulfilment of the requirements for the degree

Doctor of Technology: (Electronic Electrical and Computer Engineering)

in the Faculty of Engineering

at the Cape Peninsula University of Technology

Supervisor: Prof. Raynichka Tzoneva

Bellville
Submitted: October 2017

STATEMENT

I, Ntuthuko Marcus Dube, declare that the contents of this dissertation/thesis represent my own unaided work, and that the dissertation/thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

	11 October 2017
Signed	**Date**

ACKNOWLEDGEMENTS

I wish to thank:

- Prof. Raynichka Tzoneva (PhD). She has been my mentor for all the years that I have been involved in research spanning over a fifteen year period. Her passionate love of Control Systems and Control Engineering has been transferred to me and this thesis attests to that. She has inspired, motivated and above all, guided me as a teacher and a parent. I wish to thank her for her patience with me for all these years and every amount of time she has ever dedicated to me.
- Mr. Bruce Hendry of Chemical Engineering Department, Cape Peninsula University of Technology (Bellville). I would like to thank Mr. Hendry for all the time he afforded us and the knowledge passed onto us about the Ion Exchange process. Without this information and knowledge, this project would have been impossible. Thank you.
- Electrical Engineering Department of Cape Peninsula University of Technology (Bellville) for all the lifetime opportunities and personal growth the department has offered me in many years spent at the department.
- Ms Koketso P. Kujane, Mr Carl Kriger, Dr Sydney du Plessis, & Dr Senthil Krishnamurthy, thank you all, for your honest and full support you provided.
- Mr. Thabiso Hepworth Nhlapo, your giving heart, compassionate and charitable aid can never be equaled; and for being a benevolent philanthropist, a big ‘Thank You’ to you Mr Nhlapo. Only a handful of individuals are capable of the thoughtfulness and wisdom as yours.

The financial assistance of the National Research Foundation towards this research is acknowledged. Opinions expressed in this thesis and the conclusions arrived at, are those of the author, and are not necessarily to be attributed to the National Research Foundation.

DEDICATION

For my parents Luke Pika & Josephine Nokuthula (MaMhlongo) Dube,

(aunt) Hlengiwe Reddie Gwala Nee Dube, (sister) Thembalihle Marcia Dube,

(uncle) Mr. Sipho Mhlongo &

my family.

ABSTRACT

Industrial processes tend to have very complex mathematical models that in most instances result in very model specific optimal estimation and designs of control strategies. Such models have many composition components, energy compartments and energy inventories that result in many process variables that are intertwined and too complex to separate from one another. Most of the derived mathematical process models, based on the application of first principles, are nonlinear and incorporate unknown parameters and unmeasurable states. This fact results in difficulties in design and implementation of controllers for a majority of industrial processes. There is a need for the existing parameter and state estimation methods to be further developed and for new methods to be developed in order to simplify the process of parameters or states calculation and be applicable for real-time implementation of various controllers for nonlinear systems.

The thesis describes the research work done on developing new parameter and state estimation methods and algorithms for bilinear and nonlinear processes. Continuous countercurrent ion exchange (CCIX) process for desalination of water is considered as a case study of a process that can be modelled as a bilinear system with affine parameters or as purely nonlinear system. Many models of industrial processes can be presented in such a way. The ion exchange process model is developed based on the mass balance principle as a state space bilinear model according to the state and control variables.

The developed model is restructured according to its parameters in order to formulate two types of parameter estimation problem – with process models linear and nonlinear according to the parameters. The two models developed are a bilinear model with affine and a nonlinear according to the parameters model. Four different methods are proposed for the first case: gradient-based optimization method that uses the process output measurements, optimization gradient based method that uses the full state vector measurements, direct solution using the state vector measurements, and Lagrange’s optimization technique. Two methods are proposed for the second case: direct solution of the model equation using MATLAB software and Lagrange’s optimisation techniques.

The state estimation problem is formulated for the original model of the process and is solved by two proposed methods: for design of a bilinear observer based on the pole placement approach, and for design of two types of Kalman filters based on minimization of the least square of the filter error using general and direct formulations of the problem.

The developed methods are verified by software simulations in the environment of MATLAB/SIMULINK for various conditions of the process and problem formulations. Data from experiments with 6 stages ion exchange process are used to perform the parameter and state estimation algorithms.

The significance of this research is that it is based on industrial bilinear and nonlinear processes with the intention for further real-time implementation of the proposed methods. The new methods and algorithms provide:

- modern and optimal control techniques that are easily manipulated for different control synthesis,
- possibility of real-time implementation of control strategies for complex nonlinear processes,
- quick recalculation of specific variables using software based on "*newly*" available process data, and
- possibility to be used for various types of linear and nonlinear models.

There is always a cross field necessity for improvement of control techniques in process control, and optimization has taken the centre stage in allowing cross-field analysis and multidisciplinary collaboration in engineering. In this research work, the interrelation between chemical and control engineering processes is an integral part of the work covered in that control system design is applicable to almost all industrial processes.

Community relevance of this research is its applicability on most industrial processes since they are nonlinear in nature. The application of research in the continuous countercurrent ion exchange process could not have come at any better time than at the current experienced problem of water scarcity as a resource that is deepening almost every day in the country.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Acronym	Description
AFKF	adaptive fading Kalman filter
ALS	adjusted least squares
AMLE	approximate maximum likelihood estimation
AR	auto regression
ARMA	auto-regression moving average
ARMAX	auto regression moving average with eXogenic term
ARO	adaptive robust observer
CCIX	continuous countercurrent ion exchange
CPUT	Cape Peninsula University of Technology
CTT	classical test theory
DE	differential evolution
EM	estimation maximization
EnKF	ensemble Kalman filter
GA	genetic algorithm
ISOPE	Integrated System Optimization and Parameter Estimation
IRT	item response theory
IX	ion exchange
K	conductivity
LMI	linear matrix inequality
MIMO	multi-input multiple output
MAP	maximum a posteriori
ML	maximum likelihood
MLE	maximum likelihood estimator

MPC	model predictive control
MSE	mean squared error
NARMA	nonlinear auto-regression moving average
PDF	probability density function
pH	potential of Hydrogen
PCG	preconditioned conjugate gradients
PID	Proportional Integral Derivative
PSO	particle swarm optimization
RPE	recursive prediction error
SA	simulated annealing
SDRE	state-dependent Riccati equation
SISO	single-input single-output
SMC	sequential Monte-Carlo
STARMA	space-time autoregression moving average
SVD	singular value decomposition
TLS	total least squares
UCT	University of Cape Town
UKF	unscented Kalman filter
WLS	weighted least squares

NOMENCLATURE

Symbolic:

$\lambda(k)$	vector of Lagrange multipliers
α	iteration gradient step
α_p	gradient step used in procedure for determining optimal unknown parameters
α_H^{Na}	separation factor described by sodium content in solution
α_{Na}^{H}	separation factor described by hydrogen content in solution
$\alpha_{H,n}^{Na}$	separation factor in each stage (n) of column of the process
$\alpha_H^{Na}(k)$	vector of separation factor representing unknown parameters
$\alpha_H^{Na}(t)$	separation factor between exchanging cations in continuous time
σ	variance of the disturbance
φ	unknown constant of a linear model
ϕ	linear regressor
$\phi_i(t)$	model variables (states or parameters) in continuous time
ϕ_m	memory vector of the process with noise, containing measured values or values calculated from measurements
θ	vector of combined unknown states and parameters vector,
$\beta(\hat{p})$	Bayes risk, estimated error based on estimated parameters p
δ_{kj}	Kronecker delta function
ρ	deterministic independent variable of the random function $f(\gamma)$
γ	random variable
χ^2	Chi-squared
$\xi_i(k)$	first state estimate at iteration i
$\zeta(k)$	extended state vector

$\zeta(k_0)$	extended state vector initial sample value
ε	predetermined very small number for stopping the estimation procedure
ε_1	small number used to terminate the estimation procedure
ε_2	small number used to terminate the estimation procedure
Δ_λ	gradient step for the *Lagrange multipliers*
$\Delta_{\alpha_{H,n}^{Na}}$	gradient step for the *separation factor*, and.
Δ_p	small deviation of each component of the parameter vector p
Δp_i	small deviation of each component of the parameter vector in the i^{th} iteration
Δp_i^j	small deviation of each component of the parameter vector in the i^{th} iteration
Δt	sampling period to convert continuous time to discrete time

General:

$1-\alpha_{H,n}^{Na}$	separation factor presented as the difference between exchange at equilibrium and sodium content in the n stage
$1-x_H$	sodium ionic fraction in the liquid phase presented as the difference between exchange equilibrium and hydrogen content of the solution
$1-\underset{(liquid)}{x_H}$	sodium ionic fraction in the liquid phase presented as the difference between exchange equilibrium and hydrogen content of the solution
$1-x_{n,H}(t)$	sodium ionic fraction in the liquid phase presented as the difference between exchange equilibrium and hydrogen content of the solution for stage n
$1-x_{Na}$	hydrogen ionic fraction represented as the difference between exchange at equilibrium and sodium content of the solution

$1-y_H$	sodium ionic fraction represented as the difference between exchange at equilibrium and hydrogen content of the solution in resin
$1-y_{H_{(resin)}}$	sodium ionic fraction presented as the difference between exchange at equilibrium and hydrogen content of the solution in resin
$1-y_{Na}$	hydrogen ionic fraction in the resin phase presented as the difference between exchange equilibrium and sodium content of the solution
a_0	coefficient of a process model
a_i	coefficients of the output polynomial in an input-output model
a	ionic fraction of resin in a stage of the process column
a_n, $n=\overline{1,N}$	model coefficients of the n^{th} stage
$\overline{a}_n$	coefficients of a process model representing the linear relationship between resin and liquid
A^+	positive cation ions attached to anion resins
A	ion exchange state model coefficients model matrix
$A\in R^{n\times n}$	state matrix with corresponding dimensions of $n\times n$
A_{err}	error matrix determining closeness of fit to the real state
$\overline{A}(k)$	combined observer gain and system transition matrices
$A(k+1,k)$	transition matrix used to transform a given state at moment k to the next moment $k+1$
$A(k+1\vert k)$	state transition matrix at $k+1$ given current measurement k
b	intercept of the pseudo equilibrium curve for liquid and resin
b_i	coefficients of the input polynomial in a process model
b_n, $n=\overline{1,N}$	coefficients of a process model
$\overline{b}_n$	coefficients of a process model representing the linear relationship between resin and liquid in resin

$b(t)$	random sequence
B^{+}	positive cation ions from liquid that are exchanging with cations B^{+} in the resin in the cation column
B	control input matrix
$B \in R^{n \times m}$	control input matrix with $n \times m$ dimensions
B_1	ion exchange bilinear term coefficients matrix
$B_1 \in R^{n \times n}$	state transition matrix with influence from both the control inputs and states with $n \times n$ dimension
c_i	weighting coefficient of the linear combination of random and unknown values of noise
$C \in R^{l \times n}$	process observation matrix
$CaCl_2$	calcium chloride
$Ca(OH)_2$	calcium hydroxide
C_{H^+}	concentration of hydrogen (H) in liquid phase [meq/l],
Cl^{-}	chloride
C_n^*	total concentration in solution for all exchanging materials
C_{Na+}	concentration of sodium (Na) in liquid phase [meq/l],
$\operatorname{cov}\{x(k_0)\}$	covariance of the initial state x_0
$C(k) \in R^{l \times n}$	system output observation matrix with dimensions $l \times n$
$C(\hat{p})$	vector of state estimate values
$\overline{C}_{H+}$	concentration of hydrogen (H) in resin phase,
$\overline{C}_{Na+}$	concentration of sodium (Na) in resin phase,
d	resin/liquid fractional balance
d_k	time delay relative to the sampling period,
$\det_1(s)$, $\det_2(s)$, $\det_3(s)$	sub-determinants of $\det_{err}(s)$ used to simplify the error determinant solution
$\det_{obs}(t)$	observer determinant determined by the characteristic equation
$\det_{err}(s)$	determinant of the estimation error in the s-plane
$\det_{des}(s)$	determinant of the desired poles in the s-plane
D	state space output model matrix with model parameters

$e_{\lambda}(k)$	minimum error determined from the derivative of the Lagrange function with respect to Lagrange multipliers
$e_{\alpha_H^{Na}}(k)$	minimum error determined from the derivative of the Lagrange function with respect to the separation factor
$e_{\alpha_{n,i}}(k)$	minimum error of the derivative of the Lagrange function based on the analytical functions $f_{1,n}(k)$
$e_{\alpha_{(n+1),i}}(k)$	minimum error of the derivative of the Lagrange function based on the analytical functions $f_{1,(n+1)}(k)$
$\tilde{e}$	error of an estimate
e	error difference between actual and estimated values
e_i	additive error difference at specific moments of i
$e(i \vert k)$	error difference of the i^{th} sample between observed state and the estimated state on the k^{th} moment
$e_0(k)$	initial error condition at moment $k = 0$
$e(k)$	error difference in discrete models
$\lVert e(k)^2 \rVert$	least square error used as a criterion for successful estimation
$e_{Na}(k)$	error difference between measured and estimated states data of model based on the sodium concentration of the liquid
$e(k \vert k)$	*a posteriori* estimate error at moment k^{th} based on measurement at the k^{th} moment
$e(k \vert k-1)$	*a priori* estimate error determined at the moment k
$e(k+1 \vert k)$	*a priori* estimate error based on a priori moment $(k+1 \vert k)$
$e(k+1 \vert k+1)$	a posteriori estimate error at $(k+1)$ moment
$e(k-i)$	delayed unknown values of noise
$e\{\gamma\}$	linear function error model
$e\{C(\hat{p})\}$	error risk
$e[Cp, \hat{p}(z)]$	estimation error

$e_i(t)$	error difference at specific moments of i for a continuous time signal
$e_x(k)$	error difference measured state values and calculated state values
$E\{\ \}$	covariance index, the expectation operator
$E\{\gamma\}$	mean of the random variable γ
$E\{e_x(t)^T e_x(t)\}$	covariance index of the error difference between estimated states and measured states
$E\{e(k)e(k)^T\}$	covariance of the prediction error
$E\{e(k\|k-1)e(k\|k-1)^T\}$	covariance of the a *priori* estimate error
$E\{e(k+1\|k)e(k+1\|k)^T\}$	covariance of the a *priori* estimate error
$E\{e(k+1\|k+1)e(k+1\|k+1)^T\}$	covariance of the a priori estimate error at the moment $(k+1)$
$E\{e(k+1\|k)v(k+1)^T\}$	covariance relating the a priori estimate error to measurement noise
$E\{v(k)\}$	mean value of the measurement noise
$E\{v(k)v(k)^T\}$	covariance of the measurement noise vector
$E\{v(k)w(k)^T\}$	covariance of the measurement and system noises
$E\{v(k+1)v(k+1)^T\}$	covariance describing the behaviour of the noise measurement at moment $(k+1)$
$E\{v(k+1)e(k+1\|k)^T\}$	covariance relating the a priori estimate error to measurement noise
$E\{w(k)\}$	mean value of the system noise
$E\{w(k)v(k)^T\}$	covariance of the measurement and system noises
$E\{w(k)w(k)^T\}$	covariance of the system noise vectors
$f(x(k),k)$	function of nonlinear deterministic discrete time system
$f(x(k),u(k),k)$	discrete time nonlinear function model
$f(\hat{x}(t),u(t),t)$	function for observer model equation based on estimated states $x(t)$, system input $u(t)$,and time t

$f(x(t),u(t),p(t),t)$	function used to define nonlinear model with the process state, $x(t)$, the input, $u(t)$ model parameters $p(t)$ in time t
$f(x(t),u(t),w(t),p,t)$	function defining a system based on the system state, $x(t)$, system input, $u(t)$, system disturbance, $w(t)$, unknown parameters, p and time t
$fsolve()$	MATLAB function used to solve unknown parameters in a nonlinear model
FC_n	fractional change of ions in the solution of exchanging ions
$F(k)$	matrix representing the function defining the relationship between the separation factor and the model states
$F(k,\alpha_H^{Na})$	matrix representing the function defining the relationship between the separation factor and the model states
F_L	molar flow rate of the liquid entering the first stage of the load column as a constant variable [min]
$F_L(t)$	molar flow rate of the liquid in time [min]
$F_{L,n}(t)$	molar flow rate of the liquid out of stage n in resin-liquid solution in time [min]
F_R	molar flow rate of resin (solid) in solution entering the top stage of the load column (used as the control input) [min]
$F_R(t)$	molar flow rate of resin (solid) in time [min]
$F_R(k) \in R^l$	ion exchange process control input in time [min]
$g(x,t)$	continuous time nonlinear output function as described by states and time
$g(x(t),\theta(t),t)$	combined state and parameter estimation output continuous time model
$g(x(t),u(t),v(t),p,t))$	function defining an output of a system based on the system state $x(t)$, system input $u(t)$, measurement disturbance $v(t)$, unknown parameters p in time t
G	weighting matrix

$G(k)$	matrix representing the function defining the relationship between the separation factor to the control input and the state in the model
$G(k,\alpha_H^{Na})$	matrix representing the function defining the relationship between the separation factor to the control input and the state in the model
h	resin holdups as a constant variable
$h_n(t)$	resin holdups rate in stage n
H	liquid holdups as a constant variable
$H_n(t)$	resin holdups rate in stage n
HCl	hydrochloric acid
H_2O	chemical formula for water
H_2SO_4	sulphuric acid resulting from magnesium sulphate ions exchanging with resin in strong hydrogen form
i	index of iteration in order of 1
I	identity matrix of $n \times n$ dimensions
IFH_n^+	ionic fraction of the hydrogen ions (H^+) in the solution of the ion exchange process
$IFH_{initial}^+$	initial ionic fraction of the hydrogen ions (H^+) in the solution of the ion exchange process
IFH_{final}^+	final ionic fraction of the hydrogen ions (H^+) in the solution of the ion exchange process
j	iteration index used in the optimization procedure
J	cost function to be minimized or maximized
$J()$	cost function to be minimized or maximized
$J(\zeta)$	discrete cost function for state estimation using
$J(k)$	optimization criterion for minimizing the error between the estimated and real states
$J(p)$	criterion for estimation based on the p parameters
$J(p_1, p_2, ..., \Delta p_i^j, p_{i+1}, ..., p_q)$	criterion with the deviated value of the i^{th} component of the unknown parameter in the j^{th} iteration

$J_{\hat{x}}(t)$	objective function defining the difference between estimated state and the real state values
$J(k+1)$	criterion of the prediction error
$J_K(k+1)$	the last term in the optimization criterion
$J(\tilde{x}(t))$	objective function for estimating the system state
$J(x)$	performance index determining closeness of estimated state values from the actual values
$(k\|k)$	*a posteriori* moment defined at the k^{th} moment
$(k\|k-1)$	*a priori* moment defined at the k^{th} moment
$(k+1\|k)$	*a priori* moment defined at the $(k+1)^{th}$ moment
$(k+1\|k+1)$	*a posteriori* moment defined at the $(k+1)^{th}$ moment
$K^*(k)$	Kalman filter gains at the moment k
$K^*(k+1)$	Kalman filter gains at the moment $k+1$
$l_1(t),\ l_2(t),\ l_3(t),...,l_N(t)$	elements of the observer gain matrix
L	Luenberger gain matrix for state observer
$L(t)$	observer gain matrix in continuous time
L_m	likelihood function
L_a	Lagrange function used to estimate unknown parameters in the nonlinear model
$L \in R^{l \times n}$	observer matrix
m	maximum number of observations
M	large number for stopping iteration procedure
M^-	negative anion ions in resin
$MgSO_4$	magnesium sulphate
N^-	negative anion ions in liquid solution that must exchange to resins in the anion column
Na^+	sodium ions
$NaCl$	sodium chloride
N_i	total number of measurement points
$N(0,Q(k))$	covariance of the system noise
$N(0,R(k))$	covariance of the measurement noise

$N(0,V^*)$	probability distribution of measurement noise
$N(0,W^*)$	probability distribution of system noise
O	linear observability matrix determined using the state and output matrices pair (C,A)
o	rank value of the observability matrix rank, $rank(O)$
$p_1,\ p_2,...,p_N$	desired poles used in the pole placement method for estimating the unknown states
p	vector of unknown parameters (to be estimated)
p_i	system parameter values at each measurement moment
p_0	initial parameter values to be estimated
$p \in R^q$	vector of the system parameters with $q \times 1$ dimensions
p_n	parameter vector of the process model with noise
$\hat{p}$	estimated value of the unknown parameter p
$p(k)$	unknown parameter of p discrete vector
P_0	initial error covariance matrix
$P(0)$	initial error covariance matrix
P_{k_0}	covariance matrix of the initial state x_0 in discrete time
$P(k)$	covariance matrix of the prediction error
$P(k+1)$	probability of the estimation error, known as the covariance matrix of the prediction error
$P(k\|k)$	error covariance matrix that represents the covariance of the error difference between the true state and its estimate,
$P(k+1\|k)$	error covariance matrix at the next moment given the current error difference,
$P(k\|k-1)$	*a priori* covariance matrix represented at moment k with measurements coming from the moment measurement from the previous discrete moment $k-1$
$P(k+1\|k)$	probability matrix known as *a priori* covariance matrix
$P(k+1\|k+1)$	probability matrix known as *a posteriori* covariance matrix of the estimate error

$P(v(k))$	probability distribution of system noise
$P(w(k))$	probability distribution of measurement noise
q	total number of parameters in the vector of unknown parameter
Q_i	symmetrical, positive (non-negative), definite weighting matrix
$Q^*{}_i$	symmetrical, non-negative, definite weighting matrix
$Q(k,\alpha_H^{Na})$	matrix representing the function that defines the relationship between the separation factor and the disturbance input of the model
$Q(k)$	covariance matrix relating observation matrix and error estimate covariance matrix
$Q(k\|k)$	a priori covariance matrix relating observation matrix and error estimate covariance matrix
$Q(k\|k)^{-1}$	inverse matrix of the a priori covariance matrix relating observation matrix and error estimate covariance matrix
$Q(k+1)$	system noise covariance matrix at the next moment $k+1$
R^+	resins in positively charged form representing the solid phase in the exchange,
R^-	resins in negatively charged form representing the solid phase in the exchange,
R	weighting matrix
R_2-Ca	calcium ions attached to the cation resin
$R-H$	strong cation resin in hydrogen H form
$R(k)$	measurement noise covariance matrix
R/L	resin to liquid ratio in the solution
R_2-Mg	magnesium ions attached to the cation resin
$R-Na$	weak cation resin in sodium Na form after the exchange
$R \equiv NHCl$	hydrochloric acid protonation in resin
S	weighted sum of the least squares minimization error
$S(k)$	cost function for state estimation of a discrete time model
$S(k+1)$	innovation covariance used to calculate filter gain

T	process upflow period,
$u(t)$	system's control input in continuous time
$u(k) \in R^l$	control input vector
$u(k-d)$	discrete model input delayed by d
$u(k-d-nu)$	discrete model input delayed by d and n
$u(k-i)$	past model inputs
$u(k-n)$	past model inputs
$u(t)$	process input variable
$v(k)$	measurement noise vector in discrete time
$v(k)$	discrete time noise vector from measurement device
$\{v(k)\}$	measurement noise sequence in discrete time
$v(k\|k)$	a posteriori measurement noise determined at k^{th} moment
$v(t)$	measurement noise in continuous time
$V^*(k)$	measurement noise covariance matrix
$w(t)$	system disturbance variable (input noise)
$w(t) \in R^l$	disturbance to the system with l dimensions
$w(k)$	discrete time noise vector representing process (disturbance input) noise
$w(k\|k)$	a posteriori system noise determined at k^{th} moment
$w(k\|k)^T$	transpose of the a posteriori system noise determined at k^{th} moment
$\{w(k)\}$	system noise sequence in discrete time
W	system disturbance matrix
$W(k)$	system disturbance coefficients matrix in discrete time
$W(k) \in R^{n\times l}$	input disturbance coefficients matrix with dimensions $n \times l$
$W^*(k)$	covariance matrix of the system noise
$\tilde{x}(t)$	error difference between model calculated and measured state values
x	state of the process model
x_0	initial state of the system at time $t=0$ or $k=0$
$x(0)$	initial state of the system at time $t=0$ or $k=0$

x_{k_0}	initial state value at discrete moment $k = 0$
x_{H^+}	equivalent ionic fraction of hydrogen ions (H^+) in the liquid phase
x_H	hydrogen ionic fraction in liquid phase
$\underset{(liquid)}{x_H}$	amount of hydrogen content in liquid phase of the solution
$x_H \overset{1}{\underset{0}{\uparrow}}$	hydrogen ionic fraction through the column from the most bottom stage (first stage) to the last stage at the top of the column
$x_{in}(t)$	input concentration coming in to the first stage of the process
$x(k)$	system state in discrete time
x_n	state variable representing liquid concentration at stage n
$\bar{x}_N(k)$	measured state variable at the output of the process (liquid concentration coming out at the top stage of the column)
x_{Na^+}	equivalent ionic fraction of sodium ions (Na^+) in the liquid phase
x_{Na}	sodium ionic fraction in liquid
$\underset{(liquid)}{x_{Na}}$	amount of sodium content in liquid phase of the solution
$x_{Na} \overset{0}{\underset{1}{\uparrow}}$	sodium ionic fraction as decreasing throughout the column from the first column (most bottom) stage to the last (the top) stage
$x(t)$	state of the process in time domain
$x_n(t)$	mole fraction of the liquid phase at stage n
$x_N(t)$	mole fraction of the liquid phase at the last column stage, N
$x_f(t)$	input feed concentration at the first stage (bottom stage) of the column which changes according to the effluent's content; it is considered a disturbance to the process
$x_f(k) \in R^l$	input disturbance vector of the process

$x(k) \in R^{n \times l}$	system state vector in discrete time with $n \times l$ dimensions
$\{x(k)\}$	state sequence in discrete time
$\hat{x}(i\|k)$	estimated state of the system for every i^{th} sample in discrete time
$\hat{x}(k\|k)$	a posteriori estimate at the moment k
$\hat{x}(k_0\|k_0)$	initial conditions of the a posteriori estimate at the moment k
$\hat{x}(0\|0)$	initial state estimate at the beginning of the estimation procedure
$\hat{x}(k\|k)$	unbiased real estimate of $x(k)$ given the measurements of $z(k)$
$\hat{x}(k\|k-1)$	estimated state at discrete time k given the previous measurement $z(k)$ at moment $k-1$
$\hat{x}(k+1\|k)$	estimated state at moment $k+1$ given the measurement $z(k)$,
$\hat{x}(t)$	estimates of the state in continuous time
$\hat{x}(t\|t)$	estimate of the state at time t given the output measurement of time t
$\hat{x}(k\|k-1)$	a priori estimated state at the k^{th} moment
$\hat{x}(k+1\|k)$	a priori estimated state at the $(k+1)$ moment
$\hat{x}(k+1\|k) \in R^n$	a priori state estimate vector at the $(k+1)$ moment
$\hat{x}(k+1\|k+1)$	a posteriori estimate at the moment at the $(k+1)$ moment
y_n	state variable representing resin concentration at stage n
y_{H^+}	equivalent ionic fraction of hydrogen ions (H^+) in the resin phase
y_H	hydrogen fractional content in resin phase
$y_H \overset{1}{\underset{0}{\downarrow}}$	hydrogen ionic fraction in resin, decreasing from the top (last) stage to the first (most bottom) stage of the column, arrow indicates resin directional flow

$\underset{(resin)}{y_H}$	amount of hydrogen content in resin phase of the solution
y_{Na^+}	equivalent ionic fraction of sodium ions (Na^+) in the resin phase
y_{Na}	hydrogen content in resin
$y_{Na} \overset{0}{\underset{1}{\downarrow}}$	sodium ionic fraction in resin increasing from the first stage (most bottom stage) to the last stage at the top of the column
$\underset{(resin)}{y_{Na}}$	amount of sodium content in resin phase of the solution
Y	exchanging compound in liquid
z	observed system output
z_i	system output observed at i^{th} specific moments
$z(k)$	system output in discrete time
$z(k) \in R^l$	output measurement vector
$\{z(k)\}$	output sequence in discrete time
$z(k-i)$	past model outputs
$z(t)$	model output in continuous time
$z(t_i)$	model output in continuous time
$z(t_i, p)$	model output determined using unknown parameters
$\bar{z}(t)$	measured output
$\hat{z}(t)$	estimated model output
$\hat{z}(\hat{p}) \in R^l$	estimated output vector based on estimated parameters p.

CHAPTER ONE

INTRODUCTION

CHAPTER ONE

1. INTRODUCTION

1.1. Introduction

Most if not all physical processes are dynamic in nature due to energy transfer within the constituents of these systems. When developing a model for a dynamic process, the resulting model has nonlinear relationship among its variables resulting in a process model called the *nonlinear dynamic model* (Chitralekhaa, et al., 2010; Safdarnejad, Gallacher & Hedengren, 2016). Nonlinear systems are usually characterised by complex structure, non-stationary behaviour and time delays and it is these characteristics that create a challenge for engineers when designing and or implementing real-time control to such system (Ding, et al., 2017). Kravaris, Hahn and Chu (2013) present the latest advances and developments in parameter and state estimation for nonlinear processes.

The research work analyses the available nonlinear methods of modelling and parameter and state estimation and suggests improved methods or develops new methods with the intention of applying such methods in industrial applications.

This chapter introduces the overall research approach followed in developing new methods for modelling, parameter and state estimation of nonlinear processes as an objective of this research. The chapter considers the following subtopics: 1) Awareness of the problem, 2) problem statement, 3) project aims and objectives, 4) hypothesis, 5) scope of the project, 6) delimitations, 7) research assumptions, 8) research methods, 9) chapters breakdown, 10) motivation and significance of research. The chapter is then ended with conclusion remarks.

1.2. Awareness of the problem

Bilinear process models are considered a subsection on nonlinear models. The bilinear models have found many applications in various physical systems including a wide variety of fields such as engineering, economics, biological, including biomedical, neuroscientific systems, etc.

Studies that were identified in bilinear systems are focused on either the control strategy of bilinear processes, including stability, state feedback and disturbance feed forward analysis or state estimation using different observer formulations etc. (Bruni, DiPillo & Koch, 1974; Derese & Noldus, 1980; Derese & Noldus, 1981; Derese, Stevens, & Noldus, 1979; Dreano, Fantini & Laurent, 1997; Hammami & Jerbi, 2001; Hou & Pugh, 1997; Ying, Rao & Sun, 1990).

Most of the methods developed for parameter and state estimation in bilinear systems tend to be very complex, in that these methods are integrated techniques involving parameters that are estimated using data obtained from state estimation by filtering methods or observers (Dai, Sinha & Puthenpura, 1989; Frick & Valavi, 1978; Liu, Xu & Wang, 2012; Ober, Lin & Zou, 2005; Frick & Valavi, 1978). The research work in the thesis aims to overcome this problem by extending the existing and developing new methods on the basis of a proposed model with reduced complexity.

The models of the industrial processes tend to be very complex – nonlinear, with nonlinear parameters and variables and described by large set of differential equations. The reduced complexity in process models results in relaxed calculations requirements (Lalonde, Hartley & De Abreu-Garcia, 1992; Peng, et al., 2001; Safdarnejad, et al., 2016; Schei & Singstad, 1998; Weiss & Preisig, 1998). These models can be obtained by capturing some realistic aspects of a nonlinear model and relaxing some others, following the assumptions that must always be included in developing a model (Bai-Lan & Xiao-Lin, 1988; Ikonen & Najim, 2002:3-12; Fliess & Normand-Cyrot, 1910). Such possibility will be considered in the thesis.

The research work considers the parameter and state estimation for bilinear affine and nonlinear in parameters process models. The interest is to produce simplified and non-integrated parameter and state estimation methods for bilinear affine or nonlinear in parameters systems. The project considers the possibility of applying the newly developed methods in real-time control strategy. The main aim is to reduce complexity in estimation and simulation of the developed control strategy for the class of nonlinear systems.

1.3. Statement of the problem

The problem has been identified as non-availability of simplified and non-integrated methods for parameter and state estimation for a bilinear process models affine or purely nonlinear in parameters. In addition application of state and parameter methods for the case of complex industrial processes is done using some simplified models which do not present the true dynamics and behavior of these processes. This puts forward the argument which is the basis for this thesis, the development of new methods for modelling and parameter and state estimation for nonlinear systems with primary focus on bilinear systems. Two main thesis development sub-problems considered are:

1) The *sub-problem for developing of new simplified models of complex processes* (*that preserve the dynamics*) *and methods* for parameters and state estimation of bilinear processes, and
2) The *sub-problem of implementation* of the developed methods for the case of a complex industrial process.

1.3.1. The development sub-problem

The development sub-problem includes the following considerations:

- Analysis of the available published models of nonlinear processes, and parameter and state estimation methods,
- Development of a simplified bilinear model that fully represents the dynamics of the process,
- Development of methods for parameter estimation of bilinear process models with affine parameters,
- Development of methods for parameter estimation of bilinear processes models with nonlinear parameters,
- Development of methods for state estimation for bilinear process models with affine parameters.

1.3.2. The implementation sub-problem

The implementation sub-problem is concerned with evaluation and validation of the developed methods using an ion exchange (IX) process as a case study. The following considerations are included:

- Develop software programs using available process data to evaluate and validate parameter estimation methods, and
- Develop software programs using available process data to evaluate and validate the developed state estimation methods.

1.4. Research aim and objectives

The research work considers the parameter and state estimation for bilinear affine and nonlinear in parameters process models, considering the possibilities of applying the new methods in real-time control strategy. The developments are directed in two fields: 1) development of compact simplified models of complex dynamic processes for the purpose of simplification of the parameter and state estimation procedures – according to the requirements of the control problem and, 2) development of new methods for parameter and state estimation of the developed models. At the end of the study the investigator hopes to reduce the time needed for algorithm calculations since these calculations can be time consuming.

The main aim of the project is to develop simplified bilinear models of the complex industrial Ion exchange counter current process for desalination of water, and new methods for parameter and state estimation for the developed models, considering affine or nonlinear parameters.

The research objectives are as follows:

1) To conduct literature review on modelling of nonlinear processes in general, with the primary focus on bilinear models,
2) To derive a state space bilinear model of the ion exchange process fully representing the structure and dynamics of the process,
3) To develop methods for parameter estimation considering the bilinear affine in parameters model of the ion exchange process,
4) To develop methods for parameter estimation considering the bilinear nonlinear in parameters model of the ion exchange process,
5) To develop state observer design for the bilinear with affine parameters model of the ion exchange process,
6) To develop a Kalman filter for state estimation of the bilinear with affine parameters model of the ion exchange process,
7) To develop algorithms and software for implementation of the developed method for parameter estimation of the bilinear affine in parameters model,
8) To develop algorithms and software for implementation of the developed method for parameter estimation of the bilinear nonlinear in parameters model,
9) To develop algorithms and software for implementing the state observer for the bilinear affine in parameters model,
10) To develop algorithms and software for implementing the developed Kalman filter for state estimation of the bilinear affine in parameters model,
11) To develop software for simulation, comparison, and evaluation of results.

All developments are described in the following thesis chapters.

1.5. Hypothesis

The research work in the thesis considers the following hypotheses:

1) Nonlinear processes tend to have very complex models and the models complexity can be reduced without losing the important aspect of the process dynamics. The ion exchange multi-stage counter current process model can be derived to be in a state space domain, with a bilinear behavior, and to have both linear and nonlinear parameters on the basis of application of the first mass balance principle and mathematical transformations.
2) Possibility to express the model parameters as affine ones allows reduction of the complexity of the methods for state and parameter estimation and reduces the time for calculation of the estimated states and parameters.
3) State space presentation of the bilinear model allows the software for implementation of the developed methods for parameter and state estimation to be more compact and simpler as the calculations are done for the whole state vector at once.

1.6. Scope of the project

The thesis scope of the research work consists in development of a state space bilinear model of the counter current ion exchange process, acquiring data from experiments with the pilot plant built at the University of Cape Town (Hendry, 1982a; Hendry, 1982b) to be used for state and parameter estimation, development of methods for state and parameter estimation, software development and investigation of the performance of the developed methods through MATLAB/SIMULINK case studies. More data to determine deeper the thesis scope are given below:

Model development: Two models of the counter current multi-stage ion exchange process, a bilinear affine in parameters and a bilinear nonlinear in parameter are developed using assumptions and the first mass balance principle. The parameters and the states of the models are not known.

Acquiring data: Data for the input, output and input disturbance from experiments with the pilot plant are available. The data are normalized and verified before to be used.

Development of methods for parameter and state estimation: The project considers current methods used in parameter and state estimation and then develops and proposes new methods and algorithms to be used in parameter and state estimation. The state estimation problem is considered only for the bilinear affine in parameters model. The parameter estimation problem is considered for both bilinear models – with affine and with nonlinear parameters. Deterministic problems are considered for parameter estimations and problems with white model and measurement noises are considered for the state estimation problem.

Generally, when designing a control strategy for physical processes, before the full strategy can be developed, it is naturally inherent that unknown parameters must be identified. The problem of parameter estimation is important to determine the unknown or the partially known process parameters. Once the unknown or partially known parameters have been identified, the state estimation is the next step that allows identification of the unknown states of a process in case the process' states are not measureable, Figure 1.1.

The state estimation problem is used for developing a strategy for control and monitoring of a plant more especially in optimal control. Parameter estimation can be developed around two strategies: 1) the offline parameter estimation or 2) the on-line parameter estimation. In offline parameter estimation, the problem of determining unknown or partially known parameters is achieved using data that is pre-collected and the full set of the data is available to use. In on-line parameter estimation, data is collected during the process of determining the unknown parameters. In this research project all the data was available beforehand.

The estimation method followed is also dependent of the model structure. The commonly used model structures are:

1) Linear model with linear parameters,
2) Nonlinear model; firstly with parameters entering the model equation in a linear way and secondly with parameters entering the model equation in a nonlinear manner.

Two models will be considered for the parameter estimation problems. The first model is the bilinear affine in parameters model and the second one, the nonlinear model.

For the state estimation problem, the following are considered:

1) Suppose the parameters are known to solve the state estimation problem.
2) State estimation problem for the nonlinear model that considers the bilinear affine model is considered.
3) This problem is solved with the bilinear observer design and the Kalman filter design.

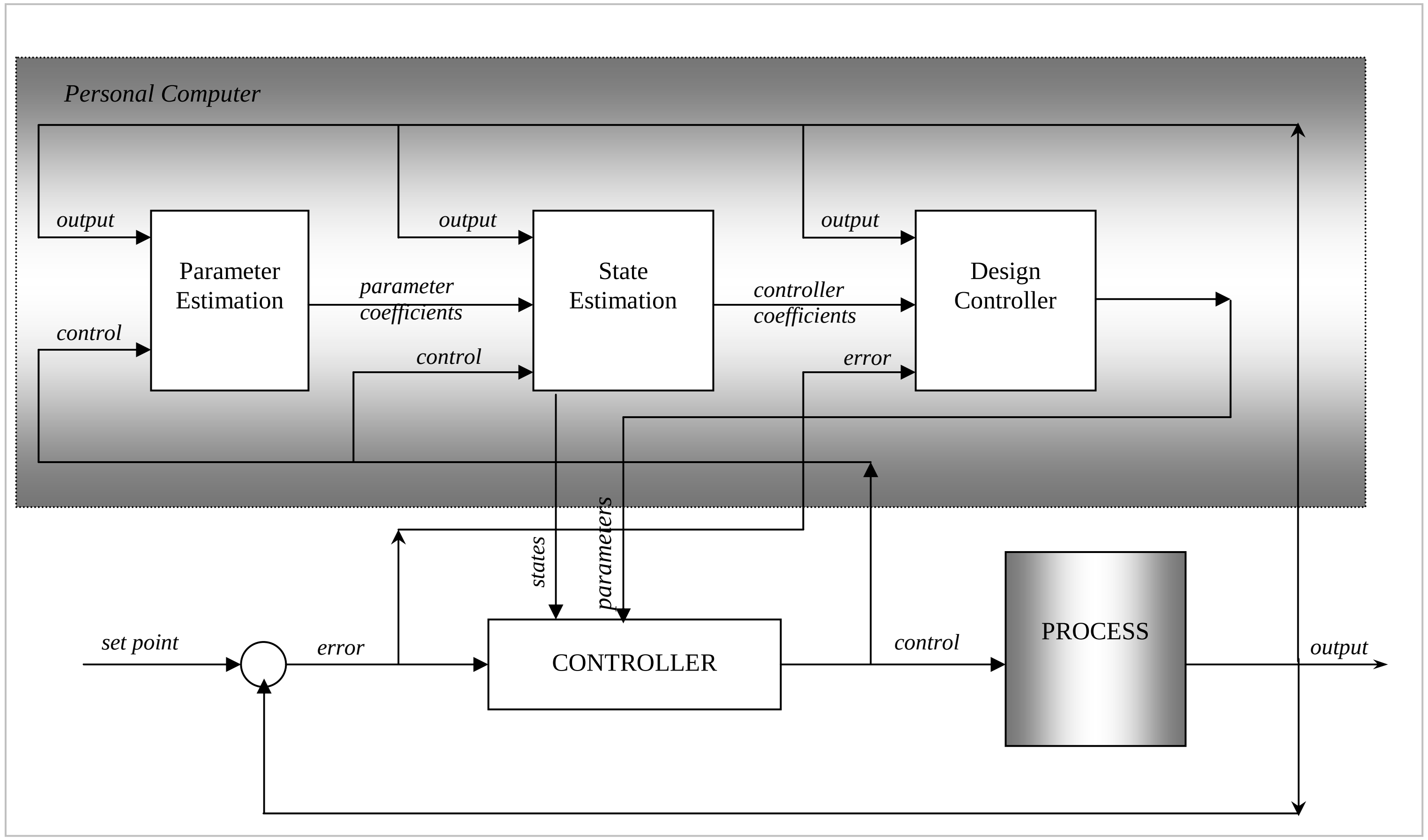


Figure 1.1. Overall proposed estimation structure

(Adapted from Ray & De, 1997; Eykhoff, 1968; Roberts & Williams, 1981)

The thesis considers optimization methods to formulate and solve the state and parameter estimation problems in a way that a cost function must be minimized. The cost function consists of a sum of the squared prediction errors; the sum of errors must be minimized. The prediction errors are the differences between the predicted values of the output of the process to that of the measured one. The estimation techniques use measured data to estimate the parameter values of the model. Similar problems are proposed in (Ikonen & Najim, 2002; Corlis & Luus, 1969; Pierre, 1969; Nicholson, Lopez-Negrete & Biegler, 2014; Olsson & Newell, 1999; Peng, et al., 2001; Schei, 2008; Schei & Singstad, 1998).

State estimation is the process of determining the state from these output measurements given the dynamic model of the system (Ching, et al., 2006; Muske & Edgar, 1997; Yang, et al., 2016).

The developed methods are innovative in that, currently from the literature, no methods were identified that propose state and parameter estimation methods for the bilinear affine in parameters model. The developed methods differ from most existing methods since they use real measured data as opposed to using an observer or filter generated states data to estimate parameters.

The developed parameter estimation methods for bilinear affine model are as follows:

1) a gradient method based on output measurements,
2) a gradient method that considers the full-order state observation,
3) a direct estimation method using state vector measurements, and
4) Lagrange’s optimization procedure based on state vector measurements.

Algorithms and software programs are developed on the MATLAB platform using the ion exchange case study.

The developed parameter estimation methods for the bilinear nonlinear in parameters model are:

1) a solution based on a function of MATLAB software, and
2) Lagrange’s optimization method using state vector measurement.

Methods developed for state estimation problem solution are:

1) a Luenberger based bilinear observer, designed using pole-placement which is a novel method in that it is being introduced for the first time in bilinear affine in parameter models, and
2) a Kalman filter – based estimation problem designed by two methods: the general formulation and the direct optimization technique. Algorithms for

the two methods are developed and only one software program is developed – for the second method.

1.6.1. Process considered as a case study

An ion exchange process is a chemical process that involves replacement of ions of the same charge from liquid to solid and vice versa, i.e., positive ions from liquid will exchange with positive ions from the solid material. The ion exchange process has a vast number of applications in fields as far apart as; mining, medical, water treatment, etc. In water desalination application, the process removes salts from water to purify it.

The basic countercurrent ion exchange process consists of four columns in pairs, Figure 1.2. The first two are cation load and cation regeneration. In the cation load column, water being purified is passed through the column and the resin is fluidized in the process, and salt ions from water are exchanged with resin ions. During the process of exchanging ions in the cation load, salts in water are extracted and the resulting solution is a weak acidic stream (Dodds, et al. 1973; Fronza, et al., 1969; Horn, 1967).

The second pair is anion load and anion regeneration columns, where the weak acid from the cation load is passed through weak base resin which has high affinity for acid. This results in the exchange of ions from the weak acid stream and strong base resin producing purified water as the final product. Regeneration columns in both cases are used for replenishing partially exhausted resins. As resin beds constantly exchange ions, after a while they get exhausted, fortunately resins allow the reversing of their exchange properties. Resins can always be renewed through regeneration by passing them through acid for the strong acid resins and through base for the weak base resin, Figure 1.2. pH and conductivity measurements are taken at the input and output of each phase to determine the amount of salt concentration in each stream (Dube, 2002; Dube & Tzoneva, 2001; Dube & Tzoneva, 2002; Dube & Tzoneva, 2003a; Dube & Tzoneva, 2003b; Dube & Tzoneva, 2005; Dube & Tzoneva, 2006; Dube & Tzoneva, 2006b).

It is of interest to note that for the pilot plant to be used in the study the liquid concentration measurements are not possible at each state, this demands the need for state estimation. The application of the state estimation solution will be to observe the dynamic behaviour of the sodium concentration in every stage of the process and for implementation of the state space controller if required.

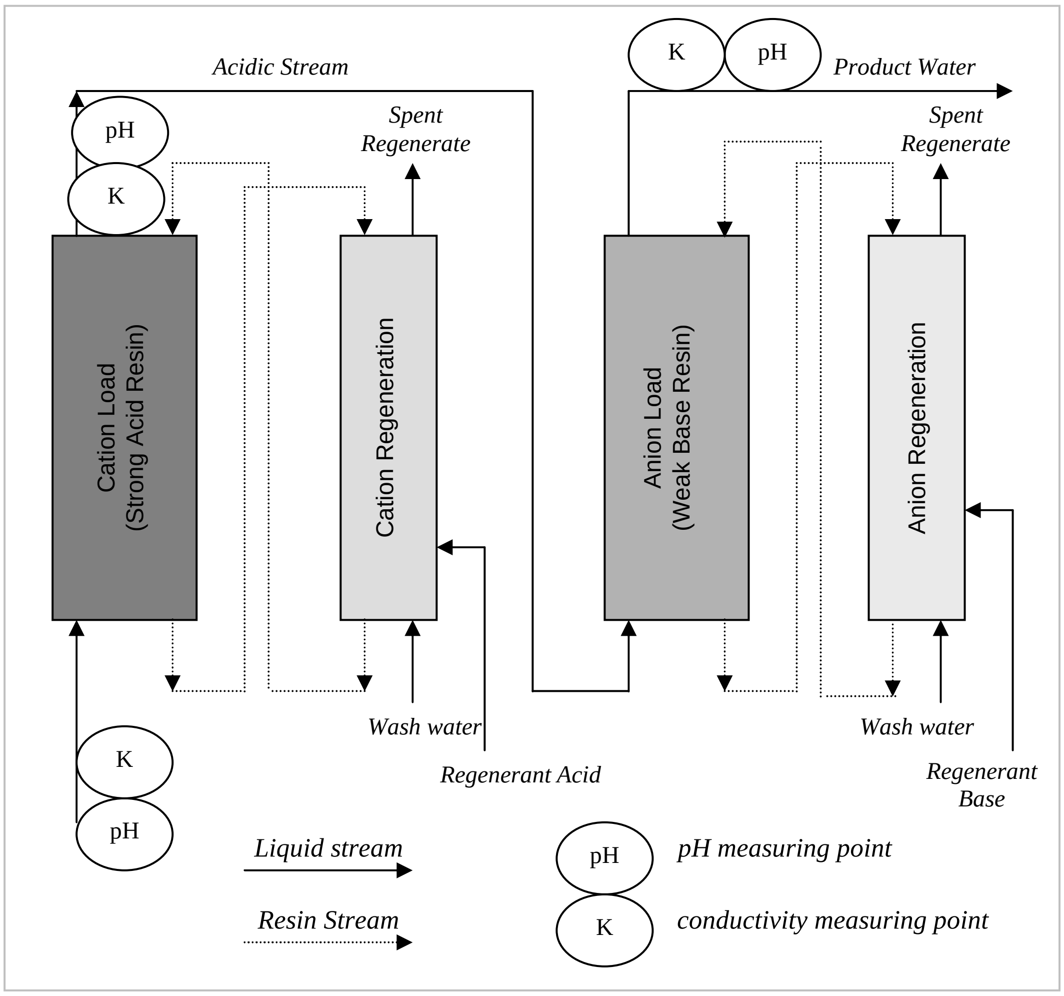


Figure 1.2. **Basic continuous countercurrent ion exchange (CCIX) process configuration**

(Adapted from CCIX University of Cape Town 1982, Hendry, 1982)

The parameter estimation problem solution will be used to determine the parameters of the model every time when the inflow concentration of sodium changes its values. This will be aiming at keeping the optimal control conditions irrespective of the disturbances introduced by the change of sodium concentration (Dube & Tzoneva, 2004: 470–483).

1.7. Delimitation of research

The research work only considers bilinear models of the ion exchange process. No other models and other processes are considered. The research is restricted to the abovementioned four methods for the parameter estimation problem and the two methods for state estimation problem. The research does not include integrated methods of parameter and state estimation.

1.8. Assumptions

The following assumptions are considered:

1) The data obtained from the experiments with the real plant is measured correctly and normalized correctly.
2) The missing data is for a small number of points and can be approximated on the base of the closest points' values.
3) All assumptions done for the derivation of the mathematical model of the counter current multi-stage ion exchange process are based on the application of the first principles of the physical laws and are correct.
4) Grouping of some of the nonlinear model parameters using their mathematical relationship does not change the dynamic behavior of the process model.
5) Initial conditions of the ion exchange process are determined by the value of the input process disturbance and change when the disturbance changes.
6) Iterative implementation of the gradient procedures in the developed methods based on introduction of a very small numbers for stopping criteria brings very small errors in the final ideal results of the calculations.
7) MATLAB/SIMULINK software environment has all functions and models necessary for implementation of the developed methods.

1.9. Research method

The procedure for the research method executed is as follows: literature review, data acquisition, model development, design of methods and algorithms, software development, application of methods and case study investigation, and analysis of the results. Short description is given below.

Literature review: Research into linear and nonlinear systems has been conducted to gain knowledge on the work developed and the current trends of research in this field of study. The research was then narrowed to be able to address the research problem by focusing on methods used for parameter and state estimation in nonlinear systems.

From the literature review, it has been observed that from many different parameter and state estimation problems, none of the studies considered a *nonintegrated estimation* procedure, particularly for bilinear affine in parameters models.

The research problem is then formulated around developing new methods and algorithms for nonintegrated parameter and state estimation for bilinear affine in parameters models. The developed methods should be such as to reduce complexity in the algorithm to help reduce the necessary calculations.

Data acquisition: Data that were obtained from a previous study of the same process (Henry, 1982a; Hendry, 1982e) is used to evaluate and validate the methods and algorithms developed to solve the research problem.

Model development: Based on the bilinear model developed for a different study (Dube, 2002a), data and a new model are reformulated for parameter and state estimation as a nonintegrated problem. Two models have been developed, 1) a bilinear affine in parameters model, and 2) a bilinear nonlinear in parameters model.

Design of methods and algorithms: New methods and algorithms for parameter and state estimation are developed independently of each other. For parameter estimation methods, both above models are used and corresponding methods are developed. For state estimation problem, only the bilinear affine model is considered and state observer and state predicting Kalman filter are developed.

Software: MATLAB software is used for data analysis and for development of software for implementation of the proposed methods. Before the data is used for implementing and testing the validity and accuracy of the developed methods, analysis of these data is done using MATLAB plots.

MATLAB programs were generated for each method based on the accompanying algorithms as a tool for testing and analysis of results. The developed methods and algorithms are tested using the available experimental data for each method developed.

Application of methods and case study investigation: A continuous countercurrent ion exchange process is used as a case study for application and analysis of the developed methods and algorithms. The investigation is done to verify the performance of the developed methods for different conditions of the process and parameters of the algorithms.

Analysis of results: Different experiments are run based on each method and algorithm developed. The results are then collated and a comparative analysis performed.

1.10. Motivation and significance of the research

In developing a model for control and monitoring of a physical plant, the model is considered satisfactory if it captures most aspects without losing the essence of the process and it is not too complex to be used in the practical control strategy. Model parameters must be verified using experimental data, and this is achieved through parameter estimation. In instances where not all states of the process are accessible, it also becomes necessary that those unknown states are estimated. This argument is the motivation behind this research work; the research considers complex nonlinear models that are applicable in real-time process control and monitoring. The aim is to develop the full strategy from model development, through estimation of parameters and the states up until the application of the model.

If one needs to develop a model for real-time control, it is not possible to include all parameters of the process, and therefore only parameters that are of interest to the study are considered. In considering some of the parameters and not others, it is important that a model does not become over simplified and thus losing out on the important aspects of the plant. The developed models allow all or a separate group of parameters to be estimated. Parameter estimation helps in qualifying a model. Based on existing data (usually experimental data) versus data received from the new model, one can estimate some parameters of the model, using different parameter estimation techniques.

State estimation is necessary to develop the optimal control strategy for physical processes. The requirement for the state estimation is that some measurement data is available. Model parameters must also be known or partially known for state estimation to be possible (Kamoun, 2007; Nicholson, et al., 2014; Zhao, et al., 2013). Also in some cases, depending on the nature of the process and the resulting model, it may be necessary to perform combined state and parameter estimation (Bezzaoucha, et al., 2013; Chitralekha, 2010; Tulsyana, et al., 2013). The methods for state estimations developed in the thesis are for real-time implementation of the state observer or the Kalman state predictor. They apply the existing methods to the case of the bilinear models of the process by introduction of a special condition for the bilinear term of the model. These methods can be used also for linear systems when the bilinear term is considered to be zero. The developed observer and Kalman filter can successfully be applied for implementation of the closed loop system controlled by state space controllers.

1.11. Chapter breakdown

The project is formulated over ten chapters as follows:

- Chapter one is the introduction of the project,
- Chapter two is the literature review of nonlinear processes, parameter and state estimation techniques,
- Chapter three deals with the theory of nonlinear systems,
- Chapter four describes the case study, the continuous countercurrent ion exchange process (CCIX) and its model development,
- Chapter five presents experimental data and the reformulation of the CCIX model for parameter and state estimation problem,
- Chapter six describes the parameter estimation methods for the bilinear affine in parameters model,
- Chapter seven presents the parameter estimation methods for the bilinear nonlinear in parameters model,
- Chapter eight is the deals with the estimation observer design for the bilinear affine in parameters model,
- Chapter nine describes the state estimation Kalman filter design for the bilinear affine model,
- Chapter 10 presents the conclusion and deliverables of the thesis.

1.12. Conclusion

The research work proposes developing new methods and algorithms for parameter and state estimation for nonlinear processes. Available estimation methods have been studied and analyzed with the intention of producing improved nonlinear models representation and estimation methods. The main goal is to develop new methods that are not only specific to certain classes of nonlinear processes, but can be extended to other classes of nonlinear systems. The study aims at producing methods that should be applicable for real-time control and monitoring of nonlinear processes. The new methods are based on optimization techniques.

This chapter covered the research methodology followed in responding to the thesis' aim and objectives. The following sections were covered: background to the need for undertaking the research, awareness and statement of the problem, the research aim and objectives, hypothesis and the scope of the project, research delimitations, assumptions, the research method, the significance and motivation for the research and finally, the chapter breakdown.

The next chapter considers comparative analysis of all existing theoretical and practical developments in the field of the modelling, state and parameter estimation of nonlinear processes by a literature review.

CHAPTER TWO

COMPARATIVE ANALYSIS AND DISCUSSION ON THE DEVELOPMENTS IN THE EXISTING LITERATURE FOR MODELLING AND ESTIMATION OF NONLINEAR SYSTEMS

CHAPTER TWO

2. COMPARATIVE ANALYSIS AND DISCUSSION ON THE DEVELOPMENTS IN THE EXISTING LITERATURE FOR MODELLING AND ESTIMATION OF NONLINEAR SYSTEMS

2.1. Introduction

This chapter deals with the comparative analysis between latest developments in the three major sections involved in the subject matter of this thesis, that is, *methods for modelling*, *parameter and state estimation for nonlinear processes*. The focus is on nonlinear systems in general, and on how parameter and state estimation in such systems are performed.

The chapter also deals with the important aspects of estimation techniques, that is, optimization theory, and approximation principles, more especially the latest developments in nonlinear systems. Complexity involved in these topics is further extended by the fact that the nonlinear systems are a subject matter; though, the system of interest is a bilinear model with *affine parameters*, the approach to the solution has been based on nonlinear systems in general.

Further, to simplify the presentation of this chapter, the chapter is divided into three main parts:

1) criteria for comparison of the existing methods for parameter and state estimation covered in Section 2.2,
2) the literature search in Section 2.3 and
3) literature review for parameter estimation, state estimation and combined state and parameter estimation in Section 2.4.

2.2. Criteria for comparison of the existing methods for parameter and state estimation

Almost all, if not all physical processes (including natural occurring or biological, and industrial systems) are nonlinear in their behaviour; this is mainly due to internal interactions (dynamics) within these systems, borne out of their existence (Salhi & Kamoun, 2015). Because of this dynamic behaviour, industrial processes also tend to exhibit high nonlinear characteristics in their operations. Generally, models developed for nonlinear systems tend to be very process specific. Table 2.1 below shows the criteria used to compare methods for parameter and state estimation procedures in the thesis.

Development in the estimation of the parameters or states has evolved from work done in the early sixties up to the end of the twentieth century; work done in the sixties and early seventies was based on dynamic linear models, Figure 2.1.–Figure 2.6. At the start of the twenty first century majority of the work focused on nonlinear models due to nonlinear dynamics and nonlinearities contained within physical systems (Demirbas, 2015; Ganguly, Sairam & Saraf 1993; Keesman, 2000; Li, Mu & Zuo, 2016; Salau, Trierweiler & Secchi, 2014).

Estimation (of parameters or states) in nonlinear models can be solved by *linearization*, *optimization,* or *statistical methods*. Linearization process does introduce some system error in the corresponding estimates of parameters or states. These errors should be taken care of when formulating the estimation problem. Optimization and statistical methods are basically search algorithms used to find a local or a global minimum of the given minimization criterion.

The stochastic search methods are easy to implement but suffer the drawback of no guarantee to locating a global minimum within finite computations. These methods are based on *probabilistic approaches*. On the other hand, the deterministic search methods do provide some assurance in terms of locating a global minimum over a finite number of computations.

Generally, in estimation procedure for parameter estimation or state estimation, the idea is to seek estimates of the real parameters or states that would minimize an objective function; this objective function is described by the error difference between the model output (based on parameter estimates) and the measured data output, or error difference between the model output (based on estimated states) and that of the measured data.

When such a procedure is used to solve for unknown parameters or states such a formulation becomes an optimization problem. This allows application of optimization techniques to solve for unknown parameters or states in a model.

Optimization techniques used for estimation problems are generally: 1) *least squares methods*, 2) *Maximum Likelihood* (ML) *methods* and 3) *Bayesian methods*. Other well-known methods include; *linearization transformation*, *minimax deviation*, *minimum* Chi-Squared (χ^2) method, and *Pseudomaximum Likelihood* (Bard, 1970:47–82).

Table 2.1: **Criteria for comparison of the methods for parameter and state estimation procedures**

Criterion	Description			
	Subdivision	Considerations		
1. Type of plant (according to variables)	1. Linear model	linear variables		
	2. Nonlinear model	nonlinear variables		
2. Type of plant (according to parameters)	1. Linear	linear parameters only		
		linear parameters	nonlinear parameters	
	2. Nonlinear	nonlinear/linear parameters	nonlinear/nonlinear parameters	
3. Type of model	1. Transfer function			
	2. Differential equation			
	3. ARMA			
	4. State space			
	5. Description	Deterministic	Stochastic	
			(type of noise considered)	
4. Method used for estimation				
5. Criterion for optimization				
6. Error of estimation				
7. Online time for estimation				
8. Novelty of the method				
9. Drawbacks of the method				
10. Numerical methods used for calculation	System order			
11. Application	1. Process			
	2. Purpose of estimation	Monitoring	Control (MPC)	Fault detection
12. Real-time implementation	Time for online estimation	Simplicity	Applicability	
13. Method of measurement data	Data processing	Online	Offline	
14. Software used	1. Simulation software			
	2. Data acquisition software			

Chi-Squared is sometimes considered an extension of least squares method and Pseudomaximum Likelihood, the extension of Maximum Likelihood method (Ching, et al., 2006; James & Petersen, 1998; Ungarala, Chen & Li, 2006).

Currently, there is also a lot of work being done in complex nonlinear systems such as weather prediction, ecosystems, seismic systems, power systems and networked control systems in terms of modelling and estimation (state estimation and parameter estimation). Most of these systems, unfortunately, are extremely dynamic, highly nonlinear, high model-dimensioned, and do not have much available data and also, have low quality data where data is available. These complications in estimation have led to development of new methods, the latest optimization methods, such as the evolutionary or swarm methods (Salau, et al., 2014; Tashkova, et al., 2012).

Comparitive analysis of the existing methods for parameter and state estimation is done according to the criteria in Table 2.1.

2.3. Literature search

In doing literature search, each key concept in the thesis title, *modelling of nonlinear systems*, *estimation*, *parameter estimation* and *state estimation* were searched individually. '*Modelling of nonlinear systems*', in this case, produced most results, and '*estimation*' produced the next best results, followed by '*parameter estimation*' and '*state estimation*', in that order. On searching the terms, '*modelling nonlinear systems*', and '*identification and estimation*', this search produced best results overall in terms of number of papers available. In terms of relevancy, '*modelling nonlinear systems*' yielded the best results.

Based on the method of solution for the thesis, searches were also conducted for the method of solution keywords: '*parameter estimation*', '*state estimation*', '*observer design*', '*Kalman filter'*, and '*combined state and parameter estimation*'. '*Observer design*' search produced most results, also in terms of relevancy, with '*parameter estimation*' producing second best and '*Kalman filter*' produced best results in terms of relevancy, followed by '*state estimation*'. '*Combined state and parameter estimation*' search did produce relevant results when the search was focused on it alone, but overall not so many papers were found.

Graphical presentation of these results are shown below, with each section showing how research work has progressed from early fifties in some cases up to date, Figure 2.1.–Figure 2.6. The graphs are organized according to the type of estimation problem to be solved in the thesis: 1) parameter estimation, 2) state estimation, 3) observer design, 4) Kalman filter, 5) other filtering methods, 6) parameter and state estimation, and 7) combined state and parameter estimation.

2.4. Literature review

2.4.1. Review of existing literature for nonlinear systems, modelling and estimation in general (the current state)

The aim of this section is to the review available literature on nonlinear systems in relation to parameter estimation, state estimation and or both, but it is by no means a generalized review on nonlinear systems. Due to complexity associated with the nonlinear systems, there is no one definitive solution of parameter estimation or state estimation problems for these systems. Each estimation solution is based on a model type as applicable to a certain class of (nonlinear) systems or even worse, on a specific system model; "*estimation in nonlinear systems is extremely difficult*" (Biagola & Figueroa 2002:4777; Ching et al., 2006:396; Inoue, et al., 2008:677; Julier & Uhlmann, 2004; Moradkhani, et al., 2005; Norgaard, Poulsen & Ravn, 2000:1627).

One could consider many different aspects of the model and decide on the direction of the solution. There are different combinations of system characteristics that make a model nonlinear. The two main ones are nonlinearity with respect to parameters or variables. This is further subdivided into models with parameters that are a mix of nonlinear and linear combination within a model.

Depending on the parameters of interest, if any nonlinear element of the process will be included in the model formulation, then the system model is considered nonlinear – Table 2.1. There is a possibility that a system may contain nonlinear parameters, and if these parameters are not part of the estimation problem formulation and only the linear parameters are considered, such a system can be solved using linear methods.

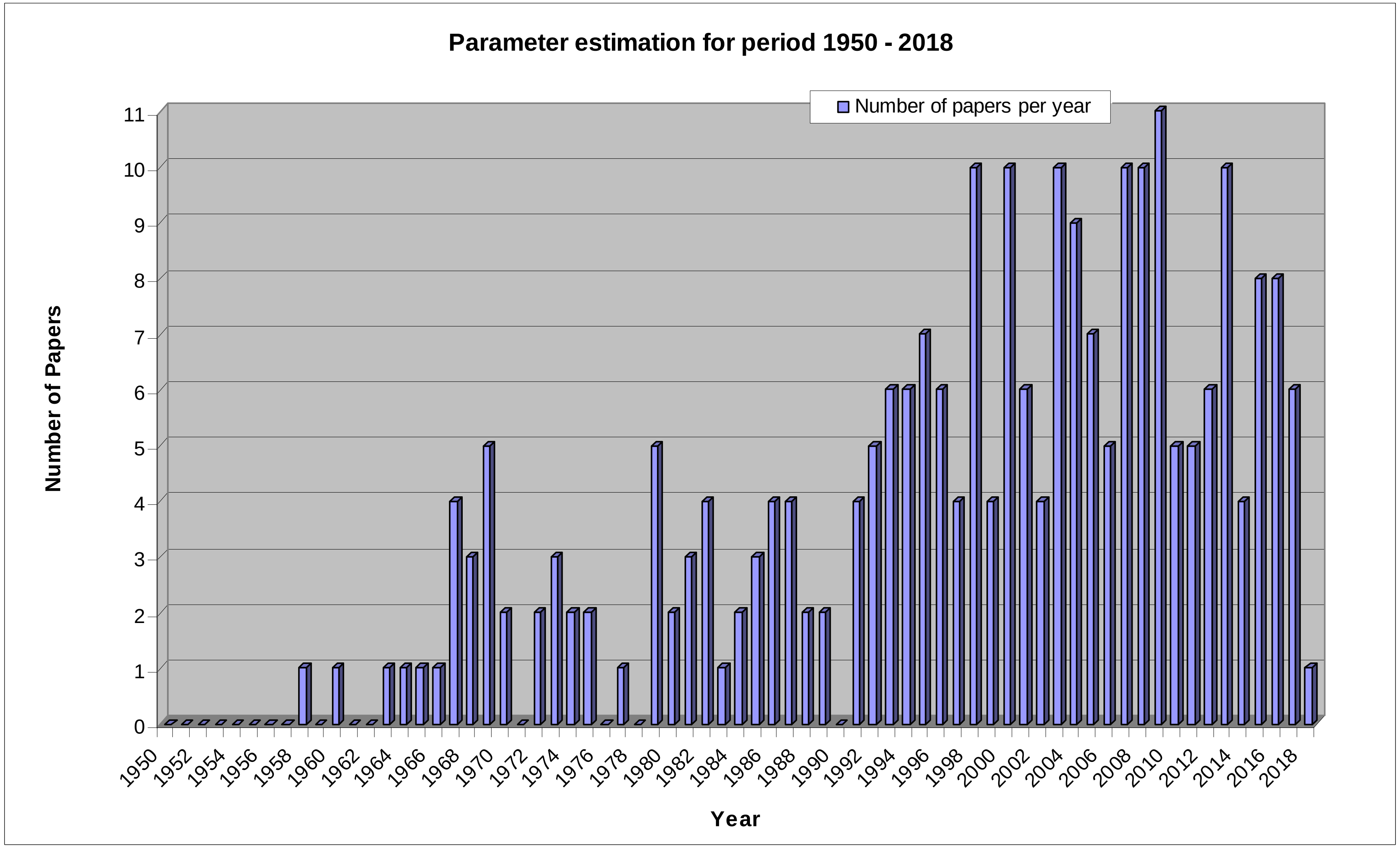


Figure 2.1. Number of the relevant papers for the parameter estimation methods for the period 1950 to 2018

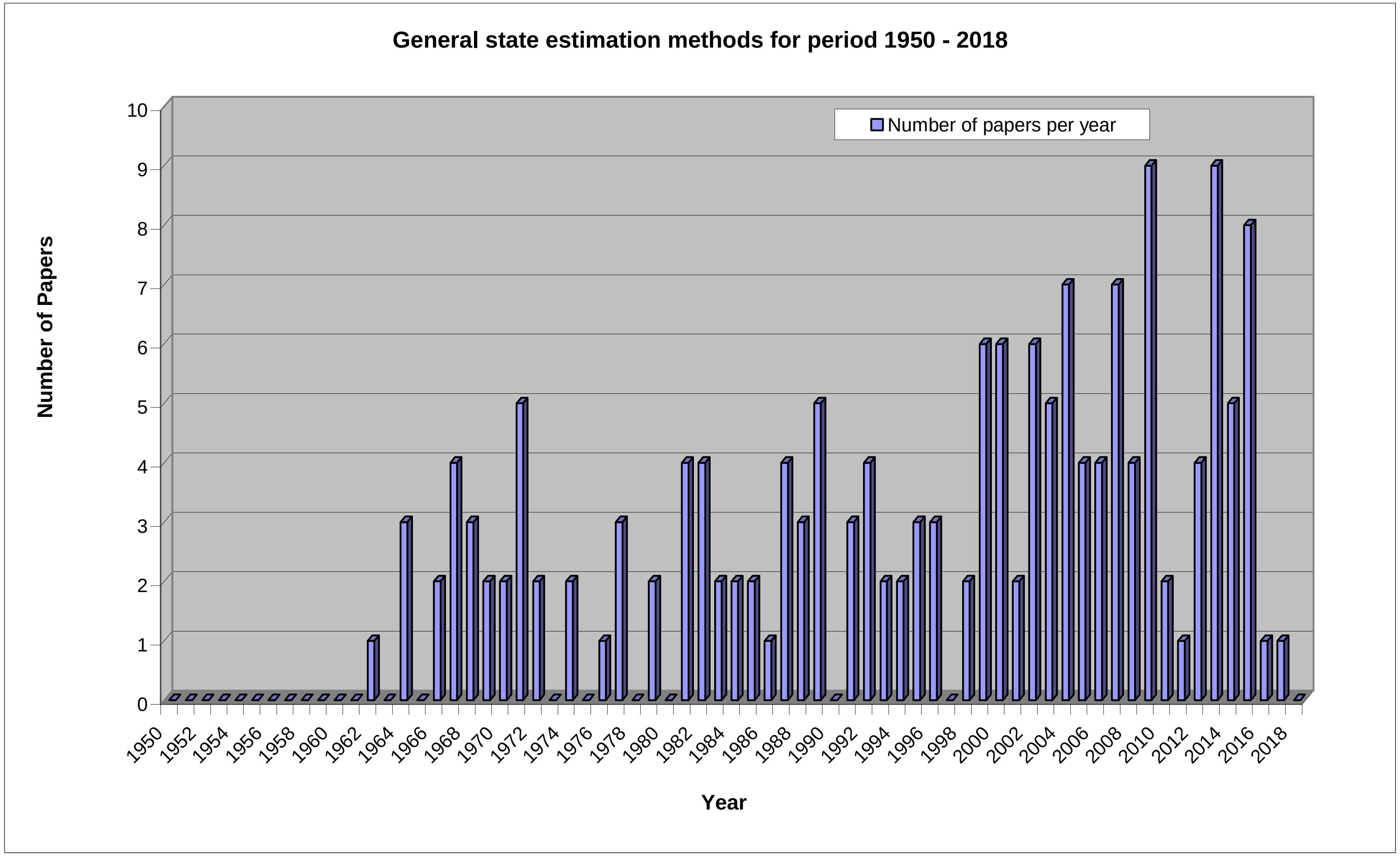


Figure 2.2. Number of the relevant papers for the state estimation methods for the period 1950 to 2018

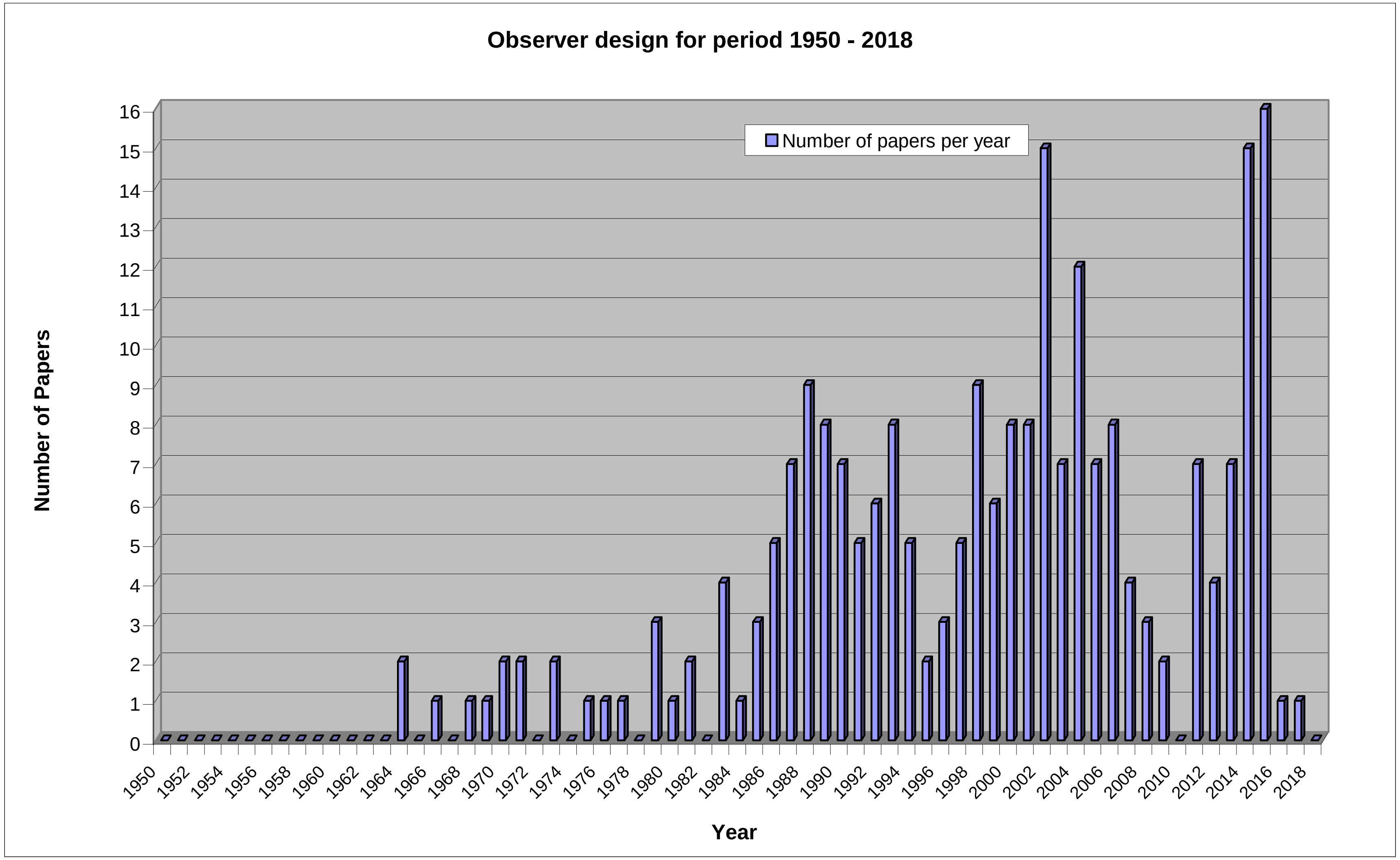


Figure 2.3. Number of the relevant papers for the observer design methods in the period 1950 to 2018

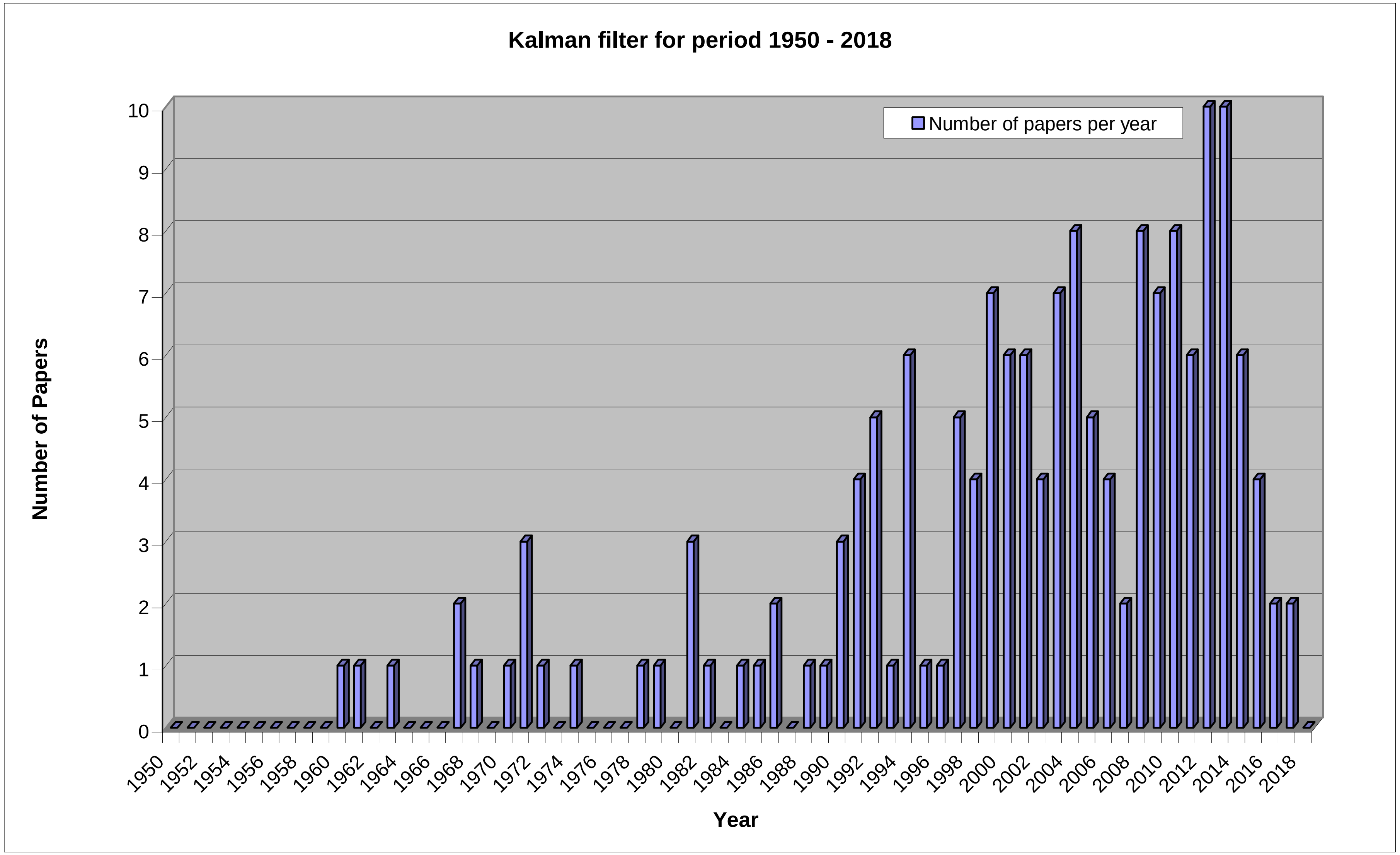


Figure 2.4. Number of the relevant papers for the Kalman filter design methods in the period 1950 to 2018

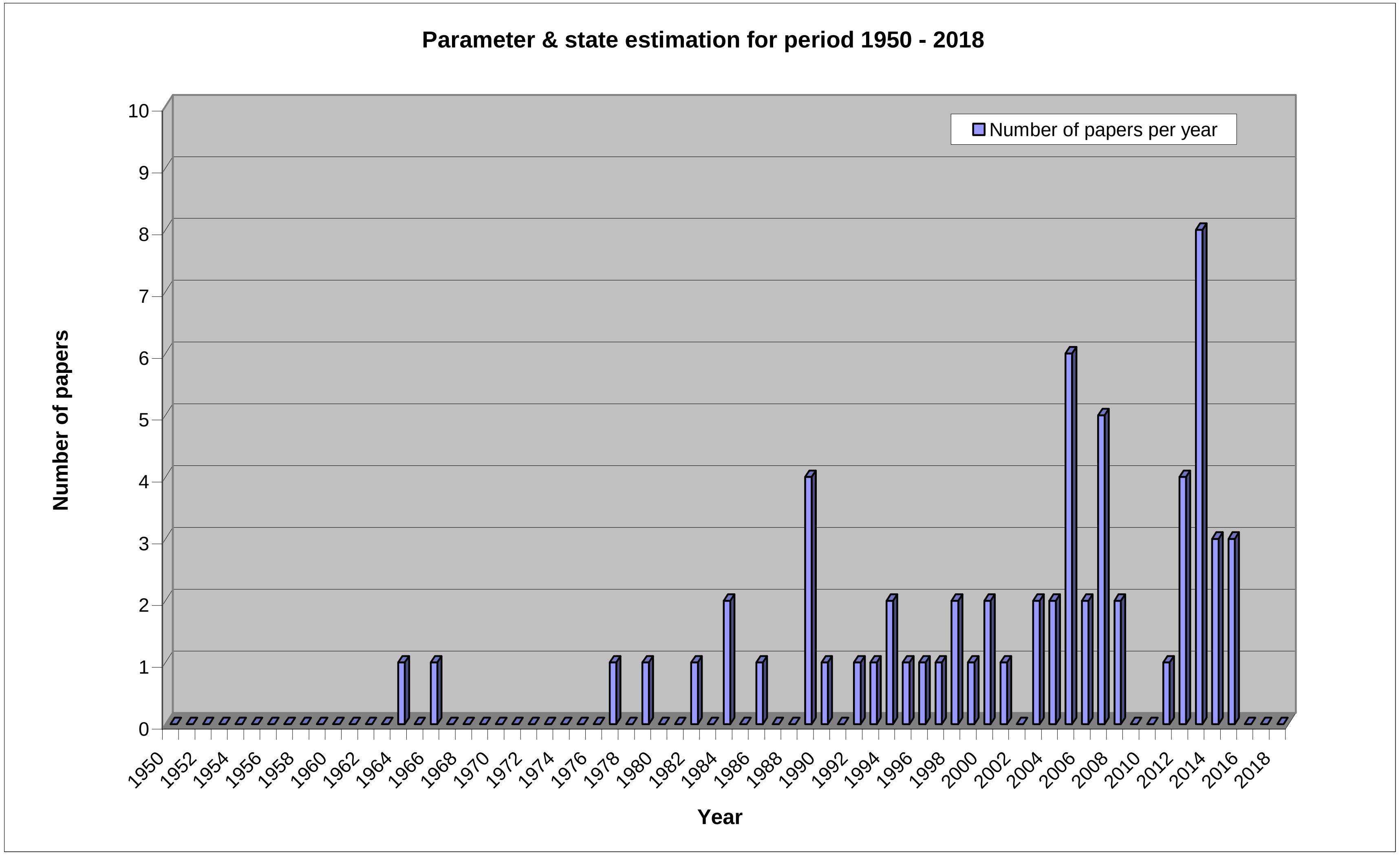


Figure 2.5. Number of relevant papers for the parameter and state estimation methods for the period 1950 to 2018

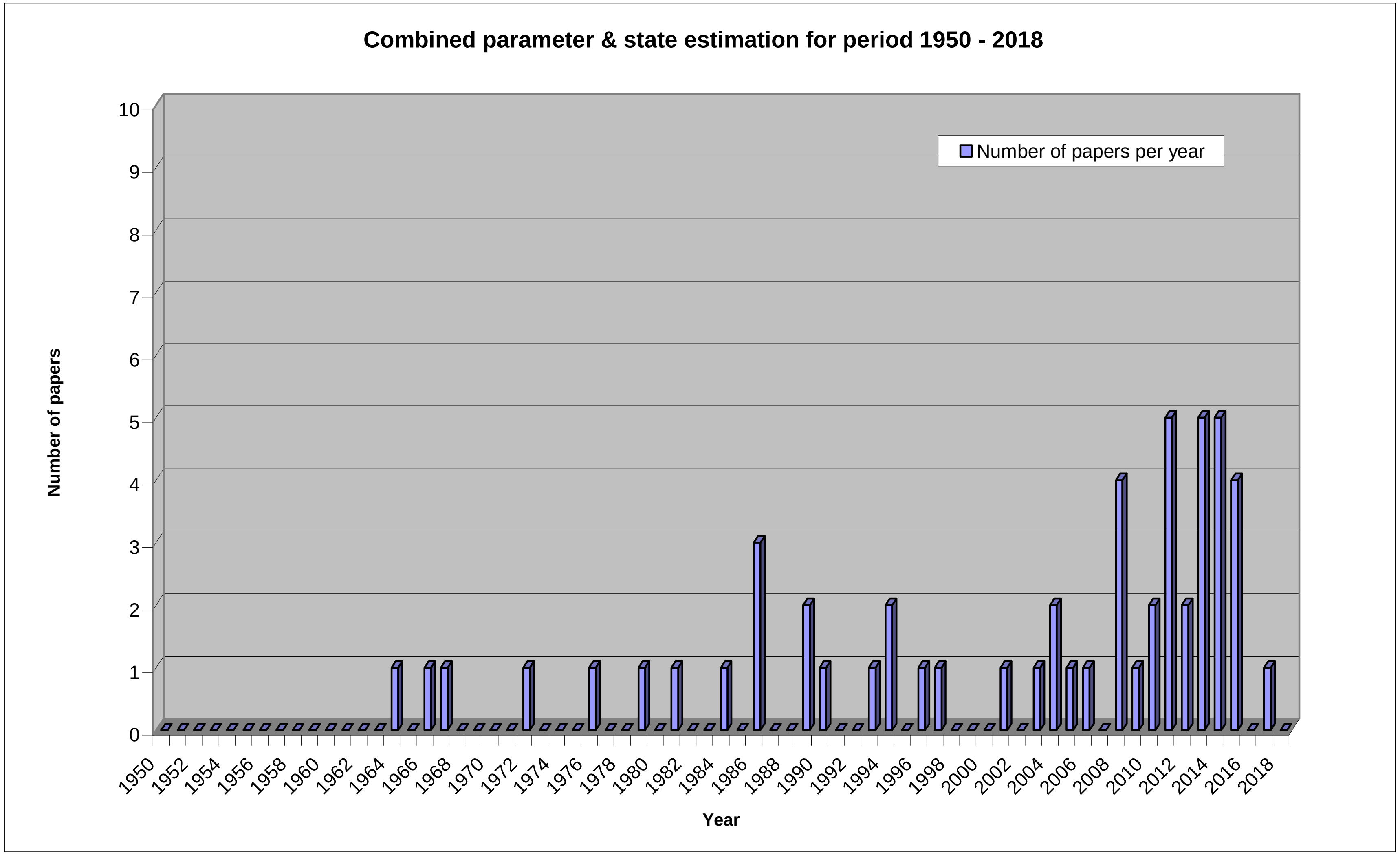


Figure 2.6. Number of relevant parers for the combined state and parameter estimation techniques for the period 1950 to 2018

This will have to be accompanied by relevant assumptions (Biagiola & Figueroa, 2002; Khodadadi & Jazayeri-Rad, 2011; Rolain, 2005; Schoukens, et al., 2005; Zhang, et al., 2010; Keesman, 2000). If the proposed solution considers the part of the model with nonlinear variables, the system will be solved based on nonlinear methods.

Another possible type of process model is that of bilinear systems, where two model variables are in a product relation, usually the input and the state variables. Depending on how parameters of such a model enter the model, linearly or in a nonlinear fashion, the solution of parameter estimation may be based on linear or nonlinear methods. And for state estimation, the bilinear part can also be solved using linear or nonlinear methods depending on how the state variable is presented in the model.

2.4.2. Discussion

The necessity of estimation of parameters and or of states of a real process forms part of an overall goal of a control system analysis, design, operation, control and monitoring, and optimization of the system. The main objective is the optimization of the system overall's performance.

According to Alaei, Salahshoora and Alaei (2010), for stabilization of processes in the presence of nonlinearities and uncertainties, there are a few design techniques that can be applied; these include Model Predictive Control (MPC) which has become the most utilized technique in control strategies in the recent decades. Analysis and design methods for linear systems can always be generalized to nonlinear systems (Jamel, et al., 2010; Ladeveze, 2016; Thangavel, Paulen, & Engell, 2016; Rad & Hancu, 2017; Schei & Singstad, 1998).

Further, it is known from control theory and observation principles that state estimation is closely associated with control and stabilization (Moireau, Chapelle & Le Tallec, 2008). The state of a system is only determinable if model parameters are known. This creates the need for estimation of both parameters and state variables in cases where they are unknown.

2.4.3. Review of the existing literature for parameter estimation

Parameter estimation has grown to become a very rich field in engineering and in other fields such as science and economics (Elliott & Hyndman, 2007). Parameter estimation is the process of determining unknown parameters using the error difference between data of a system's measured output and that produced by the system's model output. The measured data are considered the real values. If parameters are well tuned, these two data sets should match. Since the concern is the minimization of the error difference between data sets, parameter estimation is basically an optimization problem (Tashkova, et al, 2012). This is also the reason why the *parameter estimation* problem is sometimes referred to as, *parameter fitting*, *model calibration* or *the inverse problem*. Ding (2014) states that in control systems, parameter estimation is the basis for 1) controller design, 2) filtering, and 3) state estimation.

Most nonlinear parameter estimation problems are formulated as nonlinear optimization problems. The two most important aspects of such parameter estimation problem are the *type of mathematical model* and *the type of objective function to be minimized*. The interest is to consider the latest developments in parameter estimation methods, the criterion used for estimation procedure and the type of error of estimation considered (Zhao, An & Xu, 2012:1696; Linga, Al-Saifi, & Englezos, 2006; Gau & Stadtherr 2000:631; Hassan, et al., 1982). Many different procedures and algorithms have been developed to solve the parameter estimation problems for different processes; these methods are highlighted in the text. They can generally be grouped under three main estimation techniques, *error methods*, *maximum likelihood methods* and *Bayesian methods* (Schindler & Phillips, 2009; Young, 1981).

Strejc (1980) and Young (1981) give some major historical aspects of the *least squares method*. In least squares sense, Strejc (1980) defines parameter estimation as a technique used to determine the '*most probable*' value of an unknown quantity for which the sum of squares of the difference between the measured values and that computed values using these probable values is minimized.

Variations of least squares methods for practical use have been proposed in the literature, these include: *the recursive least squares* methods, *instrumental variable method*, *generalized least squares*, *extended least squares, square root filtering method*, *recursive prediction error least squares method*; *implicit least squares algorithm* applicable to models with regression terms that are subjected to noise, *adjusted least squares* (ALS), *multi-fold least squares* etc., (Cimpoesu, Ciubotaru & Stefanoiu, 2013; Hassan, et al, 1982; Iqbal, Bhatti, Iqbal, Khan & Kazmi 2009; Young, 1981; Strejc, 1980; Wang, Dai & Ding, 2009; Zhu, 2005).

Wang, Wang and Ji (2017) propose a very interesting novel parameter estimation method. Their method has two stages; the first stage is based on *bias-*

eliminating least squares method and a second stage that uses *singular value decomposition method*.

Other least squares methods presented include, a *two level algorithm* for parameter estimation using *multiple projections*; the *total least squares* (TLS), also known as *orthogonal regression* in statistics, this method is based on *singular value decomposition* (SVD) technique; the *kernel partial least squares* formulation, *quadratic partial least squares*; *partial least squares combined with neural networks*; *dictionary-based estimation; noise compensation technique* methods have been introduced recently (Austin, Ash & Moses, 2013; Chen & Suter, 2007; Zhang, et al, 2010, Chianeh, Stigter & Keesman, 2011; Currier & Danai, 2011; Khalil, et al., 2015; Lagrange, et al., 2006; Lee, et al., 2004; McCusker, et al., 2011; Zhang, et al., 2010).

Numerical methods

Ding (2014) states that numerical methods are generally used for solving matrix equations thus the application in determination of model parameters for dynamic systems. Typical numerical methods include: *gradient search*, *least squares* and *Newton* methods. These methods can be further divided into *iterative* or *recursive* techniques (Ding, 2014; Schittkowski, 1994; Salhi & Kamoun, 2015). Ding, Liu and Liu (2016) propose a recursive least squares method for Wiener-type nonlinear models, while Kazemi & Arefi (2017) present a fast iterative recursive least squares method for Wiener-type highly nonlinear model. Liu, Xiao and Ding (2013) presented a Newton based iterative parameter estimation method for Wiener nonlinear systems.

Local and global optimization techniques

Many global and local optimization techniques have been proposed by different authors: Esposito and Floudas (2001), Faber, et al. (2007), Linga, et al. (2006), Luus (2001) and Zarzini, McAuley and McLellan (2008a). These methods include sub-techniques such as the *branch-and-bound* (B&B) *algorithm* that is derived from *deterministic global optimization* (Li, et al., 2015; Zarzini, McAuley & McLellan, 2008a), the *incremental identification procedure* Michalik, et al., (2009a) etc. Other work under optimization techniques include that of Esposito and Floudas (2000), Michalik, et al. (2009a), Esposito and Floudas (2001), Chianeh, et al. (2011). Esposito and Floudas (2000) and Michalik, et al. (2009a) considered the problem of *multiple local minima in optimization technique*s as a solution to parameter estimation.

Li and Ding (2013) propose a method that transforms a complex nolinear optimization problem into a linear or a simpler nonlinear optimization problem.
Other optimization based methods include: *quasilinearization method*, the *Marquardt method Levenberg-Marquardt algorithm* which is a *Marquardt method* incorporated to the quasilinearization algorithm to improve on convergence region. The *Levenberg-Marquardt algorithm* is defined as a nonlinear parameter estimation learning algorithm that is known to converge quickly and accurately; the *branch-and-bound* (B&B) *algorithm* that is derived based on *deterministic global optimization*; the *incremental identification procedure* (Remaker, Smith & Murrill, 1970; Zarzini, et al., 2008a, Michalik, et al., 2009a; Shawash & Selviah, 2013; Simunek, Wendroth & van Genutchen, 1998).

Maximum likelihood methods

The *maximum likelihood optimal minimization technique* is one of the best of minimization techniques. It always guarantees a global minimization. Many authors have attempted parameter estimation problems using direct maximum likelihood or using it as part of the solution. In cases where *noise probability density function* is known, the *maximum likelihood technique* is the most viable option, or in cases where *prior probability density function* is available for the parameter vector, the *maximum a posteriori estimation* is the best method to use. The *bounded-error estimation* methods should be applied in cases when only bounds are available for measurement noise and state disturbances. Therefore, to ensure proper parameter estimation, the parameter set has to be defined to be consistent with available model structure, available measurement data and noise bounds (Britt & Luecke, 1973; Johansen, Doucet & Davy, 2006; Kok, et al., 2015; Schon, et al., 2006).
Other maximum likelihood techniques include, the *Estimation Maximization* (EM); *approximate* maximum *likelihood estimation* (AMLE); *maximum a posteriori* (MAP) *estimation*; *contrast functions* for static parameters; *pseudo-likelihood function* which is maximized using EM type algorithms (Andrieu & Doucet, 2003; Andrieu, Doucet, & Tadic, 2005; Hanson, 2000; Zarziri, et al., 2008a; Zarziri, McAuley & McLellan 2008b; Yang, et al., 2001; Wills, Schon, & Ninness, 2008).
The *Estimation maximization* (EM) is an estimation method that calculates *maximum likelihood* and *Bayes modal parameter estimates* in cases where some data may be missing, the algorithm is performed over two steps, 1) the *expectation step* and 2) the *maximization step*. EM is also generally used to

provide maximum likelihood estimates in nonhomogeneous trees in dynamic trees models.

Heuristic methods

Heuristic optimization methods are general purpose optimization methods based on *empirical evolutionary rules* that resemble optimization in natural systems (Abonyi, et al., 1999; Oh, et al., 2004; Schwaab, Biscaia Jr., Monteiro & Pinto, 2008). Heuristic is referred because the optimization solution does not guarantee the exact solution.

A number of heuristic methods are available and have been suggested for solving nonlinear parameter estimation problems. These include: *simulated annealing* (SA), the *genetic algorithm* (GA), *particle swarm optimization* (PSO) *algorithm*, usually applied in complex stationary models, *hybrid differential evolution* (DE) algorithm for solving kinetic parameter estimation and *dynamics on trees* methods (Schwaab, et al., 2008; Wang, Su & Jang 2001; Zhao, et al., 2012). Kannan, et al., (2000) presented an algorithm for estimation of the parameters for a *multi-scale stochastic process* using *scale-recursive dynamics on trees*.

Observer based parameter estimation

In some cases, parameters are modelled into models such that they are entered as a subset of the state variables and once parameters to be estimated are chosen, a state estimation algorithm is used (Al-Hosani & Utkin, 2012; Safdarnejad, et al., 2015; Zhou, 2013; Farza, et al., 1998). Iqbal, et al. (2011) introduced an algorithm for robust parameter estimation of *uncertain nonlinear dynamic systems* using a *variable-structure differentiator observer*.

Other observer based methods are presented in different settings; Zhou (2013) presented a parameter estimation method and an adaptive observer design for a class of uncertain systems; due to nonlinear complexities, an *adaptive law* for parameter estimation is introduced (Na, et al., 2011). Other authors considered a large class of time-varying nonlinear systems (Faber, et al., 2007; Kenne, et al., 2008; Ortega, et al., 2015; Schwaab, et al., 2008).

Parameter estimation in conjunction with other estimation methods

In some cases the focus of parameter estimation is based on the type of the model whether it is linear, nonlinear or bilinear; deterministic or stochastic (Alaei, Salahshoor & Alaei, 2010; Bregon, Biswas & Pulido, 2012; Cari & Alberto, 2011; Faber, et al., 2003; Faber, Li & Wozny, 2004; Faber, et al., 2007; Ganguly, et al., 1993; Kenne, et al., 2008; Lototsky & Rosovskii, 2000; Meskin, et al., 2013; Kieffer, Walter & Simeonov, 2003; Michalik, Hannemann & Marquardt, 2009b; Poyton, et al., 2006; Varziri, et al., 2008a; Mahyuddin, et al., 2012; Na, et al., 2011; Liu, Xu & Wang, 2012; Soroush, 1998; Wang, Su & Jang 2001).

Lin, Zhai and Antsaklis (2003) suggest that practical systems' parameters are always affected by some *unknown probability time-varying perturbations* which suggest the need to deal with uncertain parameters when dealing with real systems. Their work dealt with a class of uncertain linear systems affected by both parameter variations and exterior disturbances.

Different moving average methods have been also studies by many authors, Halim, et al., (2009) present a space-time *autoregression moving average* (STARMA) *models family*, generally used to describe *stationary* or *weak stationary space-time processes*.

Chen and Suter (2007) used *a low-rank matrix approximation* to solve a general parameter estimation problem based on a *bilinear approach*. He, Yuan and Mu (2011) developed an estimation method for *affine transformation parameters*. Iqbal, et al., (2009) presented a method for parameter estimation of *uncertain nonlinear systems* based on accurate and robust derivatives using *high-order sliding modes*. Xiang, Mueller and Cheng (2007) proposed an estimation method that consists of three algorithms in a sequence: 1) *least squares estimates*, 2) *grid searching* and 3) *the gradient search algorithms*. Schön, Wills and Ninnes (2006) introduced a different parameter estimation technique for a general class of *nonlinear dynamic* systems, using maximum likelihood framework. (Bonilla, et al., 2009) presented a convex approach for parameter estimation involving parameter *affine dynamic systems*.

Kar, Moura and Rawanan (2012) studied *distributed static parameter estimation in sensor networks* with nonlinear observation models and noisy inter-sensor communications. Parameter estimation in frequency analysis were considered by Yang, et al., (2001), Xiang, et al., (2007), and Kim (2013). Meng, Li and Veres (2010) used parameter estimation in an *aerodynamic system for aerospace system modelling* using measured data from flight test. Some work on *biological and chemical systems* has been presented, (Alonso, et al., 2000; Wu & Mu, 2009; Ubeda, et al., 2012; Chianeh, et al., 2011; Simunek, et al., 1998) A psychological and educational measurement theory called the *item response theory* (IRT) which is an improvement from the *classical test theory* (CTT) was presented by Wang, Chen and Ma (2010) and Henson (2000).

On-line parameter estimation methods

A number of *online parameter estimation* solutions are also presented by a number of authors (Bregon, et al., 2012; Fujii & Karoda, 2011; Ravindra & Gudi, 2010; Badwe, et al., 2010; Tatiraju & Soroush, 1997; Tatiraju & Soroush, 1998; Schei & Singstad, 1998; Sirohi & Choi, 1996; Zhang, et al., 2010). According to Tatiraju and Soroush (1998) existing online parameter estimation include: 1) *prediction-error-based methods*, 2) parameter *estimation via state estimation*, 3) parameter *estimation via online optimization*, and *calorimetric methods* (for kinetic and thermal processes).

Software implemented and real application parameter estimation solutions

Alaei, et al., (2010) used the HYSYS *real-time simulation software* to collect data of a distillation column. From the obtained data, they applied MATLAB software program for identification and application of their predictive control schemes. Bengea, et al. (2011) presented a method for demand-response control of a building system and district HVAC systems.

Discussion

Initially, parameter estimation was mostly found in identification where models are developed using experimental data, mostly this work could be defined as *parameter identification*. With growth in modelling of physical processes, this led to development of parameter estimation techniques based on first principle of systems' energy transfer.

As it can be seen in the work presented in this section, parameter estimation is almost applicable to vast fields of engineering and non-engineering fields. In control engineering, it seem to have taken the centre stage, and it is further applied in different kinds of control system's requirements and properties, including; *real-time control*, *nonlinear property* of most models. The calculation algorithm divergence issues are the main indicator if the proposed algorithm will be a success or a failure. Parameter estimation methods have since evolved into including other estimation methods as means of solution.

Many new techniques based on heuristic solution have been developed, and seem to cover even most complex nonlinear systems, e.g. the *particle swarm optimization algorithm* and its variations. Models that consider *measurement errors* are known as *error-in-variables*, with other variations such as *bilinear multivariate errors-in-variables* model, *interval analysis* etc., (Gau & Stadtherr 2000; Kukush, Markovsky & van Huffel, 2003; Esposito & Floundas, 1998;

Lagrange, et al., 2006) presented a system of parameter estimation in cases where the *input is not measurable*.

Rumschinski, et al. (2010) investigated the difference between parameter estimates of a continuous-time nonlinear model and that of a corresponding nonlinear discrete-time approximation of the same continuous-time nonlinear process model (Vajda, Valkos & Godfrey, 1987; Rumschinski, et al., 2010).

2.4.4. Review of the existing literature for the state estimation problem

In physical systems, it is often a necessity to estimate unknown parameters and reconstruct states that cannot be measured due to: 1) *lack of sensors*, 2) *unreachable positions in a plant*, 3) *mixtures that are impossible to measure* or 4) *measurements that may be too costly to perform* (Bogaerts, 2008; Hulhoven, Vande Wouwer, Zhu & Pagilla, 2003; Mangold, et al., 1994; Moradkhani, et al., 2005; Patwardhan, et al., 2012; Shahrokhi & Fanaei, 2001; Sundarapandian 2011a; Tyukin, et al., 2008).

In such cases the unknown model states can be reconstructed using approximation techniques. This suggests that state estimation is one of the major aspects of analysis, design, and control and monitoring of physical processes (Afshari, et al., 2015; Dunik,et al., 2015; Chen, et al., 2004; Ching, et al., 2006).

States in physical processes define the process's behaviour and carry information about the dynamics of the process, which thus dictates the structure of the control system to be implemented. State estimation techniques, as it is the case in parameter estimation can be broadly divided into three main categories; the *direct*/*error* methods, *maximum likelihood* (ML) methods and *Bayesian* methods (Nicholson, et al., 2014; Kottakki, et al., 2014).

Generally, the state estimation problem for continuous time linear systems with Gaussian noise is solved using a Luenberger observer design, or the Kalman filter for discrete time linear systems. The extended Kalman filter is best suited for discrete time nonlinear systems. Pure nonlinear systems state estimation problems are best solved using Bayesian based methods (Ahmed, 1994; Lopez, 2012; Norgaard, et al., 2000). According to Robertson and Lee (1995) nonlinearity and prior knowledge requirements for state estimation can be addressed by considering state estimation problem as a least squares problem.

Linearization and approximation techniques

Statistical approximation techniques use a model and measurement functions are represented by polynomial series expansion. The idea is to find the minimum of mean square error (MSE) between the system model and the polynomial. The parameters are determinable by taking derivatives of mean square error with respect to each coefficient (Muske & Edgar, 1997). The state *conditional probability density* of the system is required to determine the solution of the expectation operation. Generally the probability density is not easily available and approximations must be used. The most general approximation is known as the *statistical linearization* or *quasilinearization* (Dunik, et al., 2015; Feng, et al., 2015; Muske & Edgar, 1997).

Norgaard et al., (2000:1627) proposed a method that uses *polynomial approximations* to determine state estimates. The novelty of this technique is the fact that the polynomial approximations take care of the uncertainty of the state estimates, whereas Taylor approximation based methods for nonlinear systems, like the extended Kalman filter, consider only the current state estimate. The method also considers the *stochastic decoupling techniques* to introduce the measurement noise.

Maximum Likelihood methods

A maximum likelihood (ML) estimation criterion uses statistics theory to estimate an unknown quantity given observation measurements. It is customary for maximum likelihood procedure in linear dynamic system to use *gradient methods* to determine the unknown quantity. Such a solution to estimation may have some implementation limitations; and another possibility is to use expectation maximization (EM) procedure to calculate maximum likelihoods (Hansson & Wallin, 2012; Schon, et al., 2006; Siouris, 1996; Zia et al., 2008). The maximum likelihood procure has the advantages of requiring no *a priori* knowledge about the unknown quantitiy, requires minimal statistical information and produces unbiased estimates (Kok, et al., 2015; Siouris, 1996).

Many different maximum likelihood estimation algorithms have been produced for many diiferent models. Most of these solutions use either the expectation maximization (EM) or the maximum aposteriori (MAP) properties. Khodadadi & Jazayeri-Rad, (2011) state that the extended Kalman filter is one of the best methods for generating approximate maximum likelihood estimates in nonlinear discrete-time systems.

Bayesian estimation techniques

The *Bayesian estimation* technique constructs the conditional probability density function of the state by combining historical information and measurements data according to the *Bayesian rule*. The problem of *Bayesian state estimation* of systems with noisy measurements revolves around construction of a state vector based on *probability density function* (PDF) using all available information (Ching, et al., 2006; Fiechter, et al., 2013; Mansouri, et al., 2013; Parslow, et al., 2013; Routtenberg & Tabrikian, 2010; Ungarala et al., 2006; Zhao, et al., 2013).

The formed probability density function contains almost all statistical information about the state. This is one of the two main advantages of using Bayesian based techniques for state estimation; the other is that these methods produce the probability density function required (Ching, et al., 2006). Once the probability density function is received, optimality criterion is then used to calculate state estimations using this complete set of information.

According to Ungarala et al., (2006), using the conditional probability function is the most complete statistical method from which practical optimization can be observed. The central issue in state estimation using probability density functions is the method used to construct the probability density function. The idea is to use the *a priori information* about the state conditional probability density function to modify it into an *a posteriori* conditional probability density function using the current measurement data.

2.4.5. Review of existing literature of observer design state estimation methods

A state observer is one of many methods used to determine unknown states of a system. Bornard, Celle-Couenne and Gilles (1995:173) describe an observer as a dynamic system used to precisely produce the current value of the state with the given input–output measurement values (Robertson & Lee, 1995; Siouris, 1996:164). State variables are used to determine the algorithm to be implemented, and thus used to calculate the required input value to be applied for a required output. The requirement for such a system to be functional is that the system it estimates must be observable (Bornard, et al., 1995; Bouraoui et al., 2015; Robertson & Lee, 1995; Siouris, 1996:163).

State observers can be designed to be local or global observers; and both these design methods are important in control engineering applications (Sundarapandian, 2012; Bornard, et al., 1995). The major drawback for most observer designs is that they become unstable in presence of uncertainties, e.g., perturbations on observer parameters (Funahashi, 1979; Garimella & Yao, 2003; Zak & Walcott, 1987).

Sundarapandian (2012) argues that the design of the local observer in nonlinear systems is one of the central problems in control system's system design. Drawbacks associated with linear Gaussian approximations, such as *biasness in residuals*, *correlation of states and non-Gaussian residuals*, *correlation of residuals in the system and measurement equations*, etc., will *result in non-robust estimation* which in turn results in *non-convergence* are easily determinable in nonlinear non-Gaussian systems (Ungarala et al., 2006; Robertson & Lee, 1995).

For this reason, the linear state observer is not adequate to reconstruct states in nonlinear systems (Gauthier, Hammouri & Othman, 1992; Krener & Respondek, 1985; Misawa & Hedrick, 1989; Shim, Seo & Teel, 2003; Wang & Zhang, 2006; Zak, 1990). Many different methods have since been developed to deal with uncertainties in the observer design structure. S*uboptimal methods* are suggested for cases where nonlinearity and non-Gaussian property are very intense such that the Taylor approximation cannot be used (Misawa & Hedrick, 1989).

The above mentioned drawbacks also necessitate the need for estimating unknown states before application of the designed optimal control strategy of the process. In designing the observer, generally, two conditions must be met, 1) *minimization of the error difference* between the model states and observer estimated state, and 2) *keeping error dynamics and its rate at zero or close to zero* under changing input or state variables (Beikzadeh & Taghirad, 2012; Gauthier, et al., 1992; Marquez & Riaz, 2005; Zak & Walcott, 1989; Zak & Walcott, 1987).

The initial nonlinear observer design was introduced by Thau in 1973. This work has grown to include or establish three most important characteristics that would determine success of the observer design for nonlinear systems; viz. are: 1) *asymptotical* stability, 2) *robustness*, and 3) *exponential convergence*. The success of the observer is based on the reduction of the dynamic error system of the observer within finite time.

The state observer design has always been classified into two but lately into three main categories; 1) *full-order state observer*, 2) *reduced-order state observer* and 3) the recently introduced *partial reduced-order state observer*. The reduced-order observer estimates only the unmeasurable states of the system under consideration; on the other hand, the full-order observer estimates all state variables of the system including states that are determinable from the system outputs (Besancon & Munteanu, 2015; Biagiola & Figuerroa, 2002; Bernstein &

Haddard, 1989; Boutat, Zheng & Hammouri, 2010; Emel'yanov & Korovin, 2011; Khan, et al., 2016; Lee & Park, 2003; Mahmoud, 1982; Zhang, et al., 2014; Zhang & Xu, 2015). The partial reduced-order considers only a certain number of the states based on the practicality of estimating the specified states (Balemi, 2008; Trinh, Fernando & Nahavandi, 2006; Zhou & Men, 2011).

Observer design based on a model description

Observer design based on deterministic or stochastic nature of the process model, continuous-time or discrete–time, time–invariant or time-varying systems has been a focus for a number of authors (Beikzadeh & Taghirad, 2012; Dani, Chung & Hutchinson, 2012; Dani, et al., 2015; Doris, et al., 2008; Germani, et al.,1975; Heemels & Weiland, 2007; Juloski, Manes & Pepe, 2002; Krokavec & Filasova, 2007; Wang, et al. 2015; Weiss, 1977; Yaz, et al., 2007; Zhang & Xu, 2015; Zhou & Men, 2011).

Other authors proposed a nonlinear observer design problem for a class of state-delayed nonlinear systems with or without time-varying uncertainties (Germani, et al, 2002; Lu, et al., 2006; Zemouche et al., 2005). Bak, Michalik & Szarfan, 2003) attempted to design a nonstationary state observer using time-varying gain matrix calculated from a Kalman filter.

Linear observers and Luenberger observer design

Directly from work of Luenberger (1964) a lot of work has been generated by different authors on the basis of the Luenberger linear observer design structure (Juloski, et al., 2007; Luenberger, 1964; Luenberger, 1966; Luenberger, 1971; Williamson, 1977; Jouili, et al., 2011; Krokavec & Filasova, 2007; Pagilla & Zhu, 2004; Pertew, et al., 2005; Robenack & Lynch, 2006; Sawant & Ginoya, 2010; Wang & Zhang, 2006). Pertew, et al., (2005) showed that the classical Luenberger-like observers are special cases of a generalized framework for dynamic observers.

Work on linear observers of systems with unknown inputs and the highlight on shortfall of this design has been presented in the literature by many authors (Bara, et al., 1999; Basile & Marro, 1969; Darouach, et al,1994; Engel & Kreisselmeier, 2002; Hou & Muller, 1992; Hou & Muller, 1994; Kazantzis & Kravaris, 1998; Mahmoud, 1982; Paraskevopoulos, Koumboulis, Tzierakis & Panagiotakis, 1991; Luenberger, 1971; Zasadzinski, Magarotto & Darouach, 2000). Some authors considered canonical observers (Busawon & Saif, 1999;

Busawon, Farza & Hammouri, 1998; Emel'yanov & Korovin, 2011; Fortmann & Williamson, 1972; Luenberger, 1966; Xia & Gao, 1988).
Some of these designs included piece-wise linear systems, systems with time delay and or with uncertainties (Doris, et al., 2008; Engel & Kreisselmeier, 2002; Hou, Zitek & Patton, 2002; Ugrinoskii, 2003).

Design of observers for bilinear systems

Bilinear systems can be considered a special class of nonlinear systems, and also to be good approximations to nonlinear systems (Bai & Liu, 2006; Busawon & Saif, 1999; Tibken & Hofer, 1989; Saif, 1993; Rafaralahy, et al., 1996; Joshi, et al., 2005; Zasadzinski, et al., 1998). Many different forms of solution for observer design of bilinear systems have been proposed by a number of authors (Behal, Jain & Joshi, 2006; Joshi, et al., 2005; Gauthier, et al., 1992; Goncharov, 2012; Grasselli & Isidori, 1981; Hammami & Jarbi, 2001; Hara & Furuta, 1976; Hsu & Desai, 1984; Rafaralahy, et al, 1996; Saif, 1993; Souley Ali, et al., 2005; Taniguchi, Eciolaza & Sugeno, 2014; Tibken, & Hofer, 1989; Wang & Burnhan 2001; Vries, Keesman & Zwart, 2007; Wang & Zhang, 2006; Williamson, 1977; Ying, Rao & Sun, 1990; Zasadzinski, et al., 1998; Zasadzinski, et al., 2000). Zasadzinski, Boutayeb and Darouach (1996) presented an observer for a singular, time-invariant descriptor bilinear system.
Krokavec and Filasova (2007) presented a state observer design procedure for the *regional pole assignment* for discrete-time uncertain system. The approach is based on controlled system model, where the model input is the difference between the actual system output and the estimated output of the system. The pole placement technique is also followed in this thesis but the method applied here differs significantly in the approach from that of Krokavec and Filasova (2007).

Nonlinear observer formulations

According to Krener and Respondek (1985), Beikzadeh and Taghirad (2012), Busawon and Saif (1999), Raghavan and Hedrick (1994), and Yaz, et al., (2007) research in nonlinear state observer design resulted in the following techniques: *extended linearization*, *feedback linearization*, *variable structure*, *high-gain observer design*, *extended Kalman filter* family, *stabilization of the original unstable system using feedback control* etc. (Alanis, et al., 2014; Atroune, 1988; Azemi & Yaz, 1999; Boker & Khalil, 2013; Braiek & Rotella 1993; Beikzadeh & Taghirad, 2012; Chen & Chen, 2007; Fortmann & Williamson, 1972; Gauthier, et al., 1992; Hou & Muller, 1994; Goncharov, 2012; Grasselli & Isidori, 1981; Kou, Elliott & Tarn, 1975; Ma & Xie, 2015; Menini & Tornambe, 2014; Misawa & Hedrick, 1989; Nagy-Kiss, et al., 2014; Paraskevopoulos, et al., 1991; Rafaralay, et.al, 1996; Raghavan & Hedrik, 1994; Saif, 1993; Sundarapandian, 2005; Tibken & Hofer,1989; Wang, Rajamani & Bevly, 2014; Yaz, et al., 2007; Walcott & Zak, 1987; Wang,et al, 2006; Williamson, 1977).

Lyapunov-based techniques and *state-dependent Riccati equation* based techniques have also been demonstrated. Another method for observer design is known as the *state-dependent coefficient* (SDC) forms, used mostly for stochastic systems. This method is sometimes referred to as *state-dependent coefficient parameterization* since it parameterizes a system's nonlinearity in an *extended linear form* (Beikzadeh & Taghirad, 2012; Yaz, et al., 2007; Walcott, & Zak, 1988).

Other approaches are based on the nature or characteristics of the systems, and this has led to further system properties being used for the observer design, properties such as; 1) *linear matrix inequality* (LMI), 2) *asymptotic design*, 3) *set-membership* and *set-value techniques*, 4) *robust and resilient designs*, 5) *dynamic nature of the system*, 6) *Lipschitz property*, 7) *error dynamic stability* and 8) *Lyapunov stability* etc.

Software simulations, software observer design solutions and some specific applications have also been considered by some authors (Behal, et al, 2006; Gauthier, et al., 1992; Raghavan & Hedrick, 1994; Song & Hedrick, 2011; Vries, et al., 2007; Yaz, et al, 2007; Ying, Rao & Sun, 1990; Wang & Zhang, 2006).

Nonlinear observer design

Many possible solutions for observer design for nonlinear systems have been developed and are continuously in a process of development to solve state estimation problem. Some authors have proposed using coordinate transformation to modify a nonlinear system into a linear one (Aguilar, Martinez-Guerra & Maya-Yescas 2003; Barbot, Boutat & Floquet, 2009; Besancon & Munteanu, 2015; Besancon & Hammouri, 1996; Boutayeb & Darouach, 2000; Braiek & Rotella, 1993; Busawon, et al., 1998; Chen & Kano, 2002; Delgado, Hou & Kambhampati, 2005; Jamel, et al., 2013; Juloski, et al., 2007; Kazantzis, Huynh & Wright 2005; Komatsu & Takata, 2009; Komatsu & Takata, 2010; Krener & Isidori, 1983; Phelps, 1989; Robenack & Lynch, 2006; Zak & Walcott,1975; Zeitz, 1987; Zemouche, et al., 2005).

In single-input single-output (SISO) systems, the solution is to find a coordinate change transformation that would modify the nonlinear system into a linear one plus at least a nonlinear term which is a function of the input and output of the system (Delgado, et al., 2005). Other solutions involved multi-input multi-output systems (Barbot, Boutat & Floquet, 2009; Deza, et al., 1993; Farza, M'Saad & Rossignol, 2004; Fortmann & Williamson, 1972; Guay, 2002; Shim & Seo, 2003).
Nonlinear control affine systems with a *maximum relative degree* and a class of nonlinear differential equations can be transformed into what is known as the *normal form*. Kravaris, et al., (2007) designed a nonlinear observer for state estimation that included uncertainty (disturbance) estimation (Kravaris, et al., 2004). Some authors considered nonlinear systems with global *Lipschitz property* (Abbaszadeh & Marquez, 2006; Chen & Chen, 2007; Gauthier, et al., 1992; Lu & Ho, 2004; Lu et al., 2006; Kou, et al., 1975; Kreisselmeier & Engel, 2003; Pagilla & Zhu, 2004; Pertew et al., 2005; Raghavan & Hedrick, 1994; Xu & Xu, 2004; Zhu & Han, 2002; Wang, et al., 2014).
Another observer design method which has become focus for most authors is the *moving horizon* observer. The moving horizon observer design was developed to handle nonlinearities of systems. This type of the observer is motivated by full order state observer but suffers the increased dimensions (Chu et al., 2007; Robertson & Lee, 1995). Some authors designed observers based on the Takagi-Sugeno model structure of neural networks (Delgado et al., 2005; Jamel, et al., 2013). The Takagi-Sugeno structure type model is mostly used in multiple model approach.

System properties used with observers

The following properties have been used by many authors to produce observer state estimators: *set-membership/set-value state estimation*, *linear matrix inequality* (LMI), *error dynamics stability* & *Lyapunov stability*. Some authors propose using set-value state estimation as means of dealing with nonlinearities in the system model, (James & Peterson, 1998; Lin, et al., 2003; Walcott & Zak, 1987; Walcott, & Zak, 1988). The linear matrix inequality (LMI) property is used to guarantee the observer convergence (Abbaszadeh & Marquez, 2006; Dani, et al., 2012; Essabre, Soulami & Elyaagoubi, 2013; Hassan, 2014; Hassan & Zribi, 2014; Jamel, et al., 2010; Jamel, et al., 2013; Jeong, Yaz & Yaz, 2011; Juloski, et al., 2007; Lien, 2004; Lu, et al., 2006; Tyukin, et al., 2013; Yaz, Jeong & Alotaibi, 2004; Zemouche, Boutayeb & Bara, 2005; Yang & Dubljevic, 2014; Wang, Shen, Zhang & Wang, 2012; Zhou & Men 2011).

Silverman and Meadows (1967) provide thorough presentation of different types of controllability and observability properties as characterized by known system coefficients (Conticelli & Bicchi, 2000; Salau, et al., 2014).

The Lyapunov condition of stabilization has been intensively applied in solving observer design problems (Biagiola & Figuerroa, 2002; Essabre et al., 2013; Jamel et al., 2013; Jouili et al., 2011; Juloski et al., 2007; Kazantzis & Kravaris, 1998; Krokavec & Filasova, 2007; Robenack & Lynch, 2006; Sundarapandian, 2002; Sundarapandian, 2006; Sundarapandian, 20011a, 2011b, 2011c; Tsinias, 1989; Zemoche et al., 2005; Zhou & Men, 2011; Zhu & Pagilla, 2006). The *error dynamics stability* procedure has been considered by some authors as means of observer design solution (Bernstein & Haddad, 1989; Lee & Park, 2003; Yuksel & Bongiorno Jr., 1971).

Based on complexities associated with nonlinearities in different system models, different types of observers have since been developed; 1) *dynamic observers*, 2) *asymptotic observers*, 3) *adaptive observers*, 4) *robust/resilient observers*, 5) *switched observer design*, 6) *partial-states nonlinear observers*, 7) *high-gain observers*, 8) *exponential observers*, 9) *non-stationary state observers*.

Discussion

Many different observer design techniques have been proposed, including: 1) feedback linearization, 2) variable structures, 3) extended linearization, 4) high gain observers and Lyapunov based techniques. Since the original work by Luenberger, the state observer has found many useful applications including system monitoring and control, detection as well as fault and failures detection in dynamical systems; but the Luenberger observer design is limited to systems with accurate models. This observer tends to ignore internal and external uncertainties. Since almost all observer designs are based on the mathematical model of the plant, the presence of disturbances, dynamic uncertainties, and nonlinearities pose great challenges in practical applications (Wang & Gao, 2003).

Multiple model approach in solving nonlinear systems modelling and estimation problems is becoming a new trend. From the very beginning, the use of state observers has shown not to be useful only for system monitoring and control, but also for detection and identification of failures in dynamic systems. Generally, one of the main challenges in designing an observer is the fact that it is heavily dependent on the accuracy of the process model, the state, the input and the output matrices.

2.4.6. Review of the existing literature of Kalman filter and other filtering methods

There are basically three main Kalman filter techniques in the Kalman family plus one, viz are: the (traditional) Kaman filter (KF), the extended Kalman filter (EKF) and the unscented Kalman filter (UKF), and the latest established ensemble Kalman (EnKF) filter. The EnKF is more of a particle filter method of state estimation rather than a real Kalman filter in the traditional sense. Generally the Kalman filter is used to solve state estimation problem for discrete-time systems. The extended Kalman filter was introduced to deal with practical implementation of the Kalman filter in nonlinear systems since the Kalman filter is only applicable to linear systems. Robertson and Lee (1995:296) describe the Kalman and the extended Kalman filters as stochatic estimation techniques. Rawlings and Lima (2008) presented lectures on state estimation in linear and nonlinear dynamic with a full discussion on EFK and UFK.

Recently, an increased interest in developing robust and optimal control strategies for complex practical system has led to a development of the unscented Kalman filter which deals with complexities in nonlinear processes. This discussion is thus going to be broadly divided according to these three estimators, since each one is simply an improvement of the previous one (Bohner & Wintz, 2013; Huber, 2013; Kao, et al., 1992; Li & Peng, 2014; Moussakhani, et al., 2014; Pritsker, 2015; Xiong & Zheng, 2015; Zghal, Mevel & Del Moral, 2014).

The original Kalman filter shall be referred to as the *classical*, or *traditional*, or the *standard* Kalman filter or simply the Kalman filter. This form of estimation has evolved to include many different variations. A few such methods include two-stage Kalman filter, the robust Kalman filter working in conjunction with the adaptive Kalman filter to produce the *robust-adaptive* Kalman filter (Huang, et al., 2012; Luo, Hoteit & Moroz, 2012; Tang & Li, 2015).

When using Kalman filter for state estimation of nonlinear systems, *propagating the state error covariance* as required by the filter procedure tends to be very difficult; and therefore, approximation methods are used in estimating the systems' states (Gillijns, et al., 2006; Robertson & Lee, 1995). Julier and Uhlmann (2004) give full review of all the three filters, and Robertson & Lee, (1995) give the main drawbacks of the Kalman filter.

Most available nonlinear system's state estimators are simply the extension of the most celebrated and widely used extended Kalman filter, and depending on

the application there would be tradeoffs in terms of *accuracy*, *implementation requirements*, *robustness* and *computational complexity* (Norgaard et al., 2000; Robertson & Lee, 1995). When using an extended Kalman filter based technique, *the algorithm requires that a Ricatti equation be solved to obtain the filter* (*estimator*) *gain*. There are two main drawbacks associated with such solutions; the first one is the requirement that *the noise model be known* to obtain optimal estimates; and worse even, *wrong associated noise model will lead to biased estimates* or *even divergence*. The second drawback is t*he relationship that exists between the filter covariance matrix and the convergence speed*; the initial values of the covariance matrix must be guessed at the beginning of the filter's algorithm (Biagiola & Figuerroa, 2002; Robertson & Lee, 1995).
Historical aspects and analysis of the least-squares methods and then development till the application of the Kalman filter have been fully analysed by Sorenson (1970). Other historical reviews have also been done by other authors (Anderson & Moore, 1991; Humpherys, Redd & West, 2012; Kalman, 1960; Welch & Bishop, 2006).

2.4.6.1. The (traditional) Kalman filter

The original work of Kalman (Kalman, 1960) was an improvement on the Wiener filer where extensions by other authors were confronted by different limitations. Kalman then introduced the new design with focus on: 1) *optimal estimates* and *orthogonal projections*, 2) *models of random processes*, 3) *solution of Wiener problem* and 4) *applications of the method*. The aim was to solve and in some cases avoid all limitations that confronted authors who worked with the Wiener's filter (Awathi & Raj, 2011; Dean, 1986; Kalman, 1960; Kalman & Bucy, 1961; Mendel, 1970; Rubinstein, 1978; Sorenson, 1970; Welch & Bishop, 2006). The original Kalman filter is a state estimator for only linear models.
Comparative studies of other filters with Kalman filter have been covered by a number of authors (Germani, Manes & Palumbo, 2002; Wang & Ho, 2003; Nakamori & Hitaji, 1982). Other filters include a *cell filter*, *robust filtering* for a class of uncertain nonlinear systems, (James & Petersen 1998; Ungarala et al., 2006). A lot of authors aimed at improving the traditional Kalman filter by working on the robust version of the Kalman filter (Anderson & Moore, 1991; Humpherys, et al., 2012; Faragher, 2012; Fitzgerald, 1971; Xia, et al., 1994; Xu & Mannor, 2009).

Robust Kalman filter

Many improvements on the original Kalman filter with the intention to create a robust filter have been dealt with by a lot of authors. These improvements include *a single iteration of Newton's method on a certain quadratic form, estimators with fading memory, adaptive fading Kalman filter* (AFKF) based on *m-interval polynomial approximation* etc., (Anderson & Moore, 1991; Faragher, 2012; Fitzgerald, 1971; Humpherys, et al., 2012; Humpherys, et al., 2012; Xia et al., 1994; Xiong, et al., 2012; Xu & Mannor, 2009).

Other authors proposed different improvements on the traditional Kalman filter so that it can be used in nonlinear systems, like using *linearization based techniques*, *suboptimal Kalman filter*, *robust Kalman filter*, *approximation methods*, etc., (Ching et al., 2006; Biagiola & Figuerroa, 2002; Diversi, Guidorzi & Soverini, 2005; Gabrea, 2003; Huang, et al., 2012; Kim, Jee, & Song, 2008; Paleologu, Benesty & Ciochina, 2013; Yang, Wang & Hung, 2002).

Many different applications of the traditional Kalman filter and its robust version have been presented by many authors, from smoothing, bilinear stochasic multivariable systems, a unified method of Kalman filtering and the errors-in-variables filtering, finite-horizon adaptive Kalman filter, adaptive fading Kalman filter, parameter estimation algorithm that uses orthogonal decomposition approach, linear two-stage Kalman filter, noise reduction methods to hydrological models (Ashayeri, Shafiee & Menhaj, 2013; Cao & Schwartz, 2004; Bhattacharjee & Das 2013; Gabrea, 2003; Germani et al., 2002; Hsieh, 2003; Kalman, 1960; Kim, Kim, & Huang, 2012; Liang, et al., 2004; Myrseth, Saetrom, & Omre, 2013; Pastorino, et al., 2013; Saab, 2004; Schilling, & Martens, 1986; Xia, et al.,1994).

2.4.6.2. Extended Kalman filter

The best definition of the extended Kalman filter is that given by Young (1981) who states that the basic idea of this estimation method is to "extend the state vector by adjoining the vector of unknown parameters". The extended Kalman filter is based on the *first order Taylor linear approximations* and the Taylor linearization does not produce accurate representation in many cases. Methods based on Taylor linearization may also suffer from *biasness of estimates* and even *convergence problems*. Almost all methods of solving the estimation of the state involve some form of linearization about the current estimates (Anderson & Moore, 1971; Bhatt, et al., 2013; Bolognani, Tubiana & Zigliotto, 2003; Chatzis, Chatzi & Triantafyllou, 2017; Haseltine & Rawlings, 2005; Kalman & Bucy, 1960;

Meng, Li & Veres, 2010; Moradkhani, et al., 2005; Norgaard et al., 2000; Sabet, Sarhadi, & Zarini, 2014).

Other methods of extended Kalman filter proposed by some authors include *iterative extended* Kalman, *maximum likelihood formulation*, *Stirling's interpolation formula* (Bolognani, et al., 2003; Cao, et al., 2016; Norgaard et al., 2000; Sargantaris & Karim 1994; Young, 1981).

2.4.6.3. Unscented Kalman filter

The unscented Kalman filter was introduced to solve some drawbacks experienced by the extended Kalman filter concerning nonlinearities; the extended Kalman filter suffers from: 1) *divergence of the filter if error propagation is not well approximated by the linear function*, 2) *does not guarantee unbiased estimates*, and 3) *calculated error covariance matrices may not represent true covariances*. The unscented transformation uses a nonlinear transform based on a group of sigma points to approximate the state probability distribution of the estimated system (Awasthi & Raj, 2011:69; Hu, et al., 2013:2942; Kulikov & Kulikova, 2017). Sarkka (2007) describes the unscented Kalman filter as a derivative free algorithm used to determine approximation solutions to discrete-time nonlinear optimal filtering problems. Full description of how to formulate and solve the unscented Kalman filter problems is given in most works (Hu, et al., 2013; James & Peterson, 1998; Julier & Uhlmann, 2004; Kolas, Foss & Schei, 2009; Maradkhani, et al., 2005; Romanenko, Santos & Afonso, 2004; UmaMageswari, Ignatious & Vinodha, 2012). Applications of unscented Kalman filter were covered by a few authors (Straka, et al., 2014; Maradkhani, et al., 2005; Romanenko & Castro, 2004; Romanenko, et al., 2004; Sabet, et al., 2014).

2.4.6.4. Ensemble Kalman filter

In the last few years, the Kalman filter family has grown to include: the ensemble Kalman filter (EnKF), the deterministic ensemble Kalman filter and the augmented Kalman filter; these filter methods were introduced to deal with the problem of high nonlinearities, high dimensions, nonGaussian distribution of variables and constraints associated with parameters and variables (Evensen, 2009:83; Simon & Bertino, 2012:1; Jahanbakhshi, Pishvaie & Boozarjomehry, 2015; Fujii & Kuroda, 2011:1035; Moradkhani, et al., 2005:136). Ensemble Kalman filter is defined as a *sequential Monte Carlo estimation method* (Evensen, 2009:83). Gillijns, et al., (2006) give the full description and background of the ensemble Kalman filter. Due to failure of the extended Kalman filter to account for nonlinear dynamics in propagating the error covariance, which in turn means failure to represent the probability density of the minimization error, the ensemble Kalman filter was introduced.

The ensemble Kalman filter falls under what is known as particle filters. Particle filters are used to avoid Jacobian matrix calculations, and the state-dependent Riccati equation (SDRE) approach. According to Gillijns, et al., (2006), the starting point for particle filters is choosing a set of sample points that is an ensemble of state estimates that capture the initial probability distribution of the state. Moradkhani, et al., (2005:137,138), Evensen, (2009) give a full description of the ensemble Kalman filter.

Discussion

Until the end of the twentieth century and earlier parts of the first decade of the twenty first century, the extended Kalman filter was the dominating technique for estimation in nonlinear models (Ching et al., 2006; Norgaard et al., 2000, Dani, et al., 2012). Ching et al., (2006) also suggested that the Kalman filter and its extensions as commonly applied to state estimation are simple *Bayesian type methods* (Ching et al., 2006; Rafieeinasab, et al., 2014).

Under nonlinear state estimation, the broad methods of solution involve: *moving horizon* estimation, *extended Kalman filter*, *unscented Kalman filter*, *ensemble Kalman filter* and *particle filter* (Kolas, et al., 2009). Julier and Uhlmann (2004) state that though there is some relation between the unscented transformation and particle filters, there are two clear distinctions between these two: 1) *sigma points* are obtained from statistics of the transformation in a deterministic manner and 2) *the transformation* itself can be interpreted more generally instead of it being considered a probability distribution.

The unscented filtering method has since been used for mostly, high-order, nonlinear couple systems, and systems with hard nonlinearities. With the growing interest in high-order, hard nonlinearities and more complex systems, the ensemble Kalman filter seem to be taking off as the best technique.

2.4.7. Review of the existing literature of combined state and parameters estimation

Work of simultaneous estimation of parameters and states of a system that has been produced covers the whole range of control systems’ descriptions: single variable systems (i.e. SISO), multivariable systems (e.g. MIMO), discrete-time, continuous-time, time-varying and time-invariant systems. In some cases it is possible to estimate both unknown states and unknown parameters by first assuming the one set of unknowns and then using these initial values to calculate the other set of unknowns (Bezzaoucha, et al., 2013; Chong, et al., 2014; Fang, de Callafon & Cortes, 2013; Ibrir, 2015; Specker, Buchholz & Dietmayer, 2014; Zhuang, Pan & Ding, 2012). Siouris (1996:190) states that the problem of combined state and parameter estimation was initially proposed as nonlinear state estimation problem formulated as an extended Kalman filter problem.

Generally, the combined nonlinear parameter and state estimation problem was solved by augmenting the state vector with the parameter vector; and the Kalman filter is then used to estimate the states (Nelson & Stear, 1976; Vafamand, & Safarinejadian, 2013). Eykoff (1968) suggests that statistical estimation techniques can be broadly divided into three categories: 1) Maximum likelihood, 2) Markov, and 3) Least squares estimation methods. Other known methods include sequential Monte-Carlo (SMC), or particle filter algorithms. These are considered online simultaneous estimation techniques, since they are based on online optimization theory (Tulsyan, et al., 2013).

Bayesian methods

Bayesian estimation formulation can be considered a general solution for all types of systems. This technique is known to produce more accurate estimates in comparison to approximate techniques such as, 1) the extended Kalman filter, 2) the moving-horizon estimator, etc. The Bayesian based estimation approach provides more information on uncertainties since the procedure estimates the whole data distribution (Chen, et al., 2004).

According to Ching, Beck, Porter, & Shaikhutdinov, (2006) the Bayesian-based state estimation methods have the following asvantages: 1) the methods are strictly based on the probability functions, and thus preserving information about the process, 2) the availability of probability density function further allows description of uncertainties within the system (Patwardhan et al., 2012). Ching et al., (2006) suggest that the first known Bayesian estimation algorithm was the Kalman filter formulated for linear systems with Gaussian uncertainties.

The main aim of the Bayesian estimation procedures is to construction estimates of the system's state sequence using stochastic difference equation process model. In the Bayesian framework the estimation is solved under certain additional assumptions, 1) the state follows a first order Markov process, and 2) measurement errors are indipendent of the given states. The objective of the

Bayesian estimation problem is to find conditional probability density function. According to Patwardhan, et al., (2012), different methods developed for estimation can be divided into two main categories:

1) methods that obtain approximation of the conditional density function and then use optimization criterion to estimate the state, and
2) methods that assume a suitable form for the priori probability density function and then converts estimation problem into an optimization problem.

Many authors have proposed Bayesian methods for solving parameter and state estimation problem jointly. In some cases, the interest is to solve the parameter estimation problem and that requires that the exact state of the system to be known, and in cases where it is not known, one is obliged to solve both the state and the parameter estimation procedure; and for the combined state and parameter estimation Bayesian based approaches tend to produce accurate estimates (Chen, et al., 2004; Ching, et al., 2006; Kabouris & Georgakakos, 1996; Patwardhan, et al., 2012; Routtenberg & Tabrikian, 2010; Zia, et al., 2008).

Maximum likelihood methods

The maximum likelihood method for estimating unknown state and parameters jointly is a natural procedure since when determining unknown parameters; the method requires that the state be known. Generally, the state estimaton problem for a class of nonlinear systems which assumes non-Gaussian process noise with uncertainties in measurements model can be considered a maximum likelihood estimation problem. This problem is solved by optimal state estimation that considers the system uncertainties. The problem can be considered state tracking within a parameter estimation procure (Zia, et al., 2008).

A maximum likelihood based method for simultaneous estimation of the state and parameters of a system, that uses a combination of *expected maximization* (EM), *nonlinear filtering* and *smoothing* algorithms was presented by Chitralekha, et al, (2010). Bar-Shalom (1972) presented an optimal estimation method based on maximum likelihood criterion. The following authors also produced combined paramerer and state estimation using maximum likelihood technique, Kabouris & Georgakakos (1996) and Zai et al., (2008).

Least squares and associated methods

Ding (2014) presents a combined parameter estimation procedure that estimates parameters using a recursive least squares algorithm; and the states of the system are then estimated using the Kalman filter based on these estimated parameters (Suzdaleva & Nagy, 2012). Dai, Sinha & Puthenpura (1989) apply the method of *stochastic decoupling* for the states to be estimated. The solution is a bootstrap method based on *Haber's principle* is developed using the extended least squares method. The model considered is *stochastic bilinear model with additive noise* having probability density function (Wang & Zhang, 2015; Zheng & Boutat, 2014).

The cost function to be approximated contains a Hessian matrix that needs to be approximated. This is achievable by solving the *model error gradient*. The novelty of the method is the fact that in using extended Kalman filter to estimate parameters, the filter simultaneously estimates the states, which appear as linear dynamic subsystem of the Wiener model. This further allows linear control techniques to be applied using *state feedback*. The gradient is approximated by numerical differentiation but can also be determined analytically. The model fitting error is greatly improved by using the Wiener model (Basin, Shi, & Calderon-Alvarez, 2011). El-Sherief & Sinha (1979) proposed a solution that eliminates the divergence and heavy computation requirement associated with original augmented method of solution (Bian, et al., 2011; Bisht & Singh, 2014; Keesman, 2015).

Joint state and parameter estimation in nonlinear & bilinear systems

Chitralekha, et al., (2010) produced a method of simultaneous state and parameter estimation for nonlinear state-space models. A simultaneous online state and parameter estimation for discrete-time stochastic nonlinear systems has been presented (Straka, Dunik, & Simandl, 2014; Tulsyan, et al., 2013; Vafamand & Safarinejadian, 2013). Halme & Selkano (1981) presented a nonlinear filter based solution for simultaneous state and parameter estimation problem. The authors developed algorithms for nonlinearities approximated by a polynomial such as the orthogonal expansion. In Bar-Shalom (1972) solution, linear discrete-time dynamic systems with noise and measurement noise are considered. Other combined state and parameter estimation methods include work of Bezzaoucha, et al., 2013; El-Sherief, 1984 and others.

Model errors and uncertainties

Hu, et al., (2010) consider simultaneous state and parameter estimation as means of dealing with model errors and uncertainties using state augmentation approach. Tyukin, et al., (2008) proposed a method for state and parameter reconstruction for uncertain dynamic systems that cannot be transformed into canonical form. Cox (1964) designed an estimator for discrete-time systems with random disturbances and measurement noise. Linear solution based on the dynamic programming approach and nonlinear solutions based on approximation technique are presented. The approximation approach applied produces real-time estimates. Ahmed (1994) presented an innovation model based solution by deriving a recursive prediction error (RPE) algorithm. Mangold et al., (1994) proposed a method that corrects model errors. Their solution is based on a simplified linear distributed parameter model. More methods that deal with nonlinearities in combined state and parameter estimation are found in Bezzaoucha, et al., 2013; Bisht & Singh, 2014; Yang, et al., 2007).

Observer based combined parameter and state estimation

Ding (2014) estimated both the parameters and states of dynamic systems jointly using an observer canonical state space system. Tyukin, et al., (2008) suggest that most observer based methods for simultaneous state and parameter estimation consider the fact that the system can be transformed into the *canonical adaptive observer* form. El-Sherief and Sinha, (1979) suggested that using a canonical form state equations simplifies the parameter estimation problem by directly relating the measurement equation to the parameters of the canonical state model. Garimella and Yao (2003); Zhu and Pagilla (2006) presented a novel nonlinear adaptive robust observer (ARO) and a stable adaptive observer respectively, that estimates both the unmeasurable states and unknown constant parameters for a class of systems described as being parametric semi-strict in feedback.

Dai et al. (1989) propose a robust bootstrap method for joint estimation of the states and parameters based on *Huber's minimax principle*. They suggest solving the nonlinear model estimation problem by introducing linear transformations based on Cholesky factor of the covariance matrix. Hulhoven et al., (2008) proposed a very attractive method of combined state and parameter estimation that combines two types of observers, the *receding-horizon* and *asymptotic* observers.

Other presentations include the work of (Ahmed–Ali, et al., 2017; Farza, 2015; Khan, Wagg, & Sims, 2016; London Jr., Mili & Bretas, 2004; Simon, et al., 2015; Tulsyana, et al., 2013; Zheng & Boutat, 2014; Zhu, 2012).

Kalman filter based combined state and parameter estimation

Hu, Zhang and Nielsen-Gammon (2010); Khodadadi and Jazayeri-Rad (2011) presented an extended Kalman filter based methods that generate state and parameter *maximum-likelihood* estimates for a discrete-time nonlinear dynamic system. Bocquet and Sakov (2013); Maradkhani, et al., (2005) propose a dual state-parameter estimation approach based on the Ensemble Kalman filter (EnKF) for hydrologic models. Havlena (1993) presents an adaptive Kalman filter method for simultaneous parameter tracking and state estimation in a linear time varying input-output ARMAX model with Gaussian noise and known noise parameters. More work on filter based filtering and parameter estimation is found in (Basin, et al., 2011; Ding, F. 2014; Gadsden, et al., 2014; Gharamti, Ait-El-Fquih, & Hoteit, 2015; Runggaldier & Visentien, 1990; Vafamand & Safarinejadian, 2013).

Measurement based combined parameter and state estimation

Singh and Hahn (2005) considered optimal location of sensor for measurements that will in turn produce highest degree of observability of the system. This is to produce optimal parameter and state values. Roberts and Williams (1981) presented an algorithm based on the design of a control scheme that can be separated into: 1) *parameter estimation* and 2) *optimization problem*. The two problems are interconnected in that the optimization problem depends on the parameters, and parameter values are affected by the controller values. An improved version of this optimization and parameter estimation that provide online optimizing control problem has been proposed by Kambhampati et al., (1989). They provide a generalized solution to methods commonly referred to as ISOPE (Integrated System Optimization and Parameter Estimation) (Ding & Chen, 2004; Jaoua, et al., 2014; Kreuzinger, Bitzer & Marquardt, 2006; Kupper, et al., 2009).

Applications

A lot of applications been covered in joint parameter and state estimation for geosciences including mesoscale meteorology, data assimilation methodologies, conceptual rainfall run-offs, hydrologic models, etc. (Moireau, et al., 2008; Maradkhani, et al., 2005; Schilling & Martens, 1986). Moireau, et al., (2008) suggest that data assimilation methods can be broadly divided into two, viz.: 1) *variational* and 2) *sequential* (*filtering*) algorithms. Variational methods minimize a criterion based on observation error, with respect to system unknown parameters. Sequential algorithms may not be conducive for distributed mechanical systems, since they require conversion steps and covariance matrices of the size of the state in their method of solution (Bezzaoucha, 2013; Mansouri, et al., 2014; Sulaiman, et al., 2013).

In the bioprocess sciences, applications include work of Hulhoven et al., (2008) who applied their design to solve kinetic parameters and states simultaneously in a bioprocess model. Moireau, et al., (2008) proposed a joint parameter and state estimation technique based on state estimation feedback system (that utilizes stabilization strategies) for a biomechanical system. Mangold et al., (1994) considered the adsorption column process with limited measuring points and no information about the state of the system. The model numerical calculations are solved using software program called DIVA.

Discussion

Combined state and parameter estimation methods take the same order like parameter estimation or state estimation as individual problems. The procedures of solution can be considered under the main categories of: 1) least squares, 2) maximum likelihood or 3) Bayesian.

Estimation problem formulations and correspondint solutions for different nonlineat models presented by different authors has been given. For combined state and parameter estimation problem the maximum likelihood based algorithms seem provide a natural solution to such problems becasue for parameter estimation problem, the state has to be known. Possible solutions include (Zia, et al., 2008):

1) the traditional method of treating the unknown parameters as extra state variables in the state vector,
2) the problem can be separated into two joint problems that of parameter estimation and state estimation, or
3) the problem that needs adaptve fitering method to decide on the most appropprite form of the model to use.

2.5. Conclusion

There are many reasons associated with building a model; models built for simulation of the dynamic behaviour of processes, for prediction and or analysis of the process behaviour, prediction of time evolution behaviour of the process. In physical systems most models are nonlinear or bilinear and contain unknown parameters and unknown states that may be due to unavailable sensors or unmeasurable parts of these processes. This chapter presented a number of different solutions and different types methodologies to solve the estimation problem.

The focus of this thesis is on parameter estimation and two state estimation methods: 1) *observer design* methods and 2) the *Kalman filter* and its extended family, and other methods associated with all these techniques for nonlinear systems. Mostly, for nonlinear system estimation specifically, the extended Kalman filter has been the intensely used method but it has some limitations in highly nonlinear systems and newer methods have since been developed to attend to this inadequacy.

Parameter estimation

The chapter considered a number of different possible solutions that have been introduced to deal with unknown parameters. The parameter estimation solutions have been presented according to the method of solution, whether based on model structure, model properties, etc. Algorithms based on numerical meethods, optimization techniques, heuristic methods, observer based methods, online procedures, methods based on other estimation techniques and software programs used to implement some of the algorithms have been discussed; and lastly, applications of these methods in real processes were also presented.

State estimation

The state estimation problem has been presented based on two commonly used methods, 1) the observer design and the Kalman filtering methods. The extended forms of Kalman filter have also been presented. Different observer design methods were presented. The original Kalman filter as an observer method of estimating unknown state variables of a system is strictly applicable to linear systems has some limitations when it comes to nonlinear models.

The available solutions range from deterministic to stochastic, discrete-time to continuous time, time varying and to time-invariant, linear, nonlinear and bilinear systems. This is the most well covered topic under state estimation.

Combined state & parameter estimation

A number of methods using combined state and parameter estimation procedure have been proposed for solving estimation problem for most systems, linear or nonlinear. Most methods are based on some form of Kalman filtering or its extensions, i.e., extended Kalman filter, unscented Kalman filter and Ensemble Kalman filter methods in conjunction with another for parameter estimation.

Different schemes have been presented that are based on three main categories of estimation, i.e., least squares formulations, maximum-likelihood methods and Bayesin methods. Most of papers produced presented observer based solutions. Other solutions are measurement based, error correction based, filtering based and or uncertainty based. A few applications have also been presented here.

The next chapter introduces theoretical aspects behind modeliling, parameter and state estimation. The focus is on how nonlinear parameter and state estimation problems are solved based on models with uncertainties and nonlinearities.

CHAPTER THREE

NONLINEAR SYSTEM THEORY

CHAPTER THREE

3. NONLINEAR SYSTEM THEORY

3.1. Introduction

This chapter deals with the relationship between three major sections involved in the subject matter of the project, that is, "*methods for modelling, parameter and state estimation for nonlinear processes*". The focus is on nonlinear systems in general, and on how parameter and state estimation in such systems is performed. The chapter also presents an overview of the most important aspects of nonlinear estimation techniques, the optimization theory.

The chapter intends to provide streamlined background information to solve parameter and state estimation problems using optimization techniques in nonlinear systems, without necessarily reducing these subjects to any less complexity than they are. The focus is on giving a specific path within a labyrinth of the subject matter involving modelling, estimation theory, optimization theory and nonlinear processes. The chapter considers nonlinear system theory, nonlinear modelling and nonlinear estimation and how these concepts are applied in parameter and state estimation methods. Fundamental model building techniques are also considered.

3.2. Nonlinear system theory: modelling and estimation

Physical systems tend to exhibit high nonlinear characteristics in their operations (Rad & Hancu, 2017). The interest of engineers is to find a way of controlling such processes in some optimal manner. The engineer analyses the system, determines variables and parameters of interest, and system constants based on laws that govern the behaviour of the system and then suggests means of controlling it (Roxin, 1997).

If a process has at least one nonlinear component, the process model will be nonlinear. Designing a real-time control system for a nonlinear process is more complex than that of a linear process. Nonlinear systems where the operation range of a process is relatively small, a nonlinear system may be approximated by a linear equivalent model that would capture all the required dynamics of the system. In linear control analysis, solutions are always obtained easily by either applying time domain or frequency domain solutions. However, for nonlinear systems, these standard solutions are not applicable, direct solutions for nonlinear variables are usually impossible and frequency domain transformations do not apply (Slotine & Li, 1991:1-16; Jamshidi, Tarokh & Shafai, 1992).

Seborg and Henson (1997:1) suggest that nonlinearity in control systems for physical processes is further translated into the relationship between *controlled* and *manipulated variables dependence on the process operating conditions*. In some instances it is possible to represent a nonlinear system by its linear approximate if the process remains in a steady state condition within its operating region (Grancharova & Johansen, 2009; Rad & Hancu, 2017; Seborg & Henson, 1997:1).

Seborg and Henson (1997:7) state that physical–based models, those designed from the first principles (e.g., mass and energy balances) tend to be more appealing because of the amount of physical detail they provide and the fact that they are applicable over a wide range of operating conditions. In most industrial operations such models are not easily available due to costs involved in developing them and their maintenance.

There are three main factors that heavily contribute to the model building process, these are: 1) *a priori knowledge*, 2) *experimental data* and 3) the *modelling objectives*. In most physical applications of modelling, the *a priori* knowledge contains high quality value such that the overall system framework and important aspects of the model can be deduced from it alone. It is also possible to use the methodology used in modelling such systems to solve the parameter estimation problem and also make minor adjustments to the model using experimental validation methods (Jeppsson, 1996:32–33).

In cases where prior experimental data is not available, it is possible to use identification methods to solve parameter estimation problem for unknown parameters. In identification the ultimate aim is to obtain or give each parameter of the model a unique value. Complex system structures with unidentifiable model parameters are unusable and reduced-order models should be considered in such cases. In some cases, it may be useful to consider a priori identification analysis but the ultimate quantification of parameter identification is determined largely by estimation of accuracy and the correlation between the estimated parameters (Jeppsson, 1996:32–33).

According to Biagiola and Figueroa (2002:4777) automation of physical processes with interest in improving product quality against reduced costs, increased product outputs, and other improvement possibilities has been of special interest in recent years which led to improvements in modelling techniques, control and monitoring techniques etc. Such requirements have led to state estimation becoming a strategic topic for control system (process control) engineers and researchers. The requirement of knowledge of some parts of, or the entire *system state vector* becomes an important feature in control system's operations since the state vector describes the internal (dynamic) behaviour of the system in full or gives some meaningful understanding of the underlying processes.

This system requirement of knowledge of the state vector, among many other possibilities, allows *improved nonlinear control*, *enhanced monitoring* of specific variables in the process which allows *well suited process measurements* for *data collection*, etc. If the system is well monitored and at highest possible frequency, it becomes possible to apply optimization techniques for the improved product quality and output at reduced cost. Optimization techniques are formulated to provide best physically sound and realistic results.

In cases where there are physical limitations or where it is impossible to measure all physical attributes associated with the state vector, state estimation techniques (such as a state observer) may be used. A number of linear and nonlinear state estimation techniques have since been established to determine unknown states using available process measurements (Bose, 1959; Goldberg & Durling, 1971; Biagiola & Figueroa, 2002; Seborg & Henson, 1997).

According to Biagiola and Figueroa (2002), nonlinear estimation techniques are structured to handle specific nonlinearities depending on the process model. This is by no stretch of imagination the anticlimax of research in nonlinear estimation given the well-established linear estimation techniques, system robustness, uncertainty in performance, system and measurement noise etc.; it is probably the only beginning (Biagiola & Figueroa, 2002; Thangavel, Paulen & Engell, 2016).

In this thesis, the reason for going through process modelling, parameter estimation, and state estimation, is concerned with *optimal control system design methodologies* for use in control and monitoring of nonlinear processes. The aim is to improve and or enhancing some aspect of the process control so as to match the real-world process dynamics with the control system's design methodologies. A short overview of methods in model building for the purposes of control, formulation of the problems for parameter and state estimation, and types of optimization problems necessary to be solved are described in the sections below.

3.3. Model building and development for purpose of estimation

Models are developed to describe the underlying principles that define the behaviour of a process. This helps in better understanding of the complex dynamical processes which makes design, control and operating strategies easy. Mathematical models are generally used to describe the process behaviour using variables that describe process parameters. A mathematical model can only successfully be applied if its structure and parameters give a proper description of the process behaviour (Dochain & Vanrolleghem, 2001:1–3; Fliess & Normand-Cyrot, 1910; IDA, 1997; Olsson & Newell, 1999; Luyben, 1990; Luyben, 1973).

A model can be thought of as an imperfect or incomplete representation of the reality. This may be due to a lack of knowledge about the process behaviour or may be because of deliberate simplification to satisfy some requirements. In general a model will be a compromise between some predetermined expectations and the means at getting to those expectations, either by experiments, simulations, etc. (Bornard, 1995; Barret, 1963; Junkins, 1991; Luyben, 1990; Luyben, 1973; Pierre, 1969).

A model may never capture all the physical aspects of the process; it all depends on the aim and objectives for developing that particular process model. What may be of interest to one designer may not be the case for the other designer, though both of them studying the same process. In this regard a model can be classified based on the objective as either being a model for: *conceptualisation*, *simulation* and or *control* (Bornard, 1995; Franks, 1972; Olsson & Newell, 1999).

Model building consists of the following steps, 1) problem specification, 2) determination of model structure based on some physical knowledge (a priori knowledge), 3) model verification, 4) fitting of parameters onto available experimental data, and 5) model validation. Mathematical models allow conceptualization of some knowledge about a process under scrutiny, formulating hypothesis about certain aspects of a system, and or testing and verification of new ideas about any process concerned.

A well-constructed model will be a true reflection of the behavioiur of a process even if some aspects have been left out; such a model will allow a designer to: 1) predict process behaviour under different operating conditions, 2) and in turn allow optimization of the process operations and control. Models may also serve as a training tool and a starting point for experimentation before constructing a full scale plant that may produce disastrous results if it was never simulated and tested as a model (Li & Ding, 2013; Jeppsson, 1996).

A model development is guided by the intended use and goals of the model based on some system requirements. A key feature of a model is its meaningful prediction of the process behaviour within a relative time frame. Models are developed for different purposes based on the goals of the designer. These purposes may be classified as: 1) *research*, 2) *process design*, 3) *process operations and control*, 4) *process optimization*, 5) *prediction*, and 6) *process/plant diagnosis* (*fault detection*) etc. It is almost impossible to develop a single model that could fit all of these purposes (Billings & Fakhouri, 1979; Ching, Beck & Porter, 2006; Dochain & Vanrolleghem, 2001; Gardiner, 1973; Franks, 1972; Ikonen & Najim, 2002; Johansen & Foss, 1995; Luyben, 1990; Kortmann & Unbehauen, 1988; Olsson & Newell, 1999).

Dynamic process models development built from first principles are derived from laws of conservation of mass and energy. These laws are guided by the fact that what goes into a vessel (system in enclosed space) is either stored inside this space or it comes out at some boundary of the space. This is referred to as conservation of mass and energy and the mathematical descriptions generated from these laws are known as *mass or energy balances* and thus the term *balance equations* (Olsson & Newell, 1999). In developing a model, balances for each component to be considered must be generated (Olsson & Newell, 1999; Schetzen, 1974; Stone & Womack, 1970; Thathachar & Ramaswamy, 1973). This produces a set of differential equations which shows the interrelationships between these components.

The general conservation (balance) equation can be expressed as follows:

$$\begin{aligned}\begin{bmatrix} Rate\ of\ change \\ of\ contents\ within \\ an\ enclosed\ system \end{bmatrix} &= \begin{bmatrix} Rate\ of \\ input\ flow \\ to\ the\ system \end{bmatrix} - \begin{bmatrix} Rate\ of \\ output\ flow \\ to\ the\ system \end{bmatrix} + \\ &+ \begin{bmatrix} Rate\ of\ generated \\ materials \\ inside\ the\ system \end{bmatrix} - \begin{bmatrix} Rate\ of \\ materials\ consumed \\ inside\ the\ system \end{bmatrix} \end{aligned} \tag{3.1}$$

The rate of change is generally described as the derivative with respect to time. Before deriving mass or energy balance, boundaries for the volume over which the balance is written must be defined. The boundary definition should be stated as part of the assumptions used to develop the model.

It is important to specify if the model is built from the first principle or it will be an empirical one, Figure 3.1. The process of using experimental data to generate a model is known as *identification* and should not be confused with parameter estimation. Parameter estimation is one step of building a model that is used to determining unknown parameters whether using the first principles or using experiments. Parameter estimation is generally referred to as *fitting of data* in identification and as parameter estimation if using the first principle; these terms are not fixed in this fashion, they are equally interchangeable in their usage (Eykhoff, 1974:8; Ikonen & Najim, 2002:7; Jeppsson, 1996:112; Shanshiashvilia & Prangishvili, 2017).

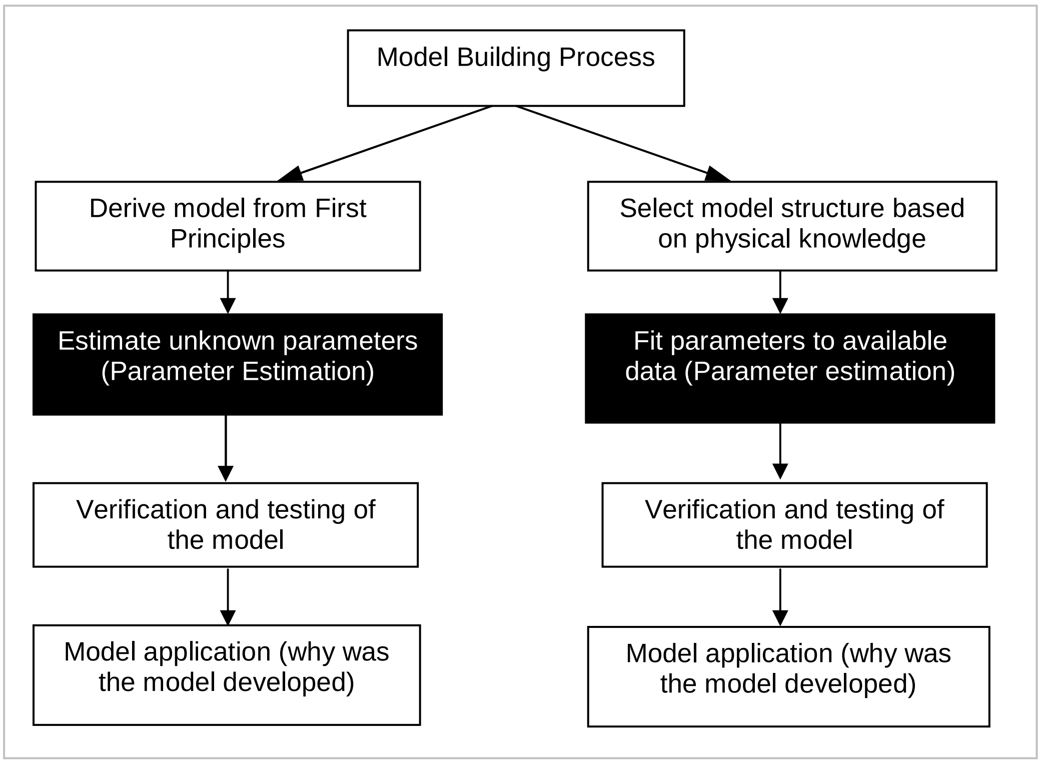

Model Building Process
Derive model from First Principles
Select model structure based on physical knowledge
Estimate unknown parameters (Parameter Estimation)
Fit parameters to available data (Parameter estimation)
Verification and testing of the model
Verification and testing of the model
Model application (why was the model developed)
Model application (why was the model developed)

Figure 3.1. Estimation of parameters and fitting of parameters in model building process

(Adapted from Eykhoff, 1974; Ikonen, Ikonen & Najim, 2002; Jeppsson, 1996)

Modelling assumptions are important in defining the process operations. In some instances assumptions may be used to simplify the process model being developed or to take care of those process aspects that cannot be defined. It is important that the assumptions made are relevant to the model and its intended use, Figure 3.1.

3.4. Dynamic process models

In general, physical processes experience some disturbance in their operations. This could result from the process environment and or measurement or approximation errors, Figure 3.2. A model should capture the essential characteristics of the disturbance as an essential component of the model (Ikonen & Najim, 2002:53).

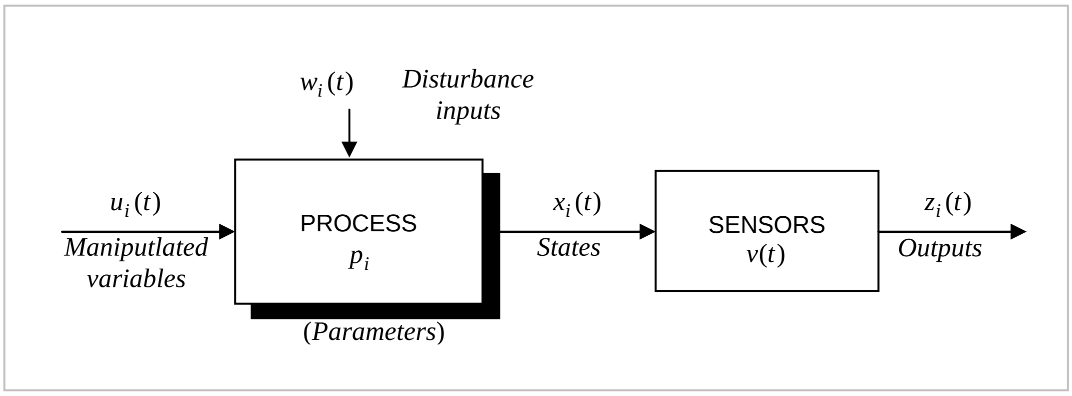


Figure 3.2. **Process model definition**

(Adapted from Olsson & Newell, 1999:41)

The measurement disturbance vector, $v(t)$ represents the error caused by the measuring instrument and the process disturbance vector, $w(t)$ represents the process unmeasured system noise that affects the process state. Disturbances can be categorized into two main classes: 1) *deterministic* and 2) *stochastic disturbances*. In some cases, deterministic disturbances are predictable. Such disturbances include constant and sinusoidal effects (Ikonen & Najim, 2002:53). In physical processes, effects of the process environment are modelled as system (process) disturbances, $w(t)$ and approximation errors resulting from generating linear equivalent models of the nonlinear physical processes are also modelled as process disturbances.

Model disturbances are further divided into two categories: 1) *measured* and 2) *unmeasured* (e.g., material compositions) disturbances. Modelling of unmeasured disturbances is based on assumption that they consist of *sequential independent random variables* $\gamma(t)$ with zero mean and variance σ and random disturbances are assumed to be stationary (Ikonen & Najim, 2002:54) and the relationship between these variables is as follows (Ikonen & Najim, 2002:54; Martin-Lof, 1966:603; Schennach, 2004):

$$b(t) = \sigma\gamma(t) + \tau \qquad (3.2)$$

where

$b(t)$ – is a random sequence,

$\gamma(t)$ – is a sequence of independent random variables,

σ – is the model variance and

τ – is a constant not dependent on t.

Equation (3.4) is based on the assumptions that:

1) the random variables have a zero mean, Equation (3.5),

$$E\{\gamma(t)\} = 0 \tag{3.3}$$

2) the variance is defined by

$$\sigma^2 = 1 \tag{3.4}$$

3) Equation (3.4) is true, only under the following assumptions,

$$E\{b(t)\} = \tau \tag{3.5}$$

$$E\{b(t)^2\} = \sigma^2 \tag{3.6}$$

Olsson and Newell (1999:101–103) suggest that parameters that are fitted to real data will have some element of uncertainty, but this does not necessarily translate to a stochastic model; this is true only in cases where parameter uncertainty is built into the model as a statistical distribution. Disturbances may also determine classification of a model, if a model is static or dynamic, based on the formulation of the solution and or design requirements for the model.

Dynamic models are defined as those that would predict the system behaviour as a function of time in response to the disturbances (Olsson & Newell, 1999:102). In process control terminology, a process is defined in terms of the *operating space* and *the state space*. The input-output relationship of a model can be used to determine the type of a model. Generally, the dynamic process models are described using the following two categories, the:

1) Input–output (dynamical) model, and
2) State space (dynamical) model.

A model can be an *input-output* or *state space* model. Input-output model is a set of transfer function(s) that relates the inputs directly to the outputs (Ding, et al., 2017; Diaz & Desrochers, 1988; Haber & Keviczky 1999; Leontaritis & Billings, 1985). The state of the system is defined as the values of the state variables at any instant of time. The state space model introduces the state variables (state vector) in a model to act as link between the inputs and outputs.

A dynamic model generated on the basis of only input and output measurements, such as the ARMAX model can be described by Equation (3.7) (Fattah, Zhu & Ahmad, 2008; Fujimoto & Takaki, 2016; Ikonen & Najim, 2002; Liao, Wang & Ding, 2009; Mukkula & Paulen, 2017; Peng, et al., 2001; Stecha, 1997):

$$\begin{aligned} z(k) &= a_1 z(k-1) + a_2 z(k-2) + \ldots \\ &\ldots + a_n z(k-n) + b_1 u(k-1) + b_2 u(k-2) + b_n u(k-n) \end{aligned} \tag{3.7}$$

where

$a_1, \ldots\ldots, a_n$ and $b_1, \ldots\ldots, b_n$ are the model coefficients,

$z(k)$ – is the discrete model output,

$z(k-n)$ – are the past outputs,

$u(k-n)$ – are past inputs and

$p=[a_1,....,a_n,b_1,...,b_N]$ – is a vector of unknown parameters.

The operating space is defined in terms of manipulated variables (process input variables). Any physical process is delineated by a set of operating constraints; and these constraints may be *process related*, *equipment related* or *safety conditions* related; this is known as *feasible operating space*. The state space is defined in terms of the state variables and it usually has a higher dimensional space then the operating space (Olsson & Newell, 1999:41).

Level of complexity of models

The level of complexity in a model has direct influence on the ultimate outcome of the problem formulation and the relevant solution, whether for control or otherwise. It also directly impacts on estimation of parameters or states as may be required. If a model is developed for control and monitoring, depending on the model structure, (including disturbances), uncertainties, the number of important parameters and the control objective, a control system used may be classified under one of the following categories: 1) *classical feedback control*, 2) *adaptive control* or 3) *optimal control*.

Complexity of models that result directly from physical laws are often simplified to models which include only part of the initial information already known and which may have to be adapted according to their behaviour (Bornard, 1995:7; Haber & Kevikczky 1974:393–414; Schei & Singstad, 1998). There needs to be a balance between *minimum information required to capture the reality* and *the complexity* accompanying that reality (Carvalho, et al., 2015; Palm III, 1986; Safdarnejad, et al., 2016; Schei & Singstad, 1998; Shanshiashvilia & Prangishvili, 2017).

3.5. Model types used in estimation

3.5.1. Linear versus nonlinear models

The mathematical expressions describing the interrelationship between the system components using vector notation are generalized as follows:

1) *for linear systems*,

$$\dot{x}(t)=Ax(t)+Bu(t)+w(t)\,,\ \ x(t_0)=x_0 \qquad (3.8)$$

$$z(t)=Cx(t)+v(t)$$

2) *for nonlinear systems*,

$$\dot{x}(t) = f(x(t), u(t), w(t), p), \quad x(t_0) = x_0 \tag{3.9}$$
$$z(t) = g(x(t), u(t), v(t), p)$$

where

$x(t) \in R^{n \times l}$ – is the state of the system of $n \times l$ dimensions,

$u(t) \in R^m$ – is the input vector of the system of $m \times 1$ dimensions,

$w(t) \in R^n$ – is the disturbance to the system with $n \times 1$ dimensions,

$v(t) \in R^l$ – is the vector of measurement disturbances with $l \times 1$ dimensions,

$z(t) \in R^l$ – is the output vector of the system of $l \times 1$ dimensions,

$p \in R^q$ – is the vector of the system coefficients of $p \times 1$ dimensions,

$g \in R^l$ – is the nonlinear output vector function with output vector dimensions, and

$A \in R^{n \times n}$, $B \in R^{n \times m}$, and $C \in R^{l \times n}$ are the state, the control and output matrices respectively with corresponding dimensions.

3.5.2. Linear-in-parameters models

A process model is called *linear-in-parameters* if the process *neglects the noise vector* and the output variable can be described by a scalar product of two vectors

$$z(k) = \phi_m{}^T p \tag{3.10}$$

where

p – is the parameter vector and

ϕ_m – is the memory vector containing measured values or values calculated from some measurements (Haber & Keviczky, 1999:3–5).

A process model with a *noise vector* is called *linear-in-parameters* if the source noise can be determined from the measurements by a scalar product of two vectors

$$w(k) = \phi_m^T p_n \tag{3.11}$$

where

$w(k)$ – is the calculated source noise called the residual, and

p_n – is the parameter vector of the process model with noise.

3.5.3. Nonlinear-in-parameters models

Haber and Keviczky (1999) define the nonlinear-in-parameters with noise and without noise as follows: A *noise-free* process model is called *nonlinear-in-parameters* if its output cannot be defined using Equation (3.11) above. A process model with a *noise vector* is said to be *nonlinear-in-parameters* if the source noise scalar product cannot be determined using Equation (3.11) above (Alexandrov, Alexandrov & Shatov, 2016; Haber & Keviczky, 1999; Kamoun, 2007).

3.5.4. Parametric and nonparametric models

A model is said to be *parametric* if it is defined by a *finite number of parameters* and *nonparametric* if a large number of *infinite parameters* are required to describe the model (Keesman & van den Brink, 2015; Haber, 1979:515–522; Vuchkov, Velev & Tsochev, 1985). Simple nonparametric models can be grouped into the following categories (Haber & Keviczky, 1999:4):

1) models linear-in-parameters,
2) block oriented models,
3) quasi-linear models with signal dependent parameters and
4) quasi-linear models with piecewise constant parameters (multi-models).

3.5.5. Deterministic and stochastic models

A mathematical model will not cover all the physical aspects of the process except those that are relevant for that particular study. The mathematical models used in estimation are therefore either *deterministic* or *stochastic*. In the deterministic representation the noise is either not acting on the system or has a negligible effect. Some deterministic approaches admit *zero mean noise* but cannot express the uncertainty caused by the noise (Strejc, 1981). In the stochastic models the assumptions are that the measured input signal is not affected by any noise and only the output is affected by noise. This means that either the process is disturbed by external noise or the measured output is affected by the noise. Stochastic discrete equation can be represented by (Strejc, 1981):

$$\sum_{i=0}^{N_a} a_i z(k-i) = \sum_{i=0}^{N_b} b_i u(k-i) + w(k) \tag{3.12}$$

where

a_i and b_i – are the system parameters,

n – is the order of the dynamic system,

$z(k-i)$ and $u(k-i)$ – are the output and input polynomials of the system,

$(k-i)$ – is the past moment and

$w(k)$ – is the noise vector representing noise acting on the system.

Stochastic nonparametric model is defined by:

$$z(k) = f[u(k-d_k),...,u(k-d_k-n_u),e(k),...,e(k-n_w)] \tag{3.13}$$

And *stochastic parametric model* is defined by

$$z(k) = f[u(k-d_k),...u(k-d_k-n_u),z(k),...,y(k-n_z),e(k),...,e(k-n_w)] \tag{3.14}$$

where

d_k – is the integer time delay relative to the sampling period,

n_u, n_z, n_w – order of the equation of the input, output and the noise respectively.

A stochastic nonparametric model can be obtained from a parametric model by omitting the output signal on the right hand side of the stochastic parametric representation, (Equation 3.14).

3.6. Parameter estimation

Parameter estimation can be thought of as a technique used to identify process model parameters by evaluating coefficients in a known mathematical model using measured data (input–output measurements) (Kamoun, 2007; Siouris, 1996:63–85). Parameters can be estimated off-line where model parameter values are identified just once and used for the rest of the control operation. Parameters can also be estimated on-line where parameter values are updated every time measurements are taken (Nicholson, López-Negrete & Biegler, 2014; Peng, et al., 2001; Schei, 2008; Schei & Singstad, 1998). This type of estimation is used for predictive control. The estimation techniques use measured data to estimate the parameter values of the model. Olsson and Newell (1999) state that, parameter estimation using experimental data is commonly referred to as *regression*.

Parameter estimation problem is an optimization problem, in that a cost function must be minimized. The cost function consists of a sum of the squared prediction errors, i.e., the sum of errors must be minimized. The prediction errors are the differences between the predicted values of the output of the process to that of the measured one. These errors are commonly referred to as *residuals* (Olsson and Newell, 1999:157; Ikonen & Najim, 2002; Corlis & Luus, 1969; Pierre, 1969).

3.6.1. Linear parameter estimation

There are many different techniques used to estimate parameters in linear systems. The three most commonly used ones are: the *least squares*, *linear regression*, *maximum likelihood*, and *Beyes methods* (Jacob, 2010; Barham & Drane, 1972; Liao, et al., 2009; Sarris & Eisner, 1973; Strejc, 1980). In the first two methods the optimal criterion is a scalar. Other techniques are either the variances of the above three methods or their extensions. Most of these techniques are developed for specific practical problems for which no solution exists or to improve on the well-accepted estimation procedures (Ding, et al., 2017).

3.6.2. Nonlinear parameter estimation

In general, nonlinear parameter estimation problem can be formulated as in Figure 3.3. One may observe any of the system's variables, the input signal $u(t)$, state vector $x(t)$, input disturbance $w(t)$ and or the output signal $z(t)$ (Tang, et al., 2016). The observation model may be represented by the following discrete model (Siouris, 1996; Soderstrom & Stoica 1989; Strejc, 1981):

$$z(k) = g(x(k), u(k), w(k), p(k), v(k), k) \tag{3.15}$$

where $p(k)$ – represents the vector of unknown parameters of the system, consisting of the mean and variance parameters of the input noise $w(k)$ and observation noise $v(k)$. It may be assumed that the state vector evolves from the stochastic nonlinear model. Such a model can be described by Equation (3.16):

$$x(k+1) = f(x(k), u(k), w(k), p(k), k) \tag{3.16}$$

3.6.3. Least squares parameter estimation

The least squares method is the most commonly used method for estimation of parameters. Its history is traceable to as far back as 1795 when Karl Friedrich Gauss formulated its basic concept and used it practically for astronomical computations. The least squares method is concerned with determining the most probable estimate value of parameter $\hat{p}$ (Barham & Drane, 1972; Billings & Voon, 1984; Liao, et al., 2009; Marquardt, 1963; Strejc, 1981). This least squares method is defined by the expression (Keating et al, 1993:1):

$$e = z - \hat{z}(\hat{p}) \tag{3.17}$$

where

z – is the observed value,

p – is the unknown parameter,

$\hat{z}(\hat{p})$ – is the output estimated value, and

e – is the additive experimental error.

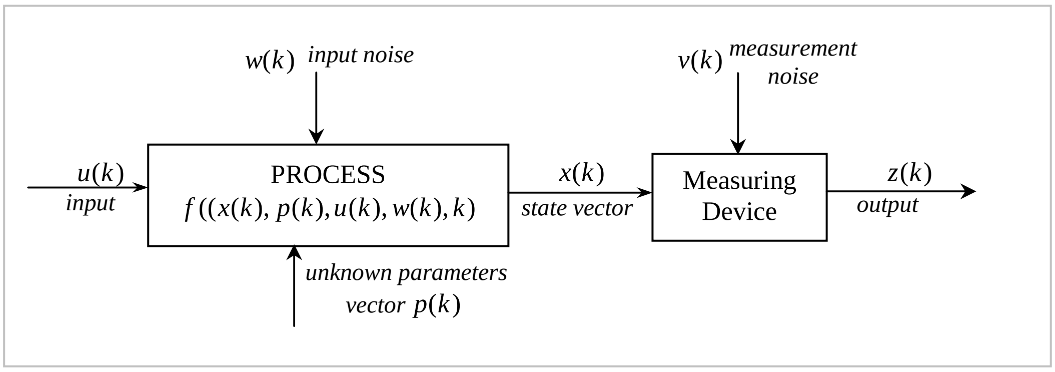


Figure 3.3. **Nonlinear process model structure**

(Adapted from Strejc, 1980)

The expression (Equation 3.17) can be extended for a number of observations taken to measure an unknown vector p.

$$e_i = z_i - \hat{z}(\hat{p}) \tag{3.18}$$

where

z_i – is the i^{th} observed value of m observations for $i = 1, 2, ..., m$ and

e_i – is the additive error at each measurement.

The estimate value is the value that minimizes the sum of squares of the estimation errors, the residuals (Strejc, 1972). The estimate value $\hat{p}$ is also known as the least squares estimator and is given by (Keating et al., 1993)

$$\hat{p} = \frac{1}{m}\left[\sum_{i=1}^{m} z_i\right] \tag{3.19}$$

The formulation of the least squares method of parameter estimation is:
To calculate p so that the cost function

$$J = (z - \hat{z}(\hat{p}))^T Q^{*-1}(z - \hat{z}(\hat{p})) = e^T Q^{*-1} e \tag{3.20}$$

is minimized.

For ordinary least squares method, Q^* is an identity matrix.

3.6.4. Maximum likelihood parameter estimation

The maximum likelihood estimation method is based on the maximization of probability functions. The aim is to determine parameters such that the likelihood function attains its maximum. For linear systems, the maximum likelihood approach yields the same conditions for the parameter calculation as the least squares (Kok, et al., 2015; Strejc, 1981:5).

The maximum likelihood estimation is based on the likelihood function L_m defined as a joint probability density function. The likelihood function can be represented by (Sarris & Eisner, 1973):

$$l(p|u_1,......,u_n) = f(u_1,.....,u_n : p) \quad (3.21)$$

where

$u_1,.....,u_n$ – is a function of the unknown parameters.

Whenever the functions $u_1,.....,u_n$ are independent and identically distributed with common density function $f(u:p)$, the model can be reduced to

$$l(p|u_1,.....,u_n) = f(u_1, p),.....,f(u_n, p) \quad (3.22)$$

Keating et al (1993) suggest that the maximum likelihood estimator (MLE) is the value of parameter that produces the absolute maximum of the likelihood function $l(p|u_1,.....,u_n)$ and is a function of $u_1,.....,u_n$. It is however, not always to represent the estimator as a function of $u_1,.....,u_n$. In some cases, it may even be difficult to establish an *absolute minimum*.

The *discrete time function* of the maximum of the likelihood can be described by:

$$L[p,R:z(K,0),u(K,0)] = f(z(K,0),u(K,0):p,R) \quad (3.23)$$

where

$$f[z(K,0),u(K,0):p,R] = \left[\prod_{k=0}^{K} f[z(k)|z(k-1,0),u(k,0):p,R]\right]\times$$
$$\times\left[\prod_{k=0}^{K} f[u(k)|z(k-1,0),u(k-1,0)]\right]$$

3.6.5. Linear regression estimation

Considering a random quantity (random variable) γ which is a function of one or more independent (deterministic) variables $\rho_1,\rho_2,....,\rho_m$. Deterministic system assumes process noise to be negligible. The aim is to estimate the relationship between γ values and the independent variables $\rho_1,\rho_2,....,\rho_m$., on basis of the given sample, γ values (Siouris, 1996:63-82; Strejc, 1980).

Considering the function $f()$ to be a function of only a single variable γ and the two are linear. The term linear, implies that the mean of γ, $f\{\gamma\}$ is known to be a linear function of ρ. This model can be expressed as follows:

$$f\{\gamma\} = p_i(\rho_1, \rho_2, ..., \rho_m) + \varphi,\ i = 1, 2, ..., m \tag{3.24}$$

where the two unknowns are the error terms φ and unknown parameters p_i.

ρ_i – vector of deterministic independent variables, the predictor.

These unknown constants (or population parameters) are to be estimated from a sample of γ values with their associated values of ρ_i vector.

The more general *discrete time* regression model is given by (Dzhaparidze, et al., 1994; Strejc, 1980; Strejc, 1981):

$$z(k) = \sum_{i=1}^{N_a} a_i z(k-i) + \sum_{i=1}^{N_b} b_i u(k-i) + w(k) \tag{3.25}$$

where N_a and N_b are the highest order of the output and input polynomials.

3.6.6. AR and ARMA regression estimation

Auto-regression models can be classified into: AR (auto-regression), ARMA (auto-regression moving average) and the so called, NARMA (nonlinear auto-regression moving average) models (Dzhaparidze, et al., 1994; Li, Liu & Ding, 2017; Sarris & Eisner, 1973; Seong, 2007; Wang, Wang & Shen, 2016). An ARMA model can be generated by extending the general regression equation using the expression:

$$w(k) = \sum_{i=0}^{N_c} c_i w(k-i) \tag{3.26}$$

where c_i are the weighting factors.

Hence $w(k)$ is a linear combination of random and unknown values of the noise w at the time instants $(k-i)$, $i = 0, 1, ..., N_c$

The full ARMA model can then be written as (Chon & Cohen, 1997:168; Ding, et al., 2017; Lu, Ju & Chon, 2001:1117; Wang, et al., 2016):

$$z(k) = \sum_{i=0}^{N_a} a_i z(k-i) + \sum_{i=0}^{N_b} b_i u(k-i) + \sum_{i=0}^{N_c} c_i w(k-i),\ c_0 = 1 \tag{3.27}$$

This type of parameter estimation is justified for closed loop control. The output noise in this case consists of the noise acting on the system at the instant of observation and the noise transferred through feedback of the system (Strejc, 1981:2).

3.6.7. Bayes parameter estimation

Bayesian methods can be used on models where a priori information about the values of the parameters to be estimated is known. The information is formalized into a prior distribution on the parameter and estimators are then formed from the posterior distribution of the parameter given the data (Ching, et al., 2006; Fujimoto & Takaki, 2016; Keating, Mason & Sen, 1993; Miguez, Marino & Vazquez, 2018; Romeres, et al., 2016; Zhao, Huang & Liu, 2013).

The interest is to determine or estimate values for system parameters. Given a continuous random variable p as a parameter to be estimated and $\hat{p}$ as its estimate value, the interest is to find the values of $\hat{p}$ that will estimate values of $\hat{p}$ over the range of the given data. The *cost* can be assigned to all data pair defined by $C[p, \hat{p}]$ over the range of interest. The assumption is that the cost depends only on the error of the estimate. The error is defined by the equation (Romeres, et al., 2016; Siouris, 1996:82–83):

$$e = p - \hat{p} \tag{3.28}$$

The cost function is given by what is referred to as Bayes risk, (Ching, et al., 2006; Miguez, Marino & Vazquez 2018; Romeres, et al., 2016) defined as:

$$\beta(\hat{p}) = w\{C[p, \hat{p}(z)]\} \tag{3.29}$$

Equation (3.29) suggests that once the cost function and the a priori probability have been specified, the risk can be minimized. The risk can be minimized by the proper choice of $\hat{p}$. This means that the estimate minimizes the risk, Equation (3.30).

$$\beta \cong w\{C(\hat{p})\} = \varepsilon\{C(p - \hat{p})\} \tag{3.30}$$

3.7. State estimation

The process model state is defined as the minimum amount of information necessary to uniquely determine the dynamic behaviour of the system at all future moments given the current inputs and parameters of the system. State estimation is the process of determining the “uknown” state of the system from output measurements given the dynamic model of the system (Ching, et al., 2006; Muske & Edgar, 1997:311–325; Yang, et al., 2016).

Some processes may be defined by a finite number of parameters that define the internal behaviour of the system, for example, $x_1, x_2, ..., x_n$. These parameters can be collectively arranged to form a vector, $(x_1, x_2, ..., x_n)$ and are then referred to as a *state* of a process model (Roxin, 1997:4).

Some possibilities for state estimation are based on the objective of the study:

1) The system exists and its parameters are known,

2) The process model exists but some states are not measurable and some are. Non-measurable states are important for control, design and real-time control of processes.

3) Determination of the non-measurable from the measured states and measured outputs of the system.

In general, the number of state variables required for model development will be equal to the number of energy storage components of the process.

3.7.1. Linear state estimation

The estimation of the state of a linear system is performed using the well-established optimal linear estimation theory. Due to complexity involved in nonlinear models, the corresponding estimation theory tends to be very complex in comparison to linear estimation theory. The commonly used techniques to estimate unknown state for linear models include: least squares method, Kalman filter and the Leunberger observer.

Before embarking on the process of estimating the states of the system, it is important to determine if the system is obserevable. The process of testing if a system is observable or not, is known as *observability*. If a linear system has the state can be determined uniquely from the output measurements, the system is onsidered *observable* (Dochain, 2003; Muske & Edgar, 1997:311–325).

3.7.1.1. Least squares state estimation

The process disturbance vector models unmeasured disturbances to the process that will affect the process state. The aim of this approach is to estimate the state of the system in Equation (3.8) by minimizing the estimated process and the measurement disturbances. The measurement disturbance vector represents the error in the measuring device caused by measuring instrument noise.

3.7.1.2. Kalman filter

Kalman filter is one of the most used state estimation algorithms. The Kalman filter generates estimates of the state variables of the process by processing the measured output variable. In conjunction with a closed loop algorithm, the input to the process is processed to minimize the difference between the estimated state values and the actual states determined using measured data. The aim is to find the best estimate of the state vector $x(t)$ according to the linear combination of the measurementvector $z(t)$ and the present state estimate $\hat{x}(t)$ so as to minimize the cost function given by (Siouris, 1996:92–109):

$$E\{[x(t)-\hat{x}(t)]^T[x(t)-\hat{x}(t)]\} \to \min \tag{3.31}$$

The solution to this problem is given by the optimal estimator equation derived from the filter in (Siouris, 1996:96)

$$\hat{\dot{x}} = A\hat{x}(t) + K^*[z(t) - C\hat{x}(t)] \tag{3.32}$$

where K^* is the filter gain. Equation (3.37) is also known as Kalman-Bucy filter.

3.7.1.3. Leunberger observer

Leunberger observer uses a constant gain matrix to meet the cost function of form:

$$\begin{aligned}\hat{x}(k|k) &= \hat{x}(k|k-1) + L(z(k) - \hat{z}(k)) = \\ &= \hat{x}(k|k-1) + L(z(k) - C\hat{x}(k|k-1))\end{aligned} \tag{3.33}$$

where

L – is the observer gain matrix,

$z(k)$ – is the current observed output, and

$C\hat{x}(k|k-1)$ – is the predicted output.

The observer gain matrix is used to correct the predicted state estimate from the difference of the observed output and predicted output, Equation (3.33). The performance is based on the dynamic behaviour of *the reconstruction error* and the *stochastic properties of the process*, and the system and measurement disturbances $w(k)$ and $v(k)$ are ignored.

Since the stochastic properties are neglected, the Leunberger observer is a deterministic estimator that is employed when the stochastic model of the process is unknown and if the optimal filter cannot be employed. The advantage of the Leunberger observer is the reduced computational requirements (Muske, & Edgar, 1997).

The reconstruction error of this method is given by:

$$e(k+1) = (A - ALC)e(k) \tag{3.34}$$

The requirement for the reconstruction error is that the eigenvalues of the matrix $(A - ALC)$ must have *a moduli* strictly less than one. The eigenvalues of this matrix are also known as the *closed loop observer poles*. The procedure of determining the relevant eigenvalues for the correction error is known as *pole placement* technique. The choice of observer poles is a compromise between fast reduction of the reconstruction error (which requires that the poles be placed close to the origin), and sensitivity to measurement noise and modelling error, (which increases as the poles are moved closer to the origin (Muske & Edgar, 1997:324–325). The continuous time equivalent equation can be given by (Muske & Edgar, 1997:324)

$$\dot{e}(t) = (A - ALC)e(t) \tag{3.35}$$

$$e(t) = x(t) - \hat{x}(t|t)$$

3.7.2. Nonlinear state estimation

Nonlinearities of a dynamic process model increase the level of complexity to the state estimation problem. The complexity applies to the derivation as well as the implementation of the nonlinear estimator. Optimal estimator methods are not generally available for the nonlinear models, therefore nonlinear estimation problems are solved using suboptimal approach to estimation problem.

Continuous-time nonlinear system state space and output model equations respectively, can be represented by (Nicholson, et al., 2014; Muske & Edgar, 1997:327; Salau, Trierweiler & Secchi, 2014):

$$\dot{x}(t) = f(x(t), u(t), t) \tag{3.36}$$

$$z(t) = g(x, t)$$

The assumption is that the model parameters are known and are not included in the argument list. The discrete time equivalent models are:

$$x(k+1) = f(x(k), u(k), k) \tag{3.37}$$

$$z(k) = g(x(k), k)$$

3.7.2.1. Extended Kalman filter

The easiest method to estimate state estimation is to linearize a nonlinear model and then to apply linear state estimation to the linearized system. The extended Kalman filter is one of the methods that can be applied to achieve this type of estimation. The extended Kalman filter computes a state estimation at each sampling time by the use of the Kalman filter on a linearized model of a nonlinear system.

3.7.2.2. Recursive state estimation methods

Siouris (1996) states that in recursive estimation methods, the prior estimate can be used as the starting point for a sequential estimation algorithm. The algorithm assigns a proper relative weighting to the old and new data. Assuming a given linear system with Gaussian noise, the state estimates of such a system will be determined from the conditional probability density of the state given the output measurements. The conditional probability density of the state with Gaussian noise is Gaussian but for a nonlinear system, it is not Gaussian even if the state and measurement disturbances, $w(t)$ and $v(t)$ are Gaussian. The determination of the conditional probability density requires the nonlinear model Equation (3.38) be represented as vector nonlinear stochastic differential equation (Muske & Edgar, 1997):

$$dx(t) = f(x(t), u(t), t)dt + d\beta \tag{3.38}$$
$$d\beta = w(t)$$

3.7.2.3. Gradient state estimation methods

Gradient methods are a general purpose unconstrained optimization methods used to estimate parameters. These methods can be broadly divided into two categories; *direct search methods* and *gradient search methods*. Gradient search methods require derivatives of the objective function whereas the direct search methods are derivative free. The derivative may be determined analytically or be approximated in some way. The direct search techniques which use function values which are more effective for highly discontinuous functions (Srinivasan, Francois & Bonvin, 2011; Englezos & Kalogerakis, 2001)

The aim of the gradient methods is to reduce value of the cost function at each moment of iteration. The basic problem is formulated in the following way: find the parameter vector k that minimizes the function, $S(k)$ (Englezos & Kalogerakis, 2001),

$$S(k) = \sum_{i=1}^{N} e_i^T Q^*{}_i e_i \tag{3.39}$$

where

$e_i = (z_i - f(x_i, k))$ – is the m dimensional vector of the residuals,

with $e = [e_1, e_2, \ldots.., e_m]^T$.

Commonly used types of gradient methods include: steepest descent, Newtons, quasi-Newton (variable metric or Secant method) and conjugate gradient. Direct methods use function evaluations. They search for the minimum of an objective function.

3.8. Integrated approximation methods

The integrated approximation techniques use the extended state vector which includes all parameters to be estimated. For the discrete time system the problem of estimating unknown states is solved by minimizing the cost function given by:

$$J(\zeta) = E\{\theta[\zeta(p)] - \theta_0[\zeta(p_0)] + \sum_{k=k_0}^{N} f[\zeta(p), u(k), w(k), k]\} \tag{3.40}$$

where

$E\{\ \}$ – is the mean value,

θ – is the vector of combined unknown states and parameters,

$\zeta(p)$ – is the extended state vector, which includes all unknown parameters,

$w(k)$ – is a vector of stochastic disturbance with presumed known probability density function.

3.9. Optimization methods in parameter and state estimation

In optimization of a system, the idea is to set possible operating conditions so as to obtain the best possible results given some limits of the system. Accompanying such an idea is the concept of defining the system and its limitations. The meaning of best possible (*optimal*) results must be clearly identified, system *boundaries*, system *limits and constraints*, *selectable operating conditions* and the manner of making the selection, known as the *optimization technique* must also be identified.

In parameter estimation, the process of adjusting parameters such that model output based on predictions matches that received from process measured data is in real terms a minimization problem. This is achieved by minimizing the objective function given by Equation (3.9). Parameters are continuously adjusted until the model output matches that of measured data.

There are a few different forms the criterion takes depending on the minimization technique to be applied. These include: 1) *sum of squared errors*, 2) *maximum likelihood* (ML), 3) *weighted least squares* (WLS), also known as *Chi squared* (χ^2), 4) *ordinary least squares,* 5) Bayesian. The most common of these is the sum of squared errors function. The other functions are all some form of extension of the maximum likelihood method, using different assumptions for each (Dochain & Vanrolleghem, 2001:215–225). In some cases, it is possible to have a multivariable objective function where more than one variable has been measured.

In the state estimation problem, the objective function is formulated in two different ways, depending whether the solution is, 1) an observer or 2) a filter. In the case of the observer, the objective function does not follow the same structure as that in the parameter estimation problem; and in the case of a filter, it takes the same structure. The two methods of solution are derived around the error difference between the measured states (obtained from the output) and the model estimated states.

The observer design follows two stages: 1) that of minimizing the error between the actual states $x(t)$ and the estimated states $\hat{x}(t)$ as t approaches infinity:

$$e_x(t) = x(t) - \hat{x}(t) \to \min \quad \text{for } t \to \infty \tag{3.41}$$

2) and that of keeping the rate of change of the error at zero (or near zero).

$$\dot{e}_x(t) = 0 \tag{3.42}$$

This rate of change must stay at or close to zero for all the input and state values.

In the filter design approach, the objective is that optimal estimates should minimize the objective function $J_{\tilde{x}}(t)$ based on the difference between estimated states and real states obtained from the measurement data. The objective function does not appear as the only equation to be solved when dealing with a filter. The algorthim involves solving a number of sequential equations based on measurement updates. The final equation for the procedure is based on the objective function given as:

$$\begin{aligned} J_{\hat{x}}(t) &= E\left\{e_x(t)^T e_x(t)\right\} = \\ &= E\left\{(x(t) - \hat{x}(t))^T (x(t) - \hat{x}(t))\right\} \end{aligned} \tag{3.43}$$

where $E\{\ \}$ – is the covariance index (expectation operator), (Siouris, 1996:96).

3.10. Parameter and state estimation algorithms

3.10.1. Parameter estimation algorithm

Parameter estimation procedure indicated here is for nonlinear system using optimization techniques. Dochain and Vanrolleghem (2001:226–228) give the generalised parameter estimation procedure. The accompanying algorithm for the parameter estimation procedure is given in Figure 3.4.

At the start of the procedure, guesses of parameter values must be provided and the model structure and measurement data must also be available. The very first algorithm procedure will calculate predictions based on these initial parameter values.

Model predictions are determined from solving the set of model equations; these values are then passed on to calculate the objective function, where the prediction's confidence is tested. If the test criterion is not met based on the applicable rule of the algorithm, determined by the user, a new set of parameters is proposed and sent to the solution step, mentioned above. If the criterion is met, the calculated parameter values are considered the optimal ones and are passed to the model for use. Alternatively, if the criterion is never met, a stopping condition is applied; usually this is based on some practical big number, which specifies how many times should the algorithm be run over should the criterion be never satisfied.

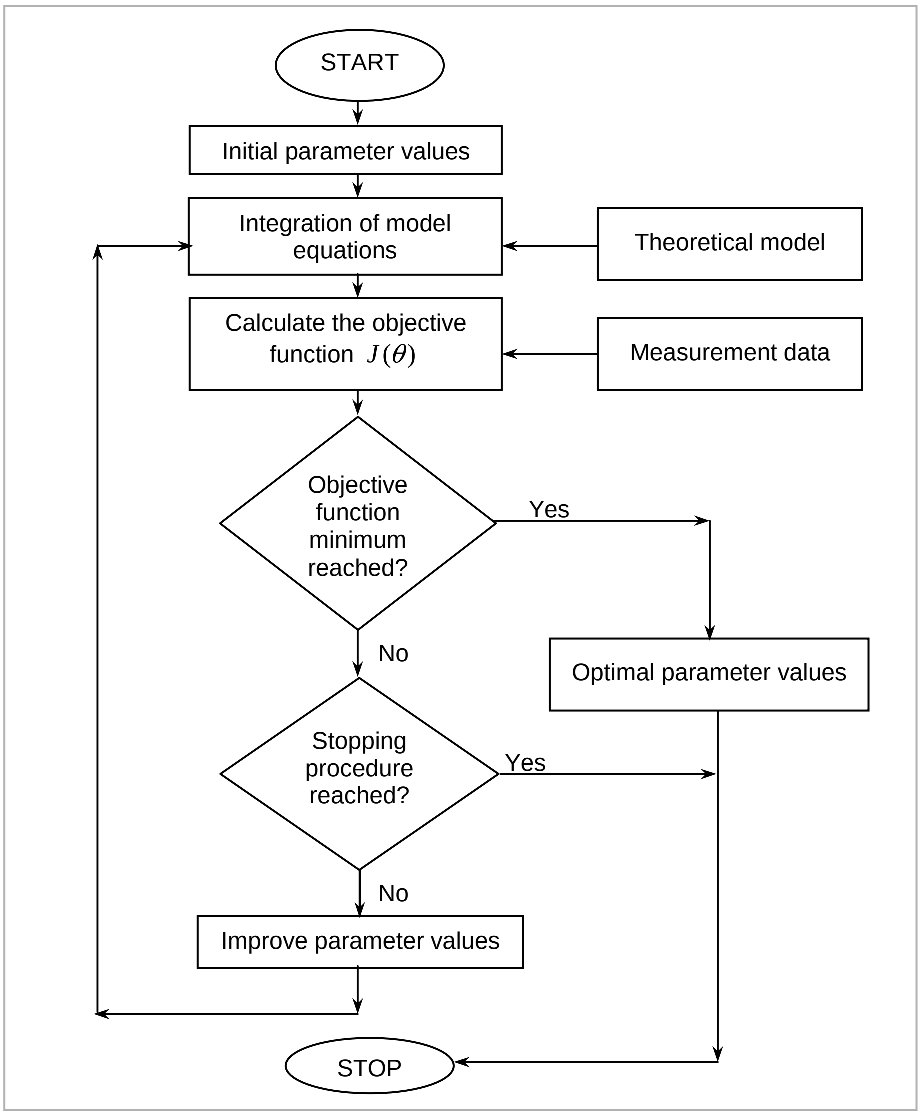

START
Initial parameter values
Integration of model equations
Theoretical model
Calculate the objective function $J(\theta)$
Measurement data
Objective function minimum reached?
Yes
No
Optimal parameter values
Stopping procedure reached?
Yes
No
Improve parameter values
STOP

Figure 3.4. **Algorithm for linear parameter estimation based on minimization procedure**

(Adapted from Dochain & Vanrolleghem, 2001:228)

3.10.2. State estimation algorithms

There are many different approaches to state estimation problem that can be classified as *maximum likelihood*, *least squares* or *Bayesian methods*. In this thesis, the state estimation problem is solved based on the two well known methods, the observer design and Kaman filter based procedures. Algorithms presented are therefore for these two cases.

The observer design

The aim of the observer is to estimate unknown state of a system, and the procedure requires the knowledge of the input and output data over the specified period $k = \overline{1,K}$ (Bornard, Celle-Couenne, & Gillles, 1995:176). An estimation (prediction) problem posed as a *least-squares* problem irrespective of application can be solved as a *non-sequential* (known as *batch*) estimation, or as a *sequential* (known as a *recursive*) estimation problem. In the non-sequential solution all measurements up to k are required for the solution whereas in the recursive solution, only data from the previous sample are necessary for the solution (Muske & Edgar, 1997:321).

The basic observer design is a recursive estimator and the estimator constant gain matrix is used to meet some stipulated performance criteria. The system reconstructs the system state using input–output measurement information, thus the name *observer* (Primbs, 2006; Siouris, 1996:164). The performance criterion is based on the estimator reconstruction error (Muske & Edgar, 1997:321; Primbs, 2006). The observer is a *deterministic operation* that ignores the stochastic behaviour of the system by ignoring the system disturbance and the measurement noises (Wang & Zhang, 2006; Muske & Edgar, 1997:321).

Linear case

For a linear discrete-time system presented by

$$x(k+1) = Ax(k) + Bu(k) \tag{3.44}$$

$$z(k) = Cx(k)$$

The full-order observer for such a system may be represented by

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L[\hat{z}(k) - z(k)] \tag{3.45}$$

$$\hat{z}(k) = C\hat{x}(k)$$

where $\hat{x}(k)$ – is the estimate of the true state $x(k)$.

The observer gain must be chosen such that the estimation error $e(k) = \hat{x}(k) - x(k)$ converges to zero, Equation (3.41).

$$e(k) = [\hat{x}(k) - x(k)] \to 0 \tag{3.46}$$

This means that the estimate $\hat{x}(k)$ converges to the true state $x(k)$ exponentially. The observer design becomes the exercise of determining the values of the observer gain matrix L. The determination of the observer gain can be solved using different methods, such as the pole placement method, constant solution of a Riccati equation etc. If the system is observable, that is, its state parameters and observation matrices pair (A,C) is an observable pair, by placing the observer matrix ($A-LC$) on the negative real part of operational spectrum, the estimation error will converge to zero, $e(k)=[\hat{x}(k)-x(k)]\rightarrow 0$ (Bornard, et al, 1995:182). The procedure for the observer algorithm according to the Equations (3.22)–(3.27) can be presented as follows:

1) Initialization: set model parameters matrices A, C and the initial estimate $x(0|0)=x_0=x(0)$ and initial error, $e_0(k)=e(0)$, set a large number M iterations for stopping of the calculation procedure,
2) set the initial discrete-time moment $k=1$,
3) set observer trajectory–set real output data for the full trajectory $k=1,2,...,K$, $u(k)=u(1),u(2),...,u(K)$,
4) state prediction – calculate the state estimate $x(k+1|k)$,
5) calculate the error difference based on estimated state and the real state from measurement data, $e(k+1|k)$,
6) store the current value of state estimate $\hat{x}(k+1|k)$ in a vector,
7) determine the value of the gain matrix L, using pole placement method based on the requirement that $\dot{e}(k+1|k)\rightarrow 0$
8) increment the discrete moment count, $k=k+1$,
9) test the stopping procedure, $e(k)\leq \varepsilon_{x(k)}$

    if 9) not true, repeat from step 4)

    if 9) true continue to step 10,
10) increment the stop procedure index $i=i+1$
11) test if large number stopping procedure is reached, $i=M$

    if 11) not true, repeat from step 4)

    if 11) true, end procedure.

Nonlinear case

Attempts have been made to design nonlinear observers using nonlinear systems theory. Such has been achieved in the nonlinear extended Kalman filter, but has not been truly extended to observer design methods (Muske & Edgar, 1997:346; Primbs, 2006). In the same fashion as in the linear case, the nonlinear observer design considers the deterministic behaviour of the state reconstruction error (Muske & Edgar, 1997:346. The idea is to therefore find a transformation that can linearize the reconstruction error. Many authors have presented many methods to deal with this limitation, methods such as, *error linearization* methods, *extended Luenberger* observers, *sliding mode* observers, *Lyapunov functions* based solutions, *Lie-algebra* methods, etc.

A deterministic discrete-time nonlinear system of the form

$$x(k) = f(x(k), k) \qquad (3.47)$$

$$z(k) = g(x(k, k)$$

is considered.

Attempting this problem using linear approach considers the estimation error and its dynamics. This will require an observer with *linear output injection* and error dynamics of the form (Primbs, 1996):

$$\hat{x}(k+1) = f(\hat{x}(k), k) + L[\hat{z}(k) - z(k)] \qquad (3.48)$$

$$\hat{z}(k) = g(\hat{x}(k), k)$$

where $L \in R^{n \times p}$ is the observer gain matrix,

$$e(k) = \hat{x}(k) - x(k) \qquad (3.49)$$

$$e(k+1) = f(\hat{x}(k), k) - f(x(k), k) + L[h(\hat{x}(k), k) - h(x(k), k)] \qquad (3.50)$$

The error dynamics is a nonlinear function and its stability is not clearly identical (Primbs, 1996). One approach to solving this problem is the attempt to linearize the error dynamics, about a fixed point $e(k) = 0$.

$$\begin{aligned} e(k+1) = f[(x(k), k) + e(k)] - f(x(k), k) + \\ + L\big[h[(x(k), k) + e(k)] - h(x(k), k)\big] \end{aligned} \qquad (3.51)$$

The procedure for observer state reconstruction methods tends to be model specific, Figure 3.5. In this sense, only the linear state estimation observer procedure has been presented. It gives enough understanding into steps that must be followed to solve the state estimation problem using observer design technique even for nonlinear systems. The final solution can be presented as

$$e(k+1) = \left[\frac{\partial f(x(k))}{\partial x(k)} + L\frac{\partial h(x(k))}{\partial x(k)}\right] e(k) \qquad (3.52).$$

The Kalman filter procedure

The Kalman filter is very attractive in real-time estimation applications, since it presents a simple and efficient algorithm. The considered system models for

Kalman filter solutions are for the linear and nonlinear, time-invariant, discrete-time systems, i.e., the system has constant coefficients over the observed period.

The linear case

The conventional discrete-time Kalman filter algorithm for a stochastic time-invariant or slowly varying linear system is based on the following Equations (3.44)–(3.46):

1) the system and observation model,

$$x(k+1) = A(k+1,k)x(k) + W(k)w(k)\,,\ k = 0,\ 1,\ 2,.... \tag{3.53}$$

$$z(k) = C(k)x(k) + v(k)$$

2) the filter noises statistics (described by the system and observation noise)

$$w(k) \approx N(0, Q(k)) \tag{3.54}$$

$$v(k) \approx N(0, R(k))$$

3) the prediction model (including the estimation error statistics)

$$\hat{x}(k+1|k) = A(k+1|k)\hat{x}(k|k) \tag{3.55}$$

$$P(k+1|k) = A(k+1|k)P(k|k)A^T(k+1|k) + W^*(k)Q(k+1)W^{*T}(k) \qquad (3.56);$$

this model can be expressed in a different notation, as

$$\hat{x}(k|k) = x(k|k-1) + K^*(k)[z(k) - C(k)\hat{x}(k|k-1)] \tag{3.57}$$

$$P(k|k) = P(k|k-1) - K^*(k)C(k)P(k|k-1) \tag{3.58}$$

4) the Kalman gain for measurement update

$$K^*(k) = P(k|k-1)C^T(k)[C(k)P(k|k-1)C^T(k) + R(k)]^{-1} \tag{3.59}$$

where $x(k)$ – is the current state value of the system,

$x(k+1)$ – is the state value at the next discrete moment of k,

$z(k)$ – is the current value of observation measurements,

$w(k)$ and $v(k)$ – are system and observation noise values respectively, that are assumed white, Gaussian and with zero mean;

- the system matrices

$A(k+1,k)$ – the transition matrix used to transform a given state at moment k to the next moment $k+1$,

$A(k+1|k)$ – is the state transition matrix at $k+1$ given $z(k)$ measurement,

$W(k)$ – the system noise coefficients matrix,

$C(k)$ – is the system output observer matrix;

- the estimator variables

$\hat{x}(k+1|k)$ – is the estimated state at time $k+1$ given the measurement $z(k)$,

$\hat{x}(k|k)$ – is an unbiased real estimate of $x(k)$ given the measurements of $z(k)$;

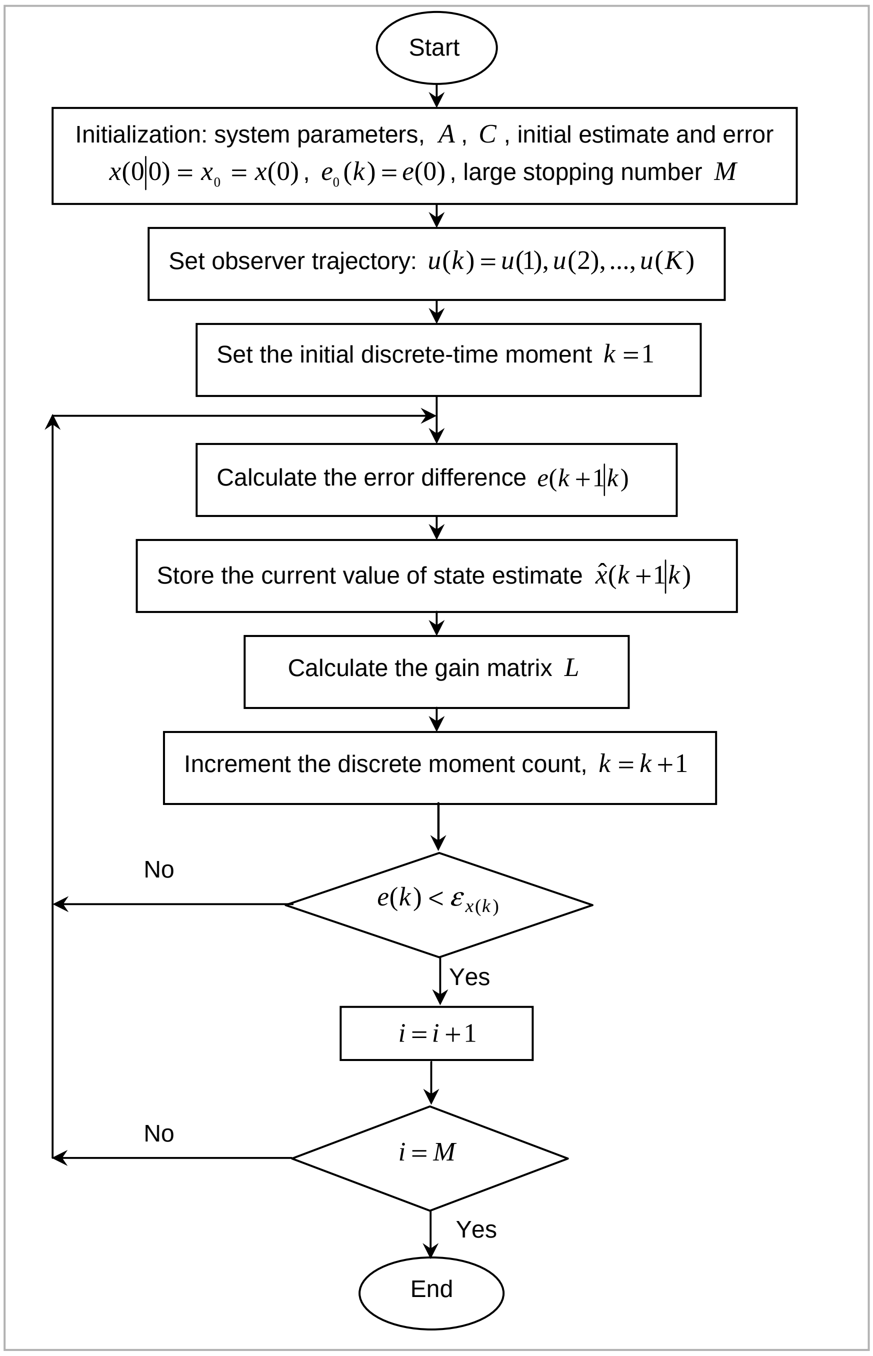

Start
Initialization: system parameters, A, C, initial estimate and error $x(0|0) = x_0 = x(0)$, $e_0(k) = e(0)$, large stopping number M
Set observer trajectory: $u(k) = u(1), u(2), ..., u(K)$
Set the initial discrete-time moment $k = 1$
Calculate the error difference $e(k+1|k)$
Store the current value of state estimate $\hat{x}(k+1|k)$
Calculate the gain matrix L
Increment the discrete moment count, $k = k+1$
No
$e(k) < \varepsilon_{x(k)}$
Yes
$i = i+1$
No
$i = M$
Yes
End

Figure 3.5. **Algorithm for nonlinear state estimation based on minimization procedure**

Based on the given noise assumptions above, the following noise statistics are deduced for the estimator,

$E\{w(k)\} = 0$ – is the system noise probability,

$E\{v(k)\} = 0$ – is the measurement noise probability,

$$\left.\begin{array}{l} E\{w(k)v^T(k)\} = 0 \\ E\{v(k)w^T(k)\} = 0 \end{array}\right\} \text{ for all values of } k,$$

$E\{w(k)w^T(k)\} = Q(k)$,

$E\{v(k)v^T(k)\} = R(k)$;

- the estimator matrices

$Q(k)$ – is the system noise covariance matrix, used to relate errors in the state propagation to the uncertainty of the current estimate,

$Q(k+1)$ – is the system noise covariance matrix at the next moment $k+1$

$R(k)$ – is the measurement noise covariance matrix,

$P(k|k)$ – is the error covariance matrix that represents the covariance of the error difference between the true state and its estimate,

$P(k+1|k)$ – is the covariance error matrix at the next moment given the current error difference, and

$K^*(k)$ – is the Kalman filter gain used to update measurements.

The procedure for implementing the filter is as given below. At the beginning of the procedure for a linear filter as applied to a linear dynamic state model of a system, the following filter properties must be provided: 1) the filter dynamics, A, W and C, 2) the noise statistics, both the system and measurement noise covariances, $Q(k)$ and $R(k)$, and 3) the a priori data $[\hat{x}(0), P(0)]$. The implementation is done over two main steps, 1) the *prediction* step and 2) the *update/correction* step as presented by Siouris, (1996:111–124), and Ikonen and Najim, (2002:42–44).

Step 1 – Prediction

1) initialization – set the initial state values and initial prediction values,

$$k = k_0,\ \hat{x}(k|k) = \hat{x}(k_0|k_0) = x_{k_0}, \text{ and } P(k_0|k_0) = P_{k_0} \qquad (3.60)$$

2) state prediction – set the initial count $k = 1$ and the state estimate,

$$\hat{x}(k+1|k) = A(k+1|k)\hat{x}(k|k) \qquad (3.61)$$

3) observation prediction – set the initial observation prediction,

$$\hat{z}(k+1|k) = C\hat{x}(k+1|k) \qquad (3.62)$$

4) innovations calculation – determine innovations using the initial observation prediction,

$$v(k) = z(k) - \hat{z}(k+1|k) \qquad (3.63)$$

5) covariance prediction – calculate the covariance matrix for the error in the current estimate $\hat{x}(k+1|k)$,

$$P(k+1|k) = A(k)P(k|k)A^T(k) + \overline{A}(k)Q(k)\overline{A}^T(k) \qquad (3.64)$$

where $\overline{A}(k) = A(k)K^*(k)$

Step 2 – Update/correction

1) obtain new measurement – observe the next measurement,

$$z(k+1) = Cx(k+1) \qquad (3.65)$$

2) innovation covariance – determine the innovation covariance to use in calculating the filter gain,

$$S(k+1) = CP(k+1|k)C^T + R(k)^{-1} \qquad (3.66)$$

3) determine the filter gain,

$$K^*(k+1) = P(k+1|k)C^T[S(k+1)]^{-1} \qquad (3.67)$$

4) state correction – update the state estimate at $k+1$ moment using the filter gain and new measurement,

$$\hat{x}(k+1|k+1) = x(k+1|k) + K^*(k)v(k) \qquad (3.68)$$

5) update covariance – update the covariance matrix based on new measurement and the filter gain,

$$P(k+1|k+1) = P(k+1|k) + K^*(k)S(k+1)K^{*T}(k) \qquad (3.69)$$

6) obtain the next sample – increase the sample index

$$k = k+1 \qquad (3.70)$$

Figure 3.6 represents the classical linear Kalman filter algorithm for estimating a system state. The classical linear Kalman filter is only applicable to linear systems with linear measurements including both the system noises and measurement noises. The system noises are considered the driving dynamics of the filter. The driving noises and measurement noises are considered uncorrelated.

The nonlinear case

For nonlinear systems of which most physical systems fall under, an extended Kalman filter was introduced. If a Kalman filter for such systems is to be designed for estimating unknown states, it means, the system must first be linearized (linear approximated) before designing the filter. Such a Kalman filter concept is

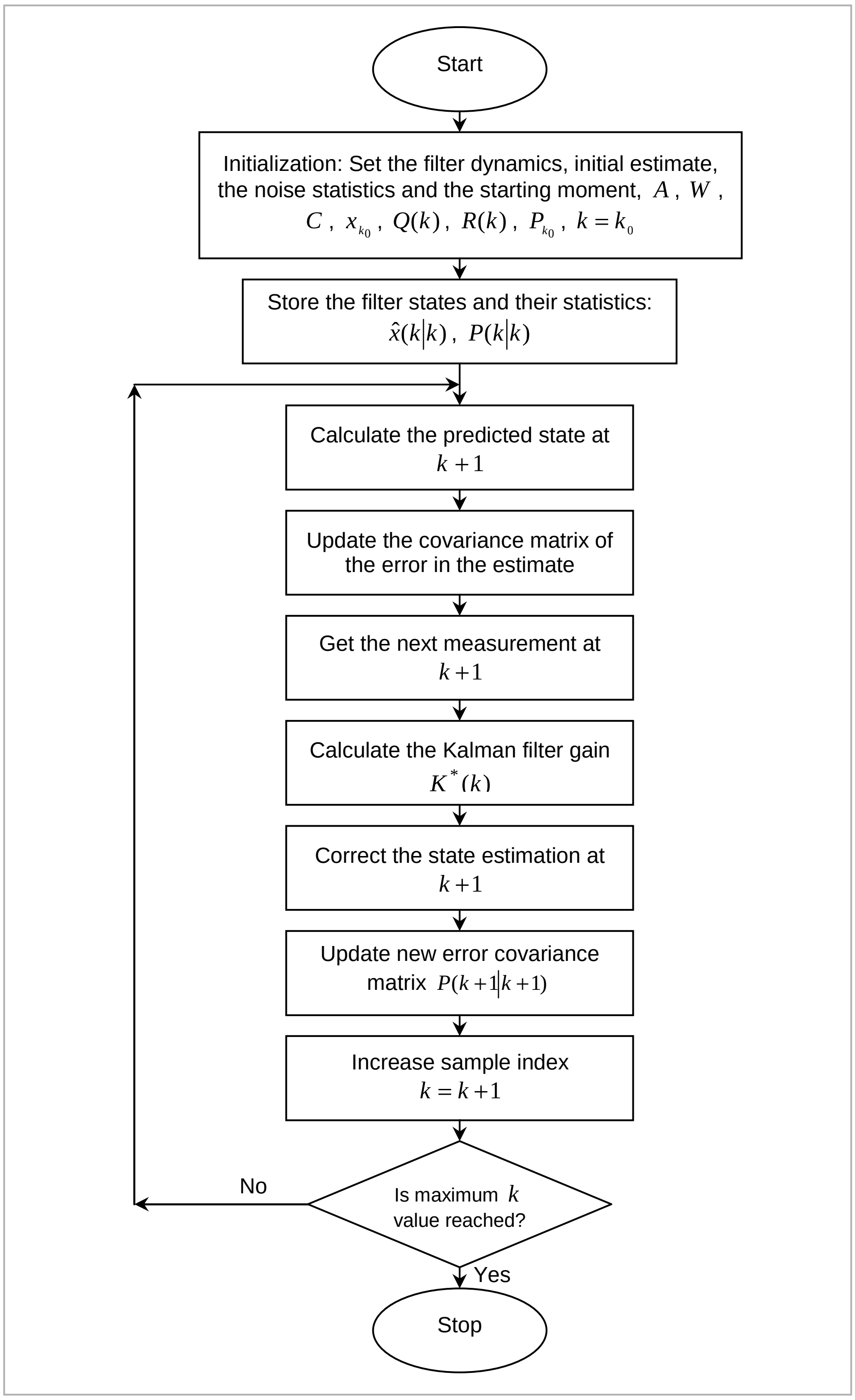

Start
Initialization: Set the filter dynamics, initial estimate, the noise statistics and the starting moment, A, W, C, x_{k_0}, $Q(k)$, $R(k)$, P_{k_0}, $k = k_0$
Store the filter states and their statistics: $\hat{x}(k|k)$, $P(k|k)$
Calculate the predicted state at $k+1$
Update the covariance matrix of the error in the estimate
Get the next measurement at $k+1$
Calculate the Kalman filter gain $K^*(k)$
Correct the state estimation at $k+1$
Update new error covariance matrix $P(k+1|k+1)$
Increase sample index $k = k+1$
Is maximum k value reached?
No
Yes
Stop

Figure 3.6. Algorithm for the basic Kalman filter

(Adapted from Ikonen & Najim, 20022:113–124 and Siouris, 1996:111–125) referred to as the extended Kalman filter (EKF). The extended Kalman filter is for use in nonlinear systems. Nonlinear systems can be classified according to noise considerations of a system model. Nonlinear systems are considered deterministic if noise disturbances are not included in the model, and stochastic if the noise disturbances are considered, either the system noise or measurement noise or both (Mukkula & Paulen, 2017; Siouris, 1996:190–191).

In its application for estimation of nonlinear systems, the extended Kalman filter assumes linearization in the system being estimated. The extended Kalman filter can also be applied in very unique systems that may be continuous-time systems but have discrete measurements, such a Kalman filter extension is referred to as the continuous-time discrete-time extended Kalman filter (CDEKF) (Siouris, 1996:190).

The procedure for applying a nonlinear extended Kalman filter is not so different from the linear Kalman filter except for the part where the nonlinear system is to be linearized. In the extended Kalman filter algorithm, the dynamics model and observation model become nonlinear systems. Siouris (1996) states that the system state and the covariance matrices are presented in the same form as that of the linear Kalman filter, Equations (3.53)–(3.59). Also the noise statistics stay the same as in the linear case.

These models are presented as follows:

1) The system and observation models,

$$x(k+1) = A(x(k),k) + W(x(k),k)\,w(k)\,,\ k = 0,\ 1,\ 2,\ldots \tag{3.71}$$

$$z(k) = h(x(k),k) + v(k)$$

2) The state and state estimation equations,

$$x(k+1|k) = A(k+1,k)x(k) + w(k) \tag{3.72}$$

$$P(k|k-1) = A(k|k-1)P(k-1)A^T(k,k-1) + Q(k) \tag{3.73}$$

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K^*(k)[z(k) - C(k)\hat{x}(k|k-1)] \tag{3.74}$$

$$P(k|k) = [I - K^*(k)C(k)]P(k|k-1) \tag{3.75}$$

where $K^*(k) = P(k|k-1)C^T(k)[C(k)P(k|k-1)C^T(k) + R(k)]^{-1}$ (3.76)

Based on these system of equations, Equations (3.34)–(3.39) and the corresponding equations from the linear Kalman filter, the procedure for implementing the nonlinear extended Kalman filter is as given here and demonstrated in Figure 3.7.

1) Initialization:

set the initial state and the initial system covariance matrix

$$\hat{x}(0|0) = \hat{x}(0)\,,\ P(0|0) = P(0) \tag{3.77}$$

2) Set the discrete moment k and the count indices i to one, and the first estimate $\xi_i(k)$

$k = 1\,,\ i = 1$

$$\xi_i(k) = x(k|k-1) \tag{3.78}$$

3) read measurement vector – filter input data

$$z(k) = C(k)x(k) + v(k) \tag{3.79}$$

4) read noises covariances – filter input data

$$R(k)\,,\ Q(k) \tag{3.80}$$

5) calculate observation model $C_i(k)$

$$C_i(k) = \left.\frac{\partial g(x(k),k)}{\partial x(k)}\right|_{\xi_i(k)} \tag{3.81}$$

where $g_i(k) = g[\xi_i(k), k]$

6) determine the filter gain matrix $K^*{}_i(k)$

$$K^*{}_i(k) = P(k)C_i^T(k)[C_i(k)P(k)C_i^T(k) + R(k)]^{-1} \tag{3.82}$$

7) increment the i index

$i = k + 1$

8) calculate filter estimation value, $\xi_i(k)$

$$\begin{aligned}\xi_i(k) &= \hat{x}(k|k-1) + \\ &\quad + K^*{}_{i-1}\big[z(k) - g_i(k) - C_{i-1}[\xi_{i-1}(k) - x(k|k-1)]\big]\end{aligned} \tag{3.83}$$

9) test if minimum (optimal) estimate is reached

$$|\xi_i(k) - \xi_{i-1}(k)| \geq test\ value \tag{3.84}$$

» if reached go back to step 5), calculate observation model,

» if not, continue to step 10),

10) determine current state estimate at the moment $\hat{x}(k)$

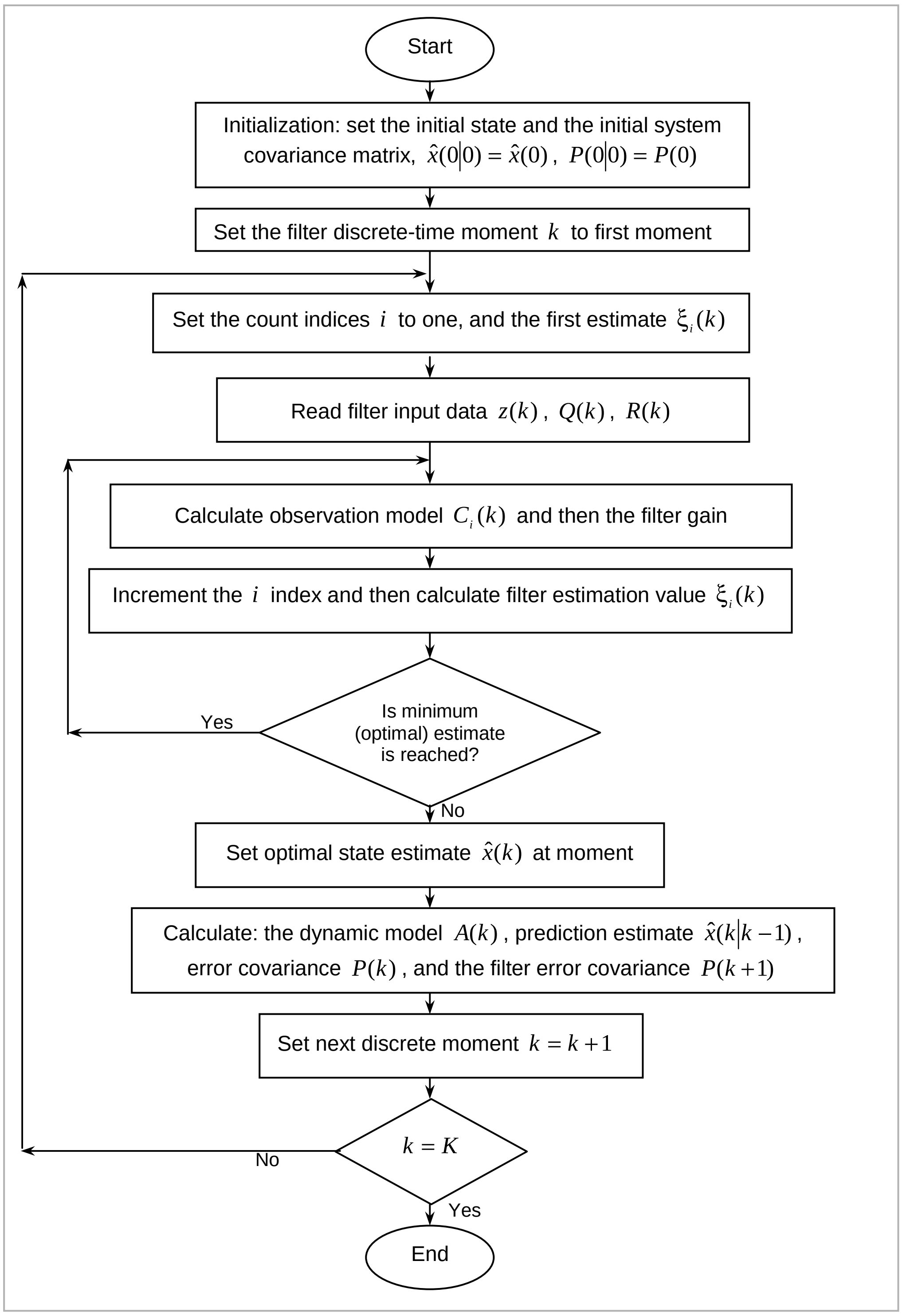
$$\hat{x}(k) = \xi_i(k) \tag{3.85}$$

11) calculate the state dynamic model, $f[x(k), k]$,

$$A(k) = \left.\dot{f}(x(k),k)\right|_{\xi_i(k)} \tag{3.86}$$

12) calculate state prediction estimate

$$\hat{x}(k+1|k) = f[\hat{x}(k), k] \tag{3.87}$$



Start
Initialization: set the initial state and the initial system covariance matrix, $\hat{x}(0|0) = \hat{x}(0)$, $P(0|0) = P(0)$
Set the filter discrete-time moment k to first moment
Set the count indices i to one, and the first estimate $\xi_i(k)$
Read filter input data $z(k)$, $Q(k)$, $R(k)$
Calculate observation model $C_i(k)$ and then the filter gain
Increment the i index and then calculate filter estimation value $\xi_i(k)$
Is minimum (optimal) estimate is reached?
Yes
No
Set optimal state estimate $\hat{x}(k)$ at moment
Calculate: the dynamic model $A(k)$, prediction estimate $\hat{x}(k|k-1)$, error covariance $P(k)$, and the filter error covariance $P(k+1)$
Set next discrete moment $k = k+1$
$k = K$
No
Yes
End

Figure 3.7. **Algorithm for nonlinear extended Kalman filter**

(Adapted from Siouris, 1996:111–125)

13) calculate error covariance, $P(k)$

$$P(k) = \left[I - K^*(k)C(k)\right] P(k-1) \tag{3.88}$$

14) work out predictor error covariance $P(k+1)$

$$P(k+1) = A(k)P(k)A^T(k) + W^*(k)Q(k)W^{*T}(k) \tag{3.89}$$

15) increase discrete time step $k = k+1$

Each of the filter stages (linear or nonlinear) contain discrete-time equations for propagation of the estimate and its covariance matrix; and in cases where a priori information is not available, batch processing can be considered (Siouris, 1996:121–122). In case of batch processing all measurements are simultaneously incorporated into the estimate. Another important property of the filter is the *stability*. There are many different techniques that were developed to deal with the Kalman filter stability, Siouris (1996) deals with a few of these methods.

3.11. Conclusion

There are many reasons associated with building a model; models are built for simulation of the dynamic behaviour of processes, for prediction and or analysis of the process behaviour, prediction of time evolution behaviour of the process, but generally, models are developed for monitoring with the main aim of being able to control the process optimally. If one could control the process based on some knowledge or understanding of its future behaviour, it gives more scope in terms of control and economic benefit associated with plant operations more especially the process feeds costs.

The focus of this chapter is the different methods used in solving nonlinear parameter estimation and nonlinear state estimation problems. Due to complexity relating modelling of nonlinear processes and application of estimation techniques the chapter is subdivided into three sections with focus on application of modelling theory, estimation theory (parameter estimation and state estimation theory) and then the application of optimization theory to solve these estimation problems. Different methods used for parameter estimation have been presented. For the state estimation problem, different estimators and some algorthims have been presented.

The next chapter deals with the continuous countercurrent ion exchange (CCIX) process model construction, and the extension of the model for parameter and state estimation to allow capabilities of applying it in optimal control. The developed model is a bilinear state model.

CHAPTER FOUR

ION EXCHANGE PROCESS AND ITS MODEL DEVELOPMENT

CHAPTER FOUR

4. ION EXCHANGE PROCESS AND ITS MODEL DEVELOPMENT

4.1. Introduction

In this chapter the continuous countercurrent ion exchange (CCIX) process is discussed. The selection of the CCIX was determined by the availability of the plant at the Chemical Engineering Department of Cape Peninsula University of Technology. The plant is based on an earlier design that was implemented at the University of Cape Town in 1982 and the plant measurement data used came from this study.

The difference between the newly designed control system and the previous one is that the current process modelling is based on state space representation in which optimization techniques are directly implementable and allows better system analysis. This allows wider system handling capabilities, provides widely and well established optimization techniques in parameter and state estimation methods, and relatively easy solution methods.

Using the CCIX process, a bilinear model has been developed using component balance equations. Different main operating methods (batch and fluidized bed) are presented and their different operational techniques. The model was initially developed for the MTech qualification (Dube, 2002); and is reviewed and extended for the purpose of parameter and state estimation.

The chapter starts with the general discussion on ion exchange process and its technical detail to the ionic level. The material balance of the process is discussed in detail giving rise to the usage of the mass balance equation to describe the internal component characteristics of the exchanging ions and their behaviour at this level. The process model is then developed with relevant accompanying assumptions. Continuous and discrete time model structures are given. The chapter ends with the state space discrete time model generated for application of different thesis solutions based on computer software programs developed.

4.2. The ion exchange process

An ion exchange process is a reversible chemical process where ions with similar polarities are exchanged between solids and an electrolyte solution if the two are contacted. This permits the separation and fractionation of electrolyte solutes. During the exchange process, there is an interaction of the ions with solid phase and also diffusion of ions within the solid phase (Hendry, 1982a; Hendry, 1982b; Rochette, 2006; Sasan, et al., 2017; Treybal, 1980:566).

According to Treybal (1980:566) and Dyer (2013) the ion exchange process resembles adsorption process in its behaviour and the ion exchange process itself tends to be more complex than the concept of exchanging of ions. Streat (1995:2841) argues that ion exchange process differs from adsorption in that the adsorbent material used is a functional group or dynamic polymer matrix or an inorganic structure with exchangeable functional groups.

The countercurrent ion exchange process configuration consists of four columns and two phases (Hendry, 1982a; Hendry, 1982d; Randal, 1984; Steinebach, et al., 2017). Each phase is made up of two columns of the four columns. The two phases are cation exchange and anion exchange. Each of the phases consists of a loading column and a regeneration column, Figure 4.1. It is stated in Hendry (1982a:1) report that there are two major decisions to be considered when designing an ion exchange plant; 1) *the nature of the contacting device* to be used and, 2) *the type of the regenerant chemicals* to be used. Contacting devices may be either of a fixed bed type or a moving bed type. In this project, the process considered is the CCIX process for treatment of salt water. In this case the process is based on the fluidized moving bed contact device (Bochenek, et al., 2011; Hendry, 1982a; Hendry, 1982b).

Ion exchange has found many applications including treatment of ore slurries for collection of valuable metals, water softening, complete water deionization and treatment and concentration of dilute waste solutions (Cappelle & Davis, 2016; Grafa, Cornwell & Boyer, 2014; Treybal, 1980:642). Water upgrading and recovery systems can be divided into two main categories (Hendry, 1982b:4):

1) systems with desalination function, and
2) systems that perform tertiary treatment function.

According to Hendry (1982b, 1982c, 1982d, 1982e) the developed systems at the time could not provide both functions of desalination and tertiary treatment in one treatment; usually systems had to be cascaded in order to achieve both treatment methods.

Desalination process can be described in simpler terms as a process concerned with removal of salts from a liquid. Hendry (1982b:4) argues that the process of desalination of water used for production of potable water varies between two extremes: 1) water with extreme saline content, e.g., sea water which has a TDS (total dissolved solids) content of about 35 mg/l and 2) water with low saline content e.g., borehole water which has TDS content of between 1 mg/l to about 10 mg/l.

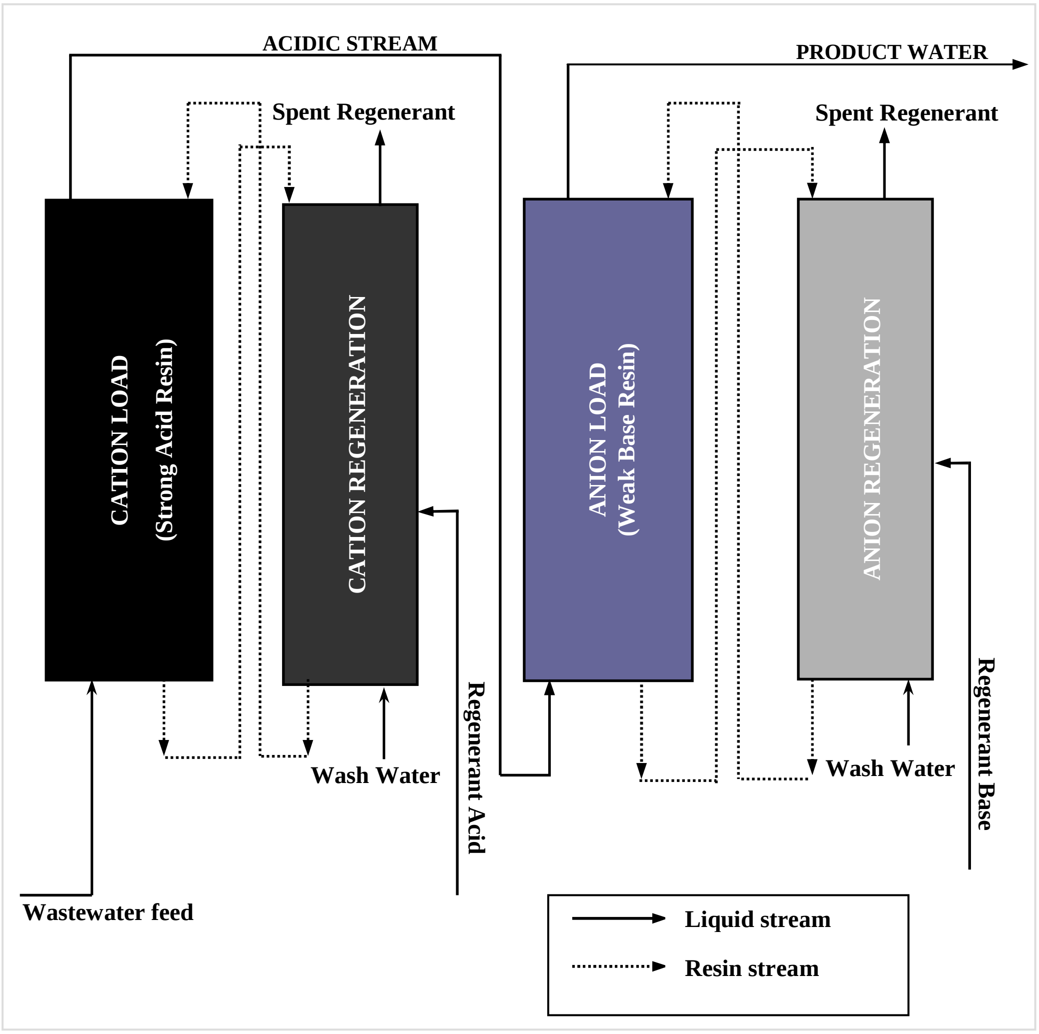


Figure 4.1. **Basic continuous countercurrent ion exchange (CCIX) configuration (Adapted from Hendry, 1982a:535, 537; Hendry 1982b,c,d)**

In Hendry (1982a–d) the project was an improvement of an earlier pilot project that produced 6 kl/day product water. Improvements of the system included a microprocessor's controlled system that used pH and conductivity (K) measurements at specific strategic points to determine product water quality. The pH and conductivity measurements were used to determine concentrations of different streams and then fed information to a computer system to make proper adjustments to the plant with the aim of obtaining optimal performance. With the new system development at Cape Peninsula University of Technology, the aim is to use modern optimal control techniques to improve the plant performance which led to the MTech study taken which was a predecessor to this current study which focuses on modelling, parameter and state estimation. The developed model is equally applicable to each phase of the CCIX process.

In Hendry (1982a, 1982c) the chemicals considered for the process of regenerarion were a combination of nitric-acid and ammonia; and sulphuric-acid and lime. Each of the chemical combinations has its merits and associated problems. In the current project, based on the findings in the UCT project, the sulphuric-acid and lime have been considered. Some of the reasons for choosing sulphuric-acid and lime regenerant system include the following (Hendry, 1982b; Hendry, 1982c; Hendry, 1982d; Hendry, 1982e; Mendow, et al., 2017):

1) these chemicals are less expensive compared to their nitric-acid and ammonia counterparts,
2) the nitric-acid and ammonia system, from the operational point of view is the easiest to deal with and has less waste disposal problems – but needs an extra column for this purpose,
3) in case of the sulphuric-acid and lime combination, both chemicals are general purpose regenerants and are readily available at a lower cost,
4) problems associated with the sulphuric-acid and lime combination are low solubility of lime as an anion regenerant and the formation of calcium sulphate scaling when sulphuric acid is used as a cation regenerant, but these problems were successfully overcome.

4.2.1. General description of an ion exchange process

According to Hendry (1982b:3), Mendow, et al., (2017), Rochette, (2006), and Treyball (1980) the accepted definition of ion exchange process is that, “an ion exchange is a reversible interchange of cations (positively charged ions) or anions (negatively charged ions) between a solid phase and a liquid phase”. In general the liquid phase is the solution being treated, and the solid phase is the resin being used for the exchange. Resins are commonly presented in a form of microporous beads that are packed together to form a bed of solid material. This ion interchange can be represented as follows (Hendry, 1982b:8; Treybal, 1980, Streat, 1980).

$$R^- - A^+ + B^+Y \longleftrightarrow R^- - B^+ + A^+Y \qquad (4.1)$$

for the cation exchange where A and B are positively charged exchanging ions (cations) and

$$R^+ - M^- + N^-Y \longleftrightarrow R^+ - N^- + M^-Y \qquad (4.2)$$

for the anion exchange where M and N are negatively charged exchanging ions (anions), where

R^+ – is the resins in positively charged form representing the solid phase in the exchange,

R^- – is the resins in negatively charged form representing the solid phase in the exchange,

Y – represents the component of the exchanging compound.

Ion exchange resins with the opposing polarities in the exchanging materials are called the *counter-ions*, e.g., A^+ and B^+ in the cation reaction, Equation (4.1) and M^- and N^- in the anion reaction, Equation (4.2). Ions that have the same polarity as that of the exchanger in a reaction are called *co-ions*, e.g., A^+ in the cation reaction, Equation (4.1) and Y in the anion reaction, Equation (4.2).

Resins are manufactured in either porous polymer spheres or in granular form. One of the most valued properties of resins is their ability to reverse exchanging ions, which allows resins to be reversed back to their original form which the resins were in before the exchange. This allows resins to be used for very long periods of time running into decades. Strong cation resins (sulphonic acid group) are only regenerated by a strong acid such as hydrochloric acid, sulphuric acid and ammonium hydroxide and in case of the weak cation resins (carboxylic acid group) resins can be regenerated by any acid that produces pH lower than the pK value of the resin (Sasan, et al., 2017).

Anion resins are rather grouped according to the structure of the nitrogen grouping of amine within the resin matrix structure; primary amine, secondary amine, tertiary amine and quaternary ammonium. The quaternary ammonium group behaves like the strong cation acid resin due to its ability to dissociate easily. The quaternary ammonium group anion resins require a strong base to regenerate back to the hydroxyl form. In the case of the weak base group (i.e., primary amine, secondary amine and tertiary amine) and one of the most important characteristics of this group is that for the for the ion exchange to occur, “the solution must have low enough pH to provide the required hydrogen ions” (Hendry, 1982c; McGarvey & Gonzalez, 1992; Sasan, et al., 2017).

4.2.2. Characteristics of ion exchange columns

Ion exchange operation can be broadly divided into two categories, 1) *the fixed bed ion exchange* or 2) *the fluidized bed ion exchange*. Both these types use columns where a solution is passed through solid particles (resins) usually a bed of beads. In the fixed type ion exchange operation, all the exchange cycles, i.e., loading, resin regeneration and resin wash are all performed in one vessel. Different forms of operations are achieved by changing the direction of flow of the feed in a vessel. In the *fixed bed type*, the feed flow is passed through a bed of resin downward thus keeping resins in a bed layer. If the feed flow direction is changed to move upwards resulting in the fluidized resin filling up the vessel, the process is a *fluidized bed type* (Hendry, 1982b:37).

The relationship between the feed and the regeneration flow directions is also used to describe the form of flow a system is using. A system where the feed direction is the same as the regeneration is known as *co-current flow* (or *co-current* load and regeneration system). A system where the feed is flowing in the opposite direction as that of regeneration is called the *countercurrent flow*. Moving beds are divided into two categories, 1) the *moving fixed beds* also known as the *moving packed beds* and 2) *the moving fluidized beds*. In the moving packed beds, the feed is done in one vessel and then resin bed is moved into another vessel for regeneration. In the fluidized moving beds, the resins are fluidized during loading and also at regeneration stage; this is explained further in the next section.

The moving beds system can be defined also by the direction of feed versus that of regeneration. If the direction of feed is the same as that of regeneration the system is said to be co-current moving bed and in the case where the feed is moving in the opposite direction to that of the regeneration, then the system is said to be countercurrent. The “moving packed beds” can either be co-current or countercurrent unlike the fluidized moving beds that are based on the countercurrent flow (Hendry, 1982b:38, 42).

The system of moving beds (packed or fluidized) has become more attractive more especially in cases where fixed beds were mostly used. The design of fluidized moving beds is based on what is generally referred to as *continuous countercurrent* flow system. Though the term continuous is used, in reality resin flow is cyclic but continuous in the sense that there is a constant flow of resin between the feed and the regeneration stage; this happens cyclically (Hendry, 1982b:42).

4.3. The fluidized bed columns system of operation

Fluidized bed systems are based on the Cloete-Streat contact design, Figure 4.1. The three phases shown below are within one column. The system has two major cycles for each of the load columns: the **loading operational** cycle which consists of *resin fluidization*, *resin settle* and then *resin pulldown* (Figure 4.2), and the **secondary operational** cycle which involves *dewatering of the resin* from the load column, *refluidisation* and resin *transportation* to the regeneration column. The system regeneration columns have **regeneration operational** cycle includes, *resin fluidization*, *resin settle* and *resin pulldown* and the **secondary operational** cycle involves only two stages, *resin refluidization* and *resin transportation* to the load column. The dewatering step is not required since the small dilution of load column product will not be detrimental to the mixing of the solution in the load column.

During the upflow period, the feed enters the bottom stage of the column and flows upward in the column fluidizing resin beds in each stage. The upflow period is determined by the required throughput based mainly on the *feed flow rate*, *column stage capacity*, the *feed concentration*, and other *operational factors*. After the required upflow period, the feed flow is stopped and the resin is allowed to settle down at the bottom of each stage which is a very short period. After the settle period the resin is then pulled out at the bottom stage of the load column for transfer to the regeneration column. Pulldown is only activated if there is enough resin (one stage amount of resin) ready to be transferred to the top stage of the load column (Hendry, 1982a; Hendry, 1982e).

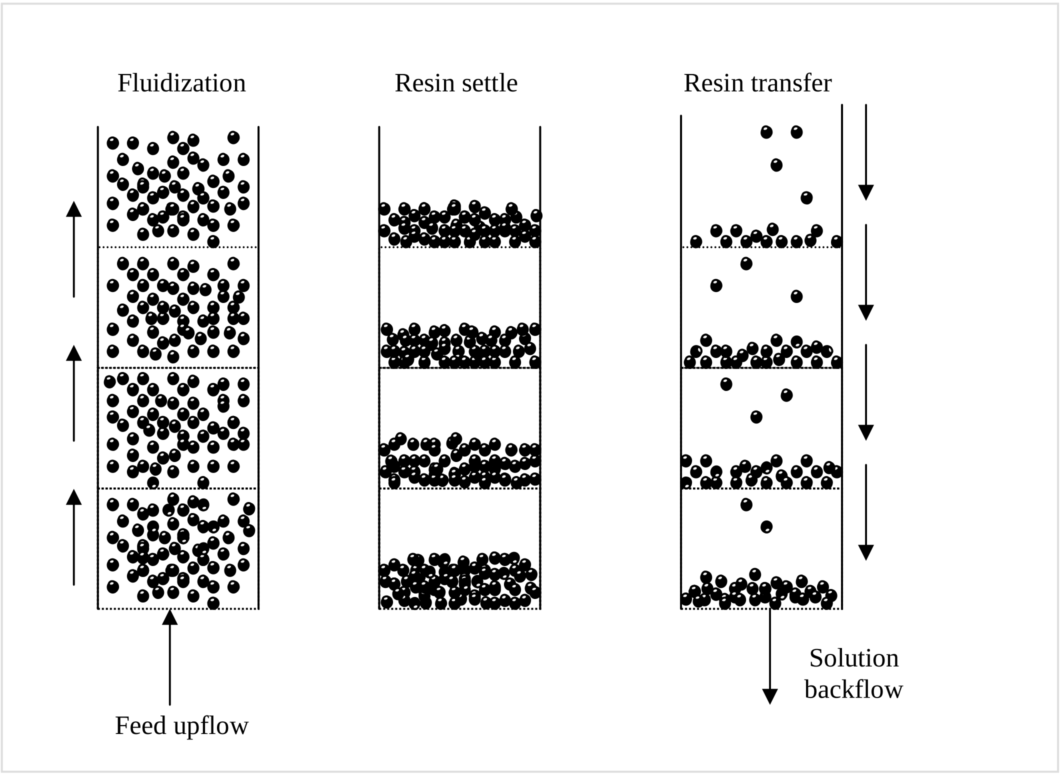


Figure 4.2. **The basic Cloete-Streat ion exchange contacting device**

(Adapted from Streat, 1995:2844)

It may be important to mention at this point that resin beds at the top end stages of a load column are less concentrated with exchanging ions from the feed compared to the beds at the bottom columns, and thus the most bottom stage is the one always taken out for regeneration. This means that the top stage of the loading column is always having the freshest resin after regenerated resin has been transfered to the loading column. Operational cycles in loading and regeneration columns may come in and out of phase during the process operation due to column sizes. There are two main coordinating events between operational cycles: 1) load pulldown can never be done if there is no enough resin available to fill the top stage of the loading column, and 2) in the secondary cycles of the load column, dewatering, refluidizing and resin transport must all occur during the load upflow period (Hendry, 1982e:9).

In the regeneration column, the three primary operational cycles identical to the ones of the loading column are also used. The only difference is that these cycle times are shorter than in the loading column since the regeneration column is smaller in diameter compared to the loading column. Due to this out of phase control of the operational cycles, it becomes very important to design a proper timing device for the sequences. The most important aspect of the operation at this point is to make sure that there is enough regenerated resin available before load column pulldown is activated (Hendry, 1982d:8, 9).

4.3.1. Cation and anion loading

In the cation load column natural salts are split into their corresponding strong acids using strong cation resin in the hydrogen form (Hendry, 1982c:10; Hendry, 1982; Randal, 1984):

$$R-H+NaCl\longleftrightarrow R-Na+HCl \tag{4.3}$$

$$2R-H+MgSO_4\longleftrightarrow R_2-Mg+H_2SO_4 \tag{4.4}$$

$$2R-H+CaCl_2\longleftrightarrow R_2-Ca+2HCl \tag{4.5}$$

where

$R-H$ – is the strong cation resin in hydrogen form,

$R-Na$ – is the sodium ions attached to resin after the ion exchange,

$NaCl$ – is sodium chloride,

HCl – is hydrochloric acid,

$MgSO_4$ – is magnesium sulphate,

$R_2 - Mg$ – is the magnesium ions attached to resin after the ion exchange,

$R_2 - Ca$ – is resin with calcium ions attached after the exchange of ions

H_2SO_4 – is the sulphuric acid and

$CaCl_2$ – is calcium chloride.

Acids formed in the cation loading column are then fed into the anion loading column where they are absorbed by the weak anion resin in the free base form. According to Hendry (1982c:12) anion resin in its free base form is uncharged and protonation must first occur to be able to attract the anions, Equations (4.6 and 4.7).

$$HCl \longleftrightarrow H^+ + Cl^- \rightarrow R' \equiv NHCl \tag{4.6}$$

$$R' - N + H^* \longleftrightarrow R' \equiv N^+H \tag{4.7}$$

where

$H^+ + Cl^-$ – is the protonation of hydrochloric acid during the exchange

H^* – is hydrogen ions

$R' - N$ – is the weak anion resin in nitrogen form,

$R \equiv' NHCl$ – is the hydrochloric acid attached to the nitrate resin

Once the protonation has occurred the entire acid molecule is then taken up by the resin, releasing ‘clean water’ molecule:

$$R' - N + HCl \longleftrightarrow R'N - H^+ + Cl^- \rightarrow R'N + HCl \tag{4.8}$$

4.3.2. Cation and anion regeneration

In the cation regeneration column, the resin that is now loaded with the counter-ions is contacted with sulphuric acid as a regenerant chemical to release all the exchanged ions and reverse the resin to its original form:

$$2R - Na + H_2SO_4 \longleftrightarrow 2R - H + Na_2 \tag{4.9}$$

$$R_2 - Ca + H_2SO_4 \longleftrightarrow 2R - H + CaSO_4 \tag{4.10}$$

$$R_2 - Mg + H_2SO_4 \longleftrightarrow 2R - H + MgSO_4 \tag{4.11}$$

where

Na_2 – is two sodium ions as a compound, and

$CaSO_4$ – is the calcium sulphate,

Regenerated cation resin coming out of the cation regeneration column needs to be washed with decationized water (feedwater may also be used for this purpose), this is to remove sulphuric acid from the resin pores, after which the resin will be ready for use in the loading column.

Hendry (1982c:14) states that for anion regeneration, loaded resin from the anion load column is regenerated using a slurry of lime,

$$2R \equiv NHCl + Ca(OH)_2 \longrightarrow 2R - N + CaCl_2 + H_2 0 \qquad (4.12).$$

where

$Ca(OH)_2$ – is the calcium hydroxide and

H_2O – is water.

The slurry of lime dissolves in the process as the reaction proceeds. The protonation step that occurs during loading is then reversed. During anion regeneration, the resulting calcium salts from the organic acids stripped from the resin form an insoluble flocculated sludge with abundant calcium ions and free lime, Equation (4.12).

Hendry (1982d:42) report states calcium is the most difficult ion to remove in the regeneration of the anions loaded resin; and the ionic fraction of calcium remaining in the regenerated resin becomes the determining factor of the resin effective working capacity. For this reason, the design of regeneration column may be reduced to calculating the number of stages required to fully extract calcium from anion loaded resin using the calcium-hydrogen equilibrium relationship. Generally it is acceptable to have more stages than calculated to offset any inaccuracies that may have occurred in the calculation.

For resin regeneration in both columns, resin is pulled down with a solution from the loading columns, feed water in the cation load and weak acid stream in the anion load. These streams must be separated from the resin before it is introduced into the top stages of the regeneration columns to prevent diluting the waste solutions and feedwater wastage. The regenerated anion resin may be rinsed with desalinated (product) water to remove electrolytes from the resin pores and to disengage lime and organic sludge from the voids. After rinsing off the electrolytes, the anion resin will be ready for use in the next anion loading cycle (Hendry, 1982d:14).

4.4. Bilinear model development for the continuous countercurrent ion exchange (CCIX) process

Sundstrom and Klei (1979:301) state that it is possible to apply thermodynamic concepts in natural water systems. These concepts allow in part or full application of thermodynamic equilibrium. Applying thermodynamic principles allows the determination of composition of the system at equilibrium (Sundstrom

& Klei, 1979:301). According to Treybal (1980:646), the controlling factor in an ion exchange reaction may be kinetics of the reaction or the rate of diffusion. In cases where the kinetics are faster than the rate of diffusion, the kinetics of the reaction become the controlling factor. During the process of exchanging ions, rate of transfer of ions can be defined by the following processes that take place (Foust, et al, 1980:408; Treybal, 1980:643):

1) diffusion of ions from the bulk liquid to the surface of the resin particle,
2) diffusion of ions from the surface of the resin into the site of exchange,
3) exchange of the ions at the active site,
4) diffusion of the replaced ions from the resin active site to the surface of the particle and,
5) the diffusion of the released ions from the resin surface to the bulk liquid.

It is on these bases that compositions in the ion exchange process are used for developing the proposed bilinear model of the Continuous Countercurrent Ion Exchange (CCIX) process. The focus is on development of a model based on the exchanging ions that control the *mass transfer rate*, i.e., sodium and hydrogen cations. According to Sundstrom & Klei (1979:301), when applying equilibrium analysis, *rates of reactions between species* should be faster than time scale of the process. Generally, in aqueous systems, thermodynamic analysis is commonly used with inorganic reactions due to the fact that inorganic reactions that involve ionic mechanism tend to be faster in nature. On the other hand organic reactions are mostly associated with radical or biochemical mechanisms and tend to be slower than ionic reactions.

Sundstrom and Klei (1979:302) argues that “all important chemical species present in the system should be included in an equilibrium model” but considerations should be given to the limitations that may arise from analytical complexities. According to Sundstrom and Klei (1979:302) it is possible for the system to reach a steady state if all flows and concentrations that enter or leave the system are not varying with time. Treybal (1980:19) suggests that the rate at which each component in the ion exchange is transferred from one phase to another depends on *the mass transfer rate* or *the mass transfer coefficien*t and also upon *the degree of departure from the equilibrium*. Mass transfer will stop when equilibrium is reached.

As the transfer of solute from one phase to another continues, the concentration within each phase also changes throughout the contacting device. Coulson, et al., (1996:526) state that mass transfer processes that involve two fluid streams are usually carried out using continuous countercurrent or co-current flows in a column equipment. According to Treybal (1980:644) operational techniques used for adsorption processes are also used for the ion exchange process, these include; batch or *stagewise treatments*, *fluidised* or *fixed beds* operations and *countercurrent* or *co-current* operations and *continuous* operations (Treybal, 1980; Coulson et al, 1996:526).

According to Coulson et al., (1996:26) material balance can be described in terms of volume or time. In cases where compositions vary with position in the column, differential element of volume must be used and in cases where compositions are constant, a unit volume must be used, for example, in a case of a well-mixed batch reactor or a continuous stirred tank. Given all the arguments above, material balance equation may be generalized as follows (Coulson et al., 1996:26):

$$\begin{bmatrix} rate\ of\ \ flow \\ of\ \ reactant \\ into\ volume \\ element \end{bmatrix} - \begin{bmatrix} rate\ of\ \ flow \\ of\ \ reactant \\ out\ of\ \ volume \\ element \end{bmatrix} - \begin{bmatrix} rate\ of\ \ reactant \\ removal \\ by\ reaction\ \ within \\ volume\ element \end{bmatrix} = \tag{4.13}$$

$$= \begin{bmatrix} rate\ of\ \ accumulation \\ of\ \ reactant\ within \\ volume\ element \end{bmatrix}$$

The proposed ion exchange *bilinear model* can be formulated based on mass transfer between the exchanging ions in the ion exchange column for each stage. The development of the model is based on the *law of conservation of mass*, the mass balance and component balance. According to the principle of conservation of mass when applied to dynamic systems; the *mass balance* (*total continuity*) equation is given by:

$$\begin{bmatrix} mass\ \ flow \\ into\ the\ system \end{bmatrix} - \begin{bmatrix} mass\ \ flow \\ out\ of\ \ the\ system \end{bmatrix} = \begin{bmatrix} time\ \ rate\ \ of\ \ mass \\ change\ \ inside\ the\ system \end{bmatrix} \tag{4.14}$$

The units for the mass balance equation in Equation (4.13) given above are $[mass\ \ per\ time]$.

Fogler (1991:543) describes the mass transfer as being concerned with the process of diffusion where diffusion could be a *spontaneous intermixing of atoms* or *molecules due to thermal heat*. In mass transfer problems diffusion is usually divided into four types: 1) *equimolar counter-diffusion*, 2) *dilute concentration*, 3) *diffusion through* a stagnant gas and 4) *forced convection* (Fogler, 1991:548).

Unlike mass or energy balances, chemical components are not conserved. If a reaction occurs inside a system, the number of moles of an individual component will increase if it is a product of the reaction or decrease if it is a reactant. The *component continuity equation* (*component balance*) for the i^{th} chemical species of a system is given by Equation (4.15). This equation is given in units of $[moles\ of\ component\ i\ per\ unit\ time]$ (Fogler, 1991:6):

$$\begin{bmatrix} flow\ of\ moles \\ of\ i^{th}\ component \\ \text{into}\ the\ system \end{bmatrix} - \begin{bmatrix} flow\ of\ moles \\ of\ i^{th}\ component \\ out\ of\ the\ system \end{bmatrix} + \\ + \begin{bmatrix} rate\ of\ formation\ of\ moles \\ of\ i^{th}\ component \\ from\ the\ chemical\ reactions \end{bmatrix} = \begin{bmatrix} time\ rate\ of\ change\ of\ moles \\ of\ i^{th}\ component \\ inside\ the\ system \end{bmatrix} \tag{4.15}$$

4.4.1. Mass transfer analysis for the continuous countercurrent ion exchange (CCIX)

The flows in and out of the system can be both *convective due to bulk flow* and *molecular due to diff*usion. The ion exchange process theoretical models are based on macroscopic descriptions using lumped variables.

In developing the considered model the following assumptions are made (Dube, 2002:137; Hendry, 1982):

1) Both the volume of every stage and the amount of resin holdup and liquid holdup in each stage are equal just before the transfer; and equal volumes of resin are transferred between stages,
2) The transfer of resin between states is instantaneous, and neither ion exchange reaction nor adsorption takes place during this period. This implies that hydrodynamic delays in liquid and resin streams are neglected,
3) The resin particles are uniform, both in size and density at all conversion levels, so that segregation does not occur,
4) The fluidized phase is perfectly mixed in each stage and the expanded fluidized bed fills the entire stage volume, i.e., concentration of material in the stage is the same throughout each stage.
5) Resin back mixing does not occur,
6) Operation of the column is at steady state, i.e., electroneutrality is maintained.

7) There exists a linear equilibrium between the liquid and the resin phases.

The developed mathematical model has to predict the liquid and resin compositions in each stage for every cycle after the step change. The calculations are based on *equilibrium and kinetic data* and *resin and liquid flow rates*. These calculations are done stagewise starting from the bottom stage, which is considered the first stage, to the top stage which is considered the last stage. Changes in the lower stages will affect the stages above.

The general formulation for the material balances are considered for any stage ($stage\ n$) using the loading column. Based on the exchanging ions within each stage, Figure 4.3, the following formulation is considered:

$F_L(t)$ is the molar flow rate of the solution (liquid), $F_R(t)$ is the molar flow rate of the resin (solid), $x_n(t)$ is the mole fraction of the material in the liquid phase going out of stage n, $x_{n-1}(t)$ is the mole fraction of material in the liquid phase of stage $n-1$ going to stage n, $y_n(t)$ is the mole fraction of the material in resin phase going out of stage n, $y_{n+1}(t)$ is the mole fraction of the material in resin phase going into stage n from the stage $n+1$.

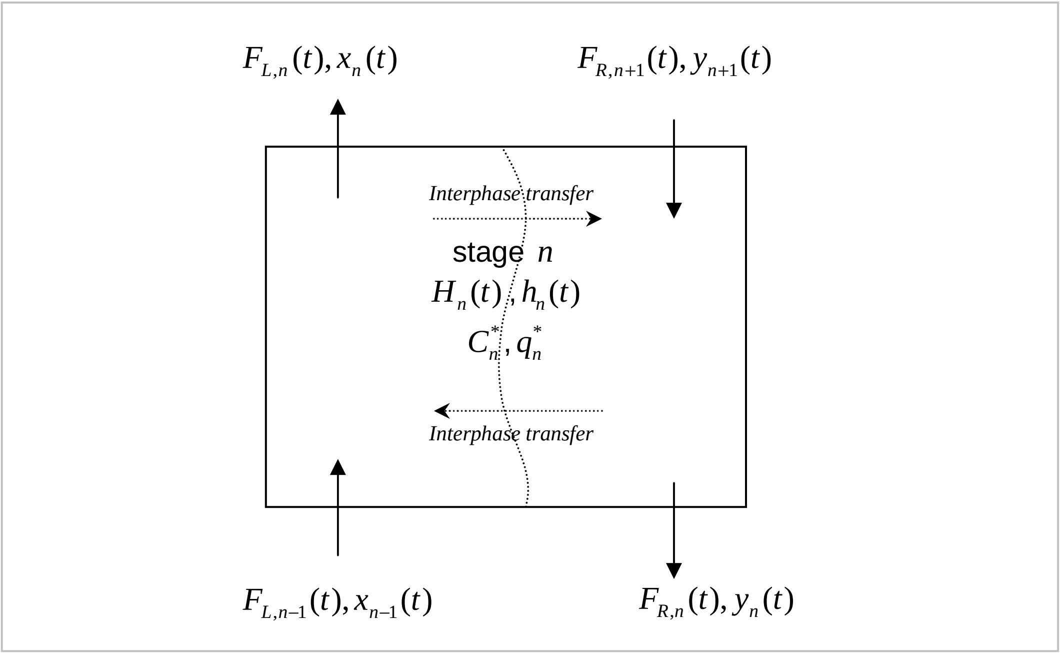


Figure 4.3. **Mass transfer in each stage ion of the exchange process (Adapted from Hendry, 1982b)**

$H_n(t)$ is the liquid holdup and $h_n(t)$ is the resin holdup in the n stage, C_n^* is the total concentration in solution of all exchanging materials, and q_n^* is the total concentration in resin of all exchanging materials. Each phase within the stage is considered uniformly mixed separate subsystem with interphase mass transfer between the two phases, liquid and solid, Figure 4.3.

4.4.2. Component balance used to develop the CCIX model

The ion exchange model is obtained based on the i^{th} component mass balance on stage n. The component mass law of material balance based on the rate of accumulation and rate of materials formation, Equation (4.14) can be expressed as (Dube, 2001):

$$\begin{aligned}\frac{d(H_n(t)x_n(t))}{dt}+\frac{d(h_n(t)y_n(t))}{dt}&= \\ =\left[F_{L,n-1}(t)\cdot x_{n-1}(t)+F_{R,n+1}(t)\cdot y_{n+1}(t)\right]&- \\ -\left[F_{L,n}(t)\cdot x_n(t)+F_{R,n}(t)\cdot y_n(t)\right]&\end{aligned}\quad,\ n=\overline{1,N} \qquad (4.16)$$

where

$F_{L,n-1}(t)\cdot x_{n-1}(t)$ – is the rate of material input with the liquid coming from the plate $n-1$,

$F_{R,n+1}(t)\cdot y_{n+1}(t)$ – is the rate of material input with the resin coming from the plate $n+1$,

$F_{L,n}(t)\cdot x_n(t)$ – is the rate of material output with the liquid leaving the plate n for plate $n+1$,

$F_{R,n}(t)\cdot y_n(t)$ – is the rate of material input with the resin leaving the plate n for $n-1$,

$\frac{d(H_n(t)x_n(t))}{dt}$ – is the rate of accumulation of species i in the liquid phase on the plate n,

$\frac{d(h_n(t)y_n(t))}{dt}$ – is the rate of accumulation of species i in the resin phase on the plate n, and

N – is the total number of stages and numbered from the bottom stage to the top one.

The total mass balance equation for the stage n can be written as

$$\begin{aligned}H_n\dot{x}_n(t)+h\dot{y}_n(t)&= \\ =F_{L,n-1}x_{n-1}(t)-F_{L,n}x_n(t)+F_{R,n+1}y_{n+1}(t)-F_{R,n}x_n(t)&\end{aligned},\ n=\overline{1,N} \qquad (4.17)$$

Based on assumptions made, holdups and flow rates are considered constants, i.e.,

$H_n(t) = H$, $h_n(t) = h$, $F_{R,n}(t) = F_R$ and $F_{L,n}(t) = F_L$ (4.18)

The equilibrium in the stage n, $n = 1, N$ assumed to be linear to maintain electroneutrality, and the relationship between exchanging cations is given by:

$$y_n(t) = a_n x_n(t) + b_n \tag{4.19}$$

where

a_n – is the slope of the pseudo equilibrium curve,

b_n – the corresponding intercept.

Equation (4.19) describes the linear relationship between the liquid and the resin at equilibrium. After substituting the conditions for the same holdups and flow rate in Equation (4.17), the component balance equation becomes,

$$H\dot{x}_n(t) + h\dot{y}_n(t) = F_L(x_{n-1}(t) - x_n(t)) + F_R(y_{n+1}(t) - y_n(t)),\, n = \overline{1, N} \tag{4.20}$$

and then substituting for the linear relationship between exchanging components Equation (4.19) in Equation (4.20), the component mass balance becomes

$$\begin{aligned} H\dot{x}_n(t) + a_n h\dot{x}_n(t) = F_L\left[x_{n-1}(t) - x_n(t)\right] + \\ + F_R\left[(a_{n+1}x_{n+1}(t) + b_{n+1}) - (a_n x_n(t) + b_n)\right] \end{aligned},\; n = \overline{1, N} \tag{4.21}$$

and finally,

$$\begin{aligned} \dot{x}_n(t) = \frac{F_L}{(H + a_n h)} x_{n-1}(t) - \frac{(F_L + a_n F_R)}{(H + a_n h)} x_n(t) + \\ + \frac{a_{n+1}F_R}{(H + a_n h)} x_{n+1}(t) + \frac{b_{n+1} - b_n}{(H + a_n h)} F_R(t) \end{aligned},\; n = \overline{1, N} \tag{4.22}$$

The variable F_R is considered the control variable and $x_n(t)$, $n = \overline{1, N}$ in Equation (4.22) are the state variables. The model has parameters that are linear and variables that are nonlinear.

The control variable F_R is a constant variable according to the design of the CCIX process and cannot be used for dynamic optimization. This problem is solved by proposing representation of resin flow rate as a variable with the same value for every stage but with values that are changing with time.

$$F_{R,n}(t) = F_R(t) \tag{4.23}$$

Then the Equation (4.23) can be rewritten in the following way:

$$\begin{aligned} \dot{x}(t) = \frac{F_L}{(H + a_n h)} x_{n-1}(t) - \frac{F_L}{(H + a_n h)} x_n(t) - \frac{a_n}{(H + a_n h)} x_n(t) F_R(t) + \\ + \frac{a_{n+1}}{(H + a_n h)} x_{n+1}(t) F_R(t) + \frac{b_{n+1} - b_n}{(H + a_n h)} F_R(t) \end{aligned},\, n = \overline{1, N}$$

(4.24).

Equation (4.24) can be used for optimal state and parameter estimation. For the total number of N stages, the model can be expressed using the state space representation. From Equation (4.24) above, for $n=1$, the variable $x_{n-1}(t)=x_f(t)$ is considered to be the input feed concentration at the first stage of the column which changes according to the effluent content. This is therefore considered the disturbance to the process $w(t)$, and thus,

$$x_f(t)=w(t) \qquad (4.25).$$

The last (top) stage liquid concentration leaving the load column is considered the output variable, i.e., $z(t)=x_N(t)$. Other conclusions that can be derived from the model equation are that; $x_{N+1}(t)=0$ since there is no stage after the N th stage and the state space model can be written as:

$$\dot{x}_n(t)=\begin{bmatrix} \frac{-F_L}{H+a_1h} & 0 & 0 & \cdots & 0 \\ \frac{F_L}{H+a_2h} & \frac{-F_L}{H+a_2h} & 0 & \cdots & 0 \\ 0 & \cdots & \ddots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \frac{F_L}{H+a_{N-1}h} & \frac{-F_L}{H+a_{N-1}h} & 0 \\ 0 & 0 & \cdots & \frac{F_L}{H+a_Nh} & \frac{-F_L}{H+a_Nh} \end{bmatrix} \times \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \\ x_6(t) \end{bmatrix}$$

$$+\begin{bmatrix} \frac{-a_1}{H+a_1h} & \frac{a_2}{H+a_1h} & 0 & 0 & \cdots & 0 \\ 0 & \frac{-a_2}{H+a_2h} & \frac{a_3}{H+a_2h} & 0 & \cdots & 0 \\ 0 & \cdots & \ddots & 0 & \cdots & 0 \\ \cdots & \cdots & \ddots & \ddots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{-a_{N-1}}{H+a_{N-1}h} & \frac{a_N}{H+a_{N-1}h} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \frac{-a_N}{H+a_Nh} \end{bmatrix} \times$$

$$\times\left[x_1(t) \quad x_2(t) \quad \ldots \quad x_n(t) \quad \ldots \quad x_N(t)\right]^T F_R(t)+$$

$$+\left[\frac{(b_2-b_1)}{H+a_1h} \quad \frac{(b_3-b_2)}{H+a_2h} \quad \ldots \quad \frac{(b_N-b_{N-1})}{H+a_{N-1}h} \quad \frac{-b_N}{H+a_Nh}\right]^T F_R(t)+$$

$$+\left[\frac{-F_L}{H+a_1h} \quad 0 \quad \ldots \quad 0 \quad 0\right]^T x_f(t) \qquad (4.26)$$

The output of the process is defined by the following equation, Equation (4.27) and it is also used for development of the parameter estimation methods, covered later in the thesis. The state space bilinear model, Equation (4.26) is used for estimation of its unknown parameters and for estimation of the process none measurable states.

$$z(t) = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_1(t) & x_2(t) & \dots & x_n(t) & \dots & x_N(t) \end{bmatrix}^T \tag{4.27}$$

The final state space representation of the model using state space matrices can be expressed by (Dube, 2002:146-148; Dube & Tzoneva, 2004:480):

$$\dot{x}(t) = Ax(t) + B_1 x(t) F_R(t) + BF_R(t) + Wx_f(t)\ ,\ x(0) = x_0 \tag{4.28}$$

$$z(t) = Cx(t)$$

where

$A \in R^{n\times n}$, $B_1 \in R^{n\times n}$, $B \in R^{n\times l}$, $W \in R^{n\times l}$ and $C \in R^{l\times n}$ – are the model matrices for the state, input, disturbance and output variables respectively,

$F_R(t) \in R^l$ – is the control input,

$x(t) \in R^{n\times l}$ – is the state vector,

$w(t) \in R^l$ – is the input disturbance vector

$x(0) = x_0$ – represents the system state initial condition and,

$z(t)$ – is the output of the model.

$$A = \begin{bmatrix} \frac{-F_L}{H + a_1 h} & 0 & 0 & \cdots & 0 \\ \frac{F_L}{H + a_2 h} & \frac{-F_L}{H + a_2 h} & 0 & \cdots & 0 \\ 0 & \cdots & \ddots & \cdots & 0 \\ \cdots & \cdots & \ddots & \ddots & \cdots \\ 0 & \cdots & \frac{F_L}{H + a_{N-1} h} & \frac{-F_L}{H + a_{N-1} h} & 0 \\ 0 & 0 & \cdots & \frac{F_L}{H + a_N h} & \frac{-F_L}{H + a_N h} \end{bmatrix}$$

$$B_1 = \begin{bmatrix} \frac{(b_2 - b_1)}{H + a_1 h} & \frac{(b_3 - b_2)}{H + a_2 h} & \dots & \frac{(b_N - b_{N-1})}{H + a_{N-1} h} & \frac{-b_N}{H + a_N h} \end{bmatrix}^T$$

$$W = \begin{bmatrix} \frac{F_L}{H + a_1 h} & 0 & \dots & 0 & 0 \end{bmatrix}^T$$ and $$C = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

$$B=\begin{bmatrix} \frac{-a_1}{H+a_1h} & \frac{a_2}{H+a_1h} & 0 & 0 & \cdots & 0 \\ 0 & \frac{-a_2}{H+a_2h} & \frac{a_3}{H+a_2h} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & \ddots & \cdots & 0 \\ \cdots & \cdots & \cdots & \ddots & \ddots & \cdots \\ 0 & 0 & \cdots & \frac{a_{N-1}}{H+a_{N-1}h} & \frac{a_N}{H+a_{N-1}h} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \frac{-a_N}{H+a_Nh} \end{bmatrix}$$

The model, Equation (4.28) is a bilinear one as it contains an expression of a product between the components of the state and control vectors. The state space bilinear model is used further for estimation of its unknown parameters and for estimation of it is none measurable states.

4.5. Conclusion

This chapter deals with the understanding of the ion exchange process used for the proof of the hypothesis for developing new techniques for modelling, parameter and state estimation of a bilinear system. The full description of the ion exchange process has been given and the method used to develop the bilinear model for the process has been discussed. The developed model is based on the continuous countercurrent ion exchange (CCIX) process as previously built at the University of Cape Town Chemical Engineering Department.

The ion exchange process is a process that involves ionic interaction in an electrolyte solution with a solid material as an adsorbant. The process involves counter exchange of dissociated ions of the same polarity between the solute and the adsorbant. Such exchange is driven by mass transfer within the system. This mass transfer continues until some equilibrium is reached, though in the case of ion exchange no 100% equilibrium is reached. As the transfer of solute from one phase to another occurs, concentration within the phase also changes in the system (Streat, 1995:2841; Treybal, 1980:19, 117, 256)

The plant considered is operated in a continuous fashion but periodical, which means that there is a continuous feed supplied and continuous resin regeneration. These two distinct operations are such that they occur periodically but simultaneously without having to stop the one operation to get to the other. Though, the system is said to be continuous, in reality it is periodical, in the sense that there are two distinct cycles of operation, the primary cycle and the secondary cycle. The primary cycle is for both columns of each phase, where there are three main cycles, the upflow period, resin settle period and the resin pulldown period. The secondary cycle is mostly for resin transfer from the regeneration column back to the loading column. The secondary cycle is considered for the load column only; it includes, resin dewatering, resin fluidisation and resin transportation.

It has been suggested by a few authors that the rate at which each component is transferred from one phase to another depends on the mass transfer rate. It is on basis of this mass transfer using the continuity equation that a model for the given CCIX has been developed. The developed model is a bilinear model, with parameters entering the model linearly and variables nonlinearly. From the developed model, unknown parameters have to be estimated and the system states estimated since it is impossible to measure these variables in the plant. In this case the system states represent the concentration in each and every stage of the column. Once values of the unknown parameters and not measurable states are estimated the process model may be used for real-time optimal control.

In the next chapter, the data used for the continuous countercurrent Ion exchange process model development and reformulation for parameter and state estimation problems are presented. The chapter describes the procedure for validating the performance of the developed methods for parameter and state estimation with the possibility of using the model for optimal control of the CCIX plant. Simulation results are presented for the state trajectories.

CHAPTER FIVE

DATA AND MODEL REFORMULATION FOR ESTIMATION PROBLEMS

CHAPTER FIVE

5. DATA AND MODEL REFORMULATION FOR ESTIMATION PROBLEMS

5.1. Introduction

Measurement data received from the project of wastewater treatment plant produced by the University of Cape Town, Chemical Engineering Department were used to build a case study to validate the performance of the developed in the thesis methods for parameter and state estimation. Data used were collected from the continuous countercurrent ion exchange (CCIX) process under steady state operating conditions. The model developed also considers disturbances to the steady state operating conditions, such as resulting from any change in feed concentration depending on the effluent level in the feed water. The data presented consider the *change in feed concentration* to be a disturbance input to the process.

According to Hendry (1982d:80) report: “typical domestic sewage effluent undergoes concentration fluctuations of about 20% to 30% during a 24 hour cycle whereas industrial effluents may vary by several hundred percent during the same period.” Fluctuations in concentration necessitate a proper control action, if the plant experiences greater concentration fluctuations in the feed, it becomes very important to control this change effectively so as to keep the cost within minimum stipulated requirements. If this condition is not met, the implemented control system cannot be considered effective.

Optimization of the CCIX process requires that the control system’s operating conditions should be such that the required quality of product water is at a constant throughput with the minimum usage of regenerant chemicals. It is on this basis that the control strategy for the process is developed and specific data measurements are needed for this purpose. It is therefore important that prior to implementation of any control strategy, reliable data is acquired and be available in a convenient form in a computer system to be used. Data (the salt concentration measurements) are collected at specific points in the plant for use in the overall optimization of the control system of the process. Salt concentration is used as the main ionic fraction that determines the process performance. This is estimated from the measurement of conductivity. The error due to the presence of other ions in the resin is negligible.

Data presented are the ionic fraction of the two main ion species involved in the exchanging process, viz. the sodium, Na^+ and hydrogen H^+ ions. For the parameter and state estimation model presented here, data are presented using the Na^{+} ionic fraction. According to Hendry (1982d:73), errors that may occur in determining concentration using this procedure due to other unknown cations present in the solution are minimal as compared to errors due to unknown anions present in the solution which may result in up to 10% errors. Another observation from the report was that organics have negligible contribution towards the conductivity of the sewage effluent used as inflow for the ion exchange process (Hendry, 1982d:75).

Given that throughout the column from the bottom stages, the concentration of counter ions are decreasing from one stage to the next and increasing in each stage throughout the number of cycles to reach steady state; from this observation it was also concluded that this data can never be that of 1) *sodium in liquid* ($x_{Na^{+}}$) or hydrogen in resin ($y_{H^{+}}$). The data were identified to be that of either 1) *sodium in the resin* ($y_{Na^{+}}$) or 2) *hydrogen in the liquid* ($x_{H^{+}}$). The state trajectories are generated using MATLAB software program based on this data; simulation results of these trajectories are presented later in this chapter.

The chapter is structured as follows: measurement data considerations; experiment methods and practical considerations for use of previous data from Hendry 1982; experimental data and results used in the thesis; model reformulation from the model developed in Dube (2001) for parameter estimation problem. Two cases of parameter estimation are considered, the linear and nonlinear parameter estimation; then, parameters selection and formulation of parameter estimation problem, and the concluding remarks.

5.2. Measurement data considerations

In the Hendry (1982d:75) report, it is mentioned that for a control system designed to maintain constant ionic composition in the product water, the conductivity of the feed solution, and the pH and conductivity of the product will be the most useful parameters. The feed solution conductivity could be used accurately to determine the total salt concentration and in turn be used to determine the nominal cycle times for the load columns (Hendry, 1982d).

The liquid concentration can be considered to be describing the process's dynamic behaviour; it is selected as the state of the process in the model, and thus used for optimization of the process based on state space analysis. Based on the process plant design it is impossible to measure concentration in each stage. It is therefore important to find means of estimating concentration values in each stage of the loading columns. State estimation techniques are thus used to determine the state trajectories in each stage.

Data are acquired using pH and conductivity measurements at the inflow and outputs of the process columns. pH and conductivity (K) measurements are then used to calculate the corresponding salt concentration values of the different streams. These data are presented as inflow-output measurements of the process. From the inflow-output measurement data, concentration for every stage is estimated using the state estimation techniques and required parameter values are determined. Based on the calculated data, the model parameter values, the state values, and the feed concentration disturbances, further control actions required for the full control of the plant can be determined.

The plant has to be able to respond correctly according to the changes in the feed concentration. What happens when the feed concentration goes up, what happens when it goes down and for how long must the change in the concentration be presented by measurements before the control action responds to the change. All these questions are addressed in the next section based on data collected from Hendry (1982a–e).

5.3. Experiment methods and practical considerations

The presented data are based on experiments that were conducted for the 6 stage column system and the 12 stage column system. In both cases a step change in feed concentration was introduced, firstly by moving from low to high and then another one by moving from high to low. The ion exchange columns used were $50mm$ inside diameter and were with $500mm$ in height. Each system had a pair of columns one for loading the resin and the other for regeneration of the loaded resin using sulphuric acid. Both the load and regeneration columns of each pair were of the same measurements $50mm$ inside diameter and $500mm$ high.

Two tanks of feed solution were prepared with enough solutions to run the experiments for each system. The solutions were $0.05N$ and $0.10N$ concentration of sodium chloride salt ($NaCl$). Data collected are presented in Tables 5.1.–5.12. These data sets show the concentration in fractional change of H^+ as determined from the equation,

$$FC_n = \frac{I.F.H_n^+ - I.F.H_{initial}^+}{I.F.H_{final}^+ - I.F._{initial}^+} \qquad (5.1)$$

where

FC_n – is the fractional change of n th cycle from the change of the inflow concentration,

$I.F.H^{+}_{n}$ – is the ion fraction concentration of H^{+} ions of the current measurement at the n^{th} cycle from the inflow concentration step change moment,

$I.F.H^{+}_{initial}$ – is the fractional concentration of H^{+} ions at the initial moment of inflow concentration change,

$I.F.H^{+}_{final}$ – is the fractional concentration of H^{+} ions at the final moment of the experiment.

The measurements include data from the start of the experiment for the time before introducing the step change. This technique of data collection can be represented as follows (Figure 5.1). After the determination of each H^{+} value, the values are then transformed to start at zero level after the introduction of the step change as indicated by the H^{+}_{n} dotted lines in Figure 5.1. using the equation:

$$FC_nH^{+} = \left[\frac{I.F.H^{+}_{n} - I.F.H^{+}_{initial}}{I.F.H^{+}_{final} - I.F.\cdot^{+}_{initial}}\right] - 1 \qquad (5.2)$$

where FC_nH^{+} – are the H^{+} fractional change values transformed to zero-level

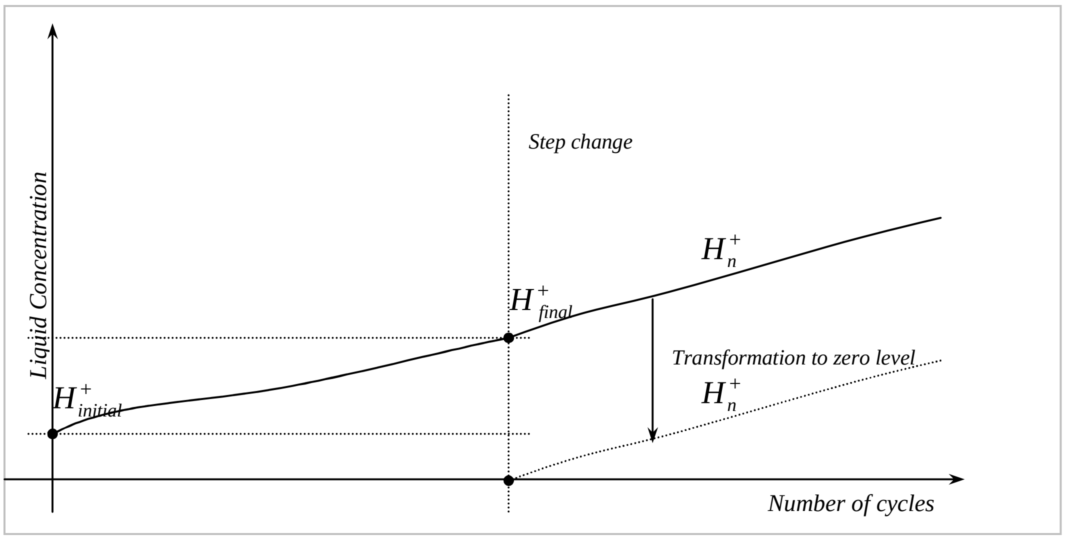


Figure 5.1. **Procedure for determining resin/liquid component concentration (Adapted from Hendry, 1982d)**

The following H^{+} values have been obtained using the above mentioned method and attached to the operating conditions for each four runs. Each experiment results are showed below with the accompanying data for the columns and the operating conditions.

5.3.1. The 6 stage system data

$Run\,1$ considers the concentration step change from LOW to HIGH in the feed concentration in the 6 stage system. The operating conditions and data received from this system are presented in Table 5.1 and Table 5.2. The operational cycles and state trajectories are presented in Figure 5.2:

Table 5.1: **6 stage system LOW-to-HIGH step change in salt concentration**

Operating conditions	
Number of stages	6
Resin volume	590 ml
Feed flow rate	750 $ml/\min$
Upflow time	15 $\min$
Feed concentration before step change (start)	41.3 meq/l
Feed concentration at step change (final)	96.2 meq/l
R/L ratio at start before step change	1.78

Table 5.2: **Run 1 H^+ fractional change values for the 6 stage system showing LOW–to–HIGH concentration step change as determined from Equation (5.2)**

Concentration as determined by fractional change FC_nH^+						
Stage No. / Cycle No.	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
1	0.221	0.000	0.000	0.000	0.000	0.000
2	0.577	0.140	0.004	0.000	0.000	0.000
3	0.730	0.314	0.066	0.004	0.000	0.000
4	0.847	0.523	0.184	0.035	0.003	0.000
5	0.920	0.656	0.295	0.082	0.020	0.000
6	0.936	0.766	0.454	0.168	0.052	0.001
7	0.968	0.842	0.601	0.277	0.113	0.024
8	0.969	0.886	0.690	0.361	0.167	0.024
9	0.974	0.900	0.758	0.440	0.207	0.063
10	0.981	0.933	0.804	0.522	0.340	0.124
11	0.989	0.958	0.877	0.698	0.474	0.167
12	0.988	0.963	0.881	0.784	0.547	0.233
13	0.997	0.982	0.951	0.899	0.780	0.482
14	1.000	0.974	0.965	0.931	0.860	0.539
15	1.000	0.991	0.972	0.966	0.899	0.672
16	1.000	0.993	0.981	0.966	0.905	0.779
17	1.000	0.994	1.00	0.991	0.975	0.940
18	1.000	0.993	0.988	0.991	0.973	0.972

$Run\,2$ considers the 6 stage system with concentration moving from a HIGH to LOW. The operational and obtained data are presented in Tables 5.3 and 5.4. The resulting process cycles and state trajectories are presented in Figure 5.3.

Table 5.3: **6 stage system showing HIGH–to–LOW step change in salt concentration**

Operating conditions	
Number of stages	6
Resin volume	590 ml
Feed flow rate	750 $ml/\min$
Upflow time	15 $\min$
Feed concentration before step change (start)	90.2 meq/l
Feed concentration at step change (final)	50.7 meq/l
R/L ratio at start before step change	0.81
R/L ratio at the end (start of step change)	1.44

Table 5.4: **Run 2 H^+ fractional change values for the 6 stage system showing HIGH–to–LOW concentration step as determined from Equation (5.2)**

Concentration as determined by fractional change FC_nH^+						
Stage No. / Cycle No.	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	stage 6
1	0.014	0.009	0.014	0.013	0.077	0.085
2	0.002	0.003	0.021	0.059	0.115	0.254
3	0.031	0.017	0.057	0.141	0.185	0.571
4	0.036	0.052	0.079	0.204	0.311	0.764
5	0.063	0.059	0.126	0.264	0.482	0.882
6	0.062	0.085	0.170	0.388	0.612	0.901
7	0.088	0.117	0.235	0.475	0.703	0.940
8	0.119	0.152	0.312	0.560	0.782	1.000
9	0.152	0.191	0.403	0.668	0.853	0.995
10	0.183	0.250	0.471	0.739	0.879	1.000
11	0.209	0.295	0.591	0.790	0.904	1.000
12	0.204	0.350	0.407	0.825	0.927	1.000
13	0.000	0.490	0.693	0.836	0.942	1.000
14	0.332	0.476	0.716	0.875	0.962	1.000
15	0.170	0.538	0.756	0.916	0.968	1.000
16	0.413	0.577	0.764	0.918	0.981	1.000
17	0.449	0.645	0.840	0.942	0.990	1.000
18	0.525	0.664	0.874	0.953	1.000	1.000
19	0.591	0.752	0.881	0.965	1.000	1.000
20	0.634	0.780	0.916	0.968	1.000	1.000
21	0.670	0.809	0.000	1.000	1.000	1.000
22	0.694	0.829	0.961	0.987	1.000	1.000
23	0.775	0.868	0.995	1.000	1.000	1.000
24	0.812	0.920	1.000	1.000	1.000	1.000
25	0.836	0.939	1.000	1.000	1.000	1.000
26	0.903	0.959	1.000	1.000	1.000	1.000
27	0.931	1.000	1.000	1.000	1.000	1.000

The results for the plots obtained from a MATLAB software programs for each of the above operational cases for the 6 stage and the 12 stage ion exchange processes are shown in Figures 5.2.–5.3. below (the corresponding MATLAB software programs are attached in APPENDIX A1–A2.

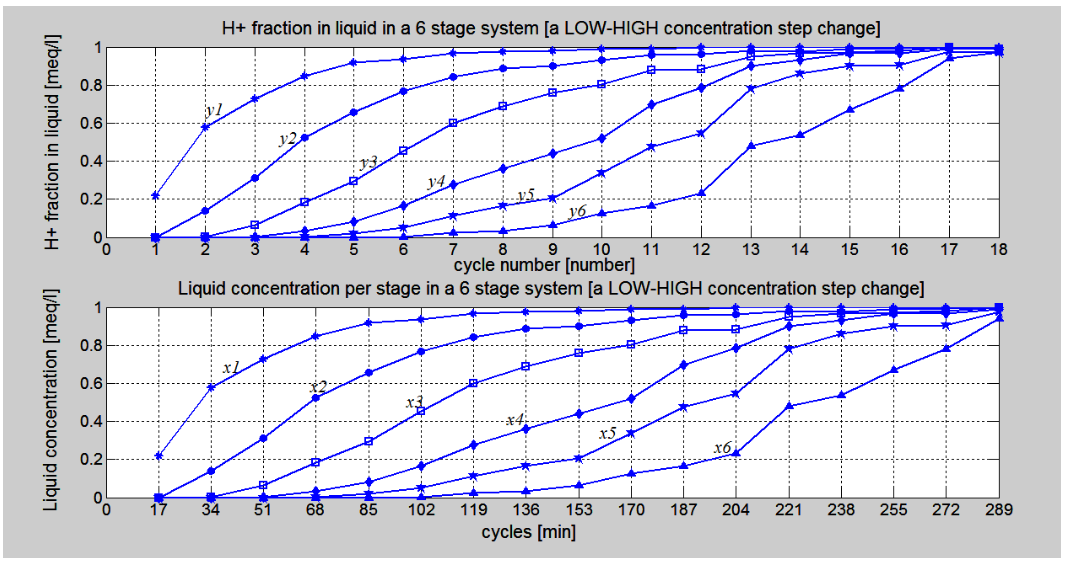


Figure 5.2. H^+ **fractional change values for the 6 stage system of a LOW to HIGH concentration step change**

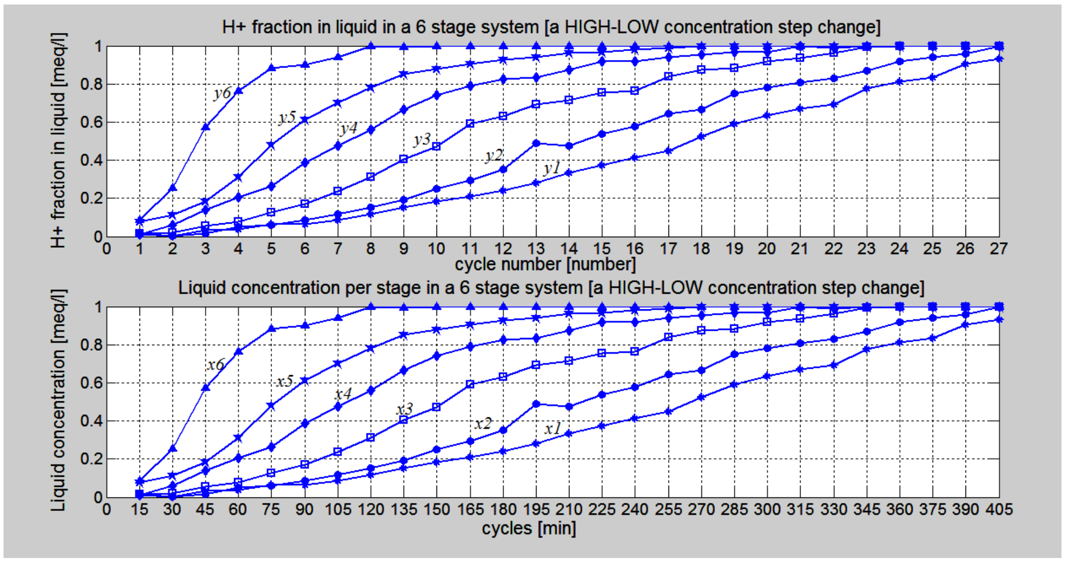


Figure 5.3. H^+ **fractional change values for the 6 stage system of a HIGH to LOW concentration step change**

5.3.2. The 12 stage system data

Table 5.5. below show the operational conditions for $Run\,3$, which considers the concentration step change at the input feed that moves from a LOW to HIGH. The operational conditions and the resulting measurement data obtained are presented in the Tables 5.5. and 5.6. The resulting stage trajectories and the data are presented in Figure 5.4–5.5.

Table 5.5: 12 stage system LOW–to–HIGH step change in concentration

Operating conditions	
Number of stages	12
Resin volume	550 ml
Feed flow rate	750 $ml/\min$
Upflow time	15 $\min$
Feed concentration before step change (start)	45.7 meq/l
Feed concentration at step change (final)	92.3 meq/l
R/L ratio at start before step change	1.50
R/L ratio at the end (start of step change)	0.74

Table 5.6: Run 3 H^+ fractional change values for the 12 stage system moving from LOW–to–HIGH as determined from Equation (5.2)

Concentration as determined by fractional change FC_nH^+						
Stage No. / Cycle No.	stage 2	stage 4	stage 6	stage 8	stage 10	stage 12
1	0.000	0.012	0.000	0.000	0.010	0.014
2	0.254	0.013	0.024	0.000	0.000	0.020
3	0.517	0.046	0.000	0.000	0.015	0.057
4	0.698	0.088	0.000	0.000	0.014	0.000
5	0.726	0.238	0.008	0.000	0.000	0.000
6	0.776	0.299	0.021	0.000	0.000	0.000
7	0.910	0.633	0.171	0.000	0.000	0.000
8	0.849	0.725	0.283	0.001	0.000	0.011
9	0.948	0.778	0.000	0.044	0.026	0.000
10	0.954	0.814	0.523	0.147	0.044	0.000
11	0.960	0.821	0.673	0.210	0.061	0.014
12	0.980	0.797	0.737	0.279	0.100	0.010
13	0.986	0.946	0.841	0.532	0.144	0.006
14	0.984	0.946	0.892	0.633	0.235	0.060
15	1.000	0.960	0.918	0.727	0.340	0.083
16	1.000	0.965	0.934	0.805	0.506	0.140
17	1.000	0.955	0.937	0.830	0.000	0.000
18	1.000	0.969	0.904	0.861	0.657	0.271
19	1.000	0.973	0.980	0.879	0.800	0.374
20	1.000	0.992	0.973	0.937	0.858	0.470
21	1.000	0.992	0.983	0.967	0.914	0.637
22	1.000	0.984	0.995	0.960	0.975	0.814
23	1.000	1.000	1.000	1.000	1.000	1.000

For simplicity in plotting of the plant behaviour, stages in the 12 stage system have been renamed stage 1 through to stage 6 instead of stages 2 through to 12.

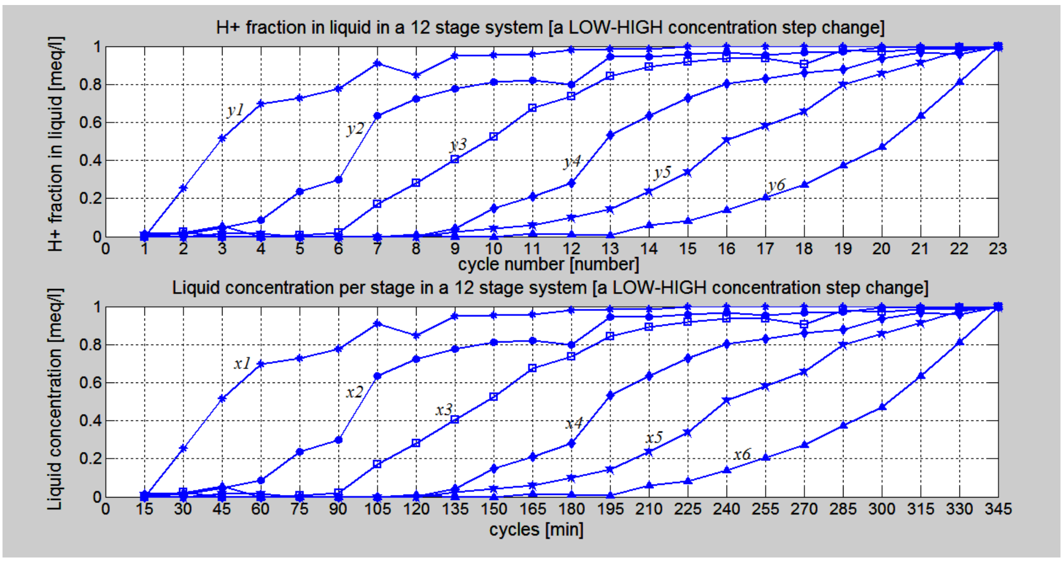


Figure 5.4. H^+ **fractional change values for the 12 stage system moving from a LOW to HIGH concentration step change**

Operational conditions for $Run\ 4$ which considers a HIGH to LOW concentration step change at the input feed of the 12 stage system are presented in Table 5.7. Measurement data obtained from this system are presented in Table 5.8. The resulting operational cycles and system trajectories are presented in Figure 5.5.

Table 5.7: **12 stage system HIGH–to–LOW step change in concentration**

Operating conditions	
Number of stages	12
Resin volume	660 ml
Feed flow rate	750 $ml/\min$
Upflow time	15 $\min$
Feed concentration before step change (start)	90.8 meq/l
Feed concentration at step change (final)	52.8 meq/l
R/L ratio at start before step change	0.89
R/L ratio at the end (start of step change)	1.56

Table 5.8: **Run 4** H^+ **fractional change values for the 12 stage system moving from HIGH–to–LOW as determined from Equation (5.2)**

Concentration as determined by fractional change FC_nH^+						
Stage No. / Cycle No.	Stage 2	Stage 4	Stage 6	Stage 8	Stage 10	Stage 12
1	0.058	0.020	0.002	0.000	0.000	0.000
2	0.066	0.009	0.033	0.101	0.000	0.000
3	0.037	0.016	0.026	0.083	0.307	0.210
4	0.075	0.024	0.134	0.204	0.567	0.540
5	0.037	0.026	0.092	0.337	0.641	0.780

6	0.073	0.071	0.310	0.489	0.762	0.810
7	0.074	0.085	0.188	0.569	0.787	0.780
8	0.089	0.164	0.380	0.764	0.875	0.931
9	0.113	0.191	0.499	0.823	0.897	1.020
10	0.139	0.245	0.569	0.861	0.926	0.991
11	0.146	0.322	0.674	0.902	0.949	1.080
12	0.170	0.376	0.690	0.909	0.982	1.000
13	0.201	0.448	0.756	0.953	0.972	1.000
14	0.227	0.548	0.815	0.962	1.000	1.000
15	0.300	0.620	0.846	0.986	1.000	1.000
16	0.332	0.596	0.887	1.000	1.000	1.000
17	0.405	0.643	0.885	1.000	1.000	1.000
18	0.554	0.839	0.956	1.000	1.000	1.000
19	0.648	0.894	0.973	1.000	1.000	1.000
20	0.760	1.000	1.000	1.000	1.000	1.000
21	0.898	1.000	1.000	1.000	1.000	1.000

5.4. Results and developments for the current project

Since the developments in the thesis are done considering the sodium concentration in the liquid, the Na^+ ionic fraction, values for the corresponding fractional change of Na^+ data are obtained from the H^+ fractional change data above using Equation (5.3) and are presented below in the next four tables, Table 5. 9–5.12.

$$FC_n Na^+ = 1 - FCH_n^+ \tag{5.3}$$

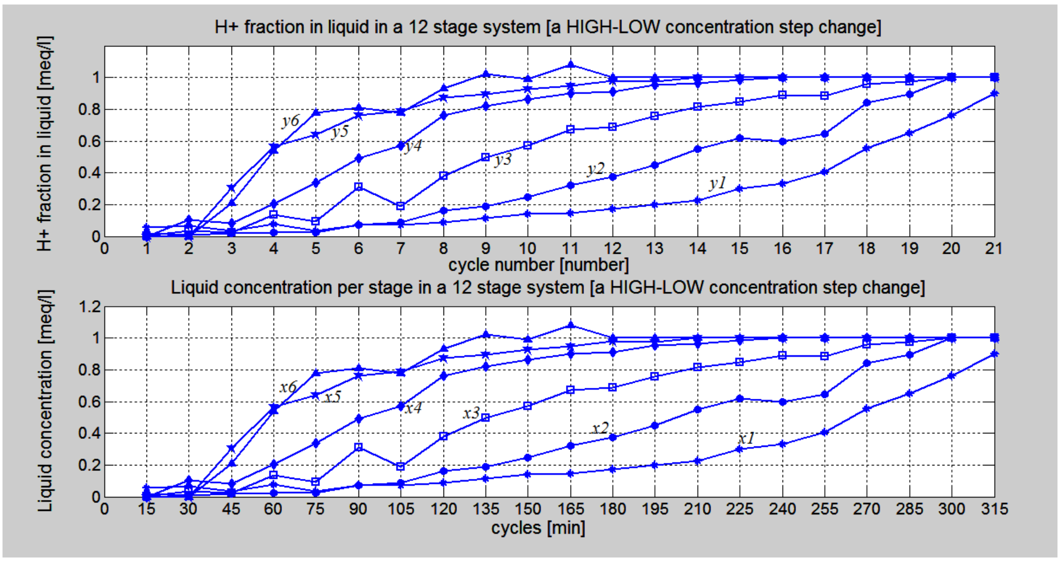


Figure 5.5. H^+ **fractional change values for the 12 stage system moving from HIGH to LOW concentration step change**

Table 5.9: Run 5 Na^+ fractional change values from LOW to HIGH concentration step change for the 6 stage system as determined from Equation (5.3)

Concentration as determined by fractional change $1-FC_nNa^+$						
Stage No. / Cycle No.	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
1	0.779	1.000	1.000	1.000	1.000	1.000
2	0.423	0.860	0.996	1.000	1.000	1.000
3	0.270	0.686	0.934	0.996	1.000	1.000
4	0.153	0.477	0.816	0.965	0.997	1.000
5	0.080	0.344	0.705	0.918	0.980	1.000
6	0.064	0.234	0.546	0.832	0.948	0.999
7	0.032	0.158	0.399	0.723	0.887	0.976
8	0.031	0.114	0.310	0.639	0.833	0.967
9	0.026	0.100	0.242	0.560	0.793	0.937
10	0.019	0.067	0.196	0.478	0.660	0.876
11	0.011	0.042	0.123	0.302	0.526	0.833
12	0.012	0.037	0.119	0.216	0.453	0.767
13	0.003	0.018	0.049	0.101	0.220	0.518
14	0.000	0.026	0.035	0.069	0.140	0.461
15	0.000	0.009	0.028	0.034	0.101	0.328
16	0.000	0.007	0.019	0.034	0.095	0.221
17	0.000	0.006	0.000	0.009	0.025	0.060
18	0.000	0.007	0.012	0.009	0.027	0.028

Table 5.10: Run 6 Na^+ fractional change values from LOW to HIGH concentration step change for the 6 stage system as determined from Equation (5.3)

Concentration as determined by fractional change $1-FC_nH^+$						
Stage No. / Cycle No.	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
1	0.986	0.991	0.986	0.987	0.923	0.915
2	0.998	0.997	0.979	0.941	0.885	0.746
3	0.969	0.983	0.943	0.859	0.815	0.429
4	0.964	0.948	0.921	0.796	0.689	0.236
5	0.937	0.041	0.874	0.736	0.518	0.118
6	0.938	0.915	0.830	0.612	0.388	0.099
7	0.912	0.883	0.765	0.525	0.297	0.060
8	0.881	0.848	0.688	0.440	0.218	0.000
9	0.848	0.809	0.597	0.332	0.147	0.005
10	0.817	0.750	0.529	0.261	0.121	0.000
11	0.291	0.705	0.409	0.210	0.096	0.000

12	0.760	0.650	0.593	0.175	0.073	0.000
13	1.000	0.510	0.307	0.164	0.058	0.000
14	0.668	0.524	0.284	0.125	0.038	0.000
15	0.830	0.462	0.244	0.084	0.032	0.000
16	0.587	0.423	0.236	0.082	0.019	0.000
17	0.551	0.355	0.160	0.058	0.010	0.000
18	0.475	0.336	0.126	0.047	0.000	0.000
19	0.409	0.248	0.119	0.035	0.000	0.000
20	0.366	0.220	0.084	0.032	0.000	0.000
21	0.330	0.191	1.000	0.000	0.000	0.000
22	0.306	0.171	0.039	0.013	0.000	0.000
23	0.225	0.132	0.005	0.000	0.000	0.000
24	0.188	0.080	0.000	0.000	0.000	0.000
25	0.164	0.061	0.000	0.000	0.000	0.000
26	0.097	0.041	0.000	0.000	0.000	0.000
27	0.069	0.000	0.000	0.000	0.000	0.000

Table 5.11: **Run 7 Na^+ fractional change values from LOW to HIGH concentration step change for the 12 stage system as determined from Equation (5.3)**

Concentration as determined by fractional change $1-FC_nH^+$						
Stage No. / Cycle No.	Stage 2	Stage 4	Stage 6	Stage 8	Stage 10	Stage 12
1	1.000	0.988	1.000	1.000	0.990	0.986
2	0.746	0.987	0.976	1.000	1.000	0.980
3	0.483	0.954	1.000	1.000	0.985	0.943
4	0.302	0.912	1.000	1.000	0.986	1.000
5	0.274	0.762	0.992	1.000	1.000	1.000
6	0.224	0.701	0.979	1.000	1.000	1.000
7	0.090	0.367	0.829	1.000	1.000	1.000
8	0.151	0.275	0.717	0.999	1.000	0.989
9	0.052	0.222	1.000	0.956	0.974	1.000
10	0.046	0.186	0.477	0.853	0.956	1.000
11	0.040	0.179	0.327	0.790	0.939	0.986
12	0.020	0.203	0.263	0.721	0.900	0.910
13	0.014	0.054	0.159	0.468	0.856	0.994
14	0.016	0.054	0.108	0.367	0.765	0.940
15	0.000	0.040	0.082	0.273	0.660	0.917
16	0.000	0.035	0.066	0.195	0.494	0.860
17	0.000	0.045	0.063	0.170	1.000	1.000

18	0.000	0.031	0.096	0.139	0.343	0.729
19	0.000	0.027	0.020	0.121	0.200	0.626
20	0.000	0.008	0.027	0.063	0.142	0.530
21	0.000	0.008	0.017	0.033	0.086	0.363
22	0.000	0.016	0.005	0.040	0.025	0.186
23	0.000	0.000	0.000	0.000	0.000	0.000

Table 5.12: **Run 8 Na^+ fractional change values from HIGH to LOW concentration step change for the 12 stage system values as determined from Equation (5.3)**

Concentration as determined by fractional change $1-FC_nH^+$						
Stage No. / Cycle No.	Stage 2	Stage 4	Stage 6	Stage 8	Stage 10	Stage 12
1	0.942	0.980	0.998	1.000	1.000	1.000
2	0.934	0.991	0.967	0.899	1.000	1.000
3	0.963	0.984	0.974	0.917	0.693	0.790
4	0.925	0.976	0.866	0.796	0.433	0.460
5	0.963	0.974	0.908	0.663	0.359	0.220
6	0.927	0.929	0.690	0.511	0.238	0.190
7	0.926	0.915	0.812	0.431	0.213	0.220
8	0.911	0.836	0.620	0.236	0.125	0.069
9	0.887	0.809	0.501	0.177	0.103	0.020
10	0.861	0.755	0.431	0.139	0.074	0.009
11	0.854	0.678	0.326	0.098	0.051	0.080
12	0.830	0.324	0.310	0.091	0.018	0.000
13	0.799	0.552	0.244	0.047	0.028	0.000
14	0.773	0.452	0.185	0.038	0.000	0.000
15	0.700	0.380	0.154	0.014	0.000	0.000
16	0.668	0.404	0.113	0.000	0.000	0.000
17	0.595	0.357	0.115	0.000	0.000	0.000
18	0.446	0.161	0.044	0.000	0.000	0.000
19	0.352	0.106	0.027	0.000	0.000	0.000
20	0.240	0.000	0.000	0.000	0.000	0.000
21	0.102	0.000	0.000	0.000	0.000	0.000

The expected outputs per stage are plotted in the following graphs, Figures 5.6–5.10. The accompanying MATLAB software programs are attached in the APPENDICES A.1–A.8. For simplicity in plotting of the plant behaviour, the even numbered stages in the 12 stage system have been renamed Stage 1 through to Stage 6 instead of Stages 2 through to 12, and are presented.

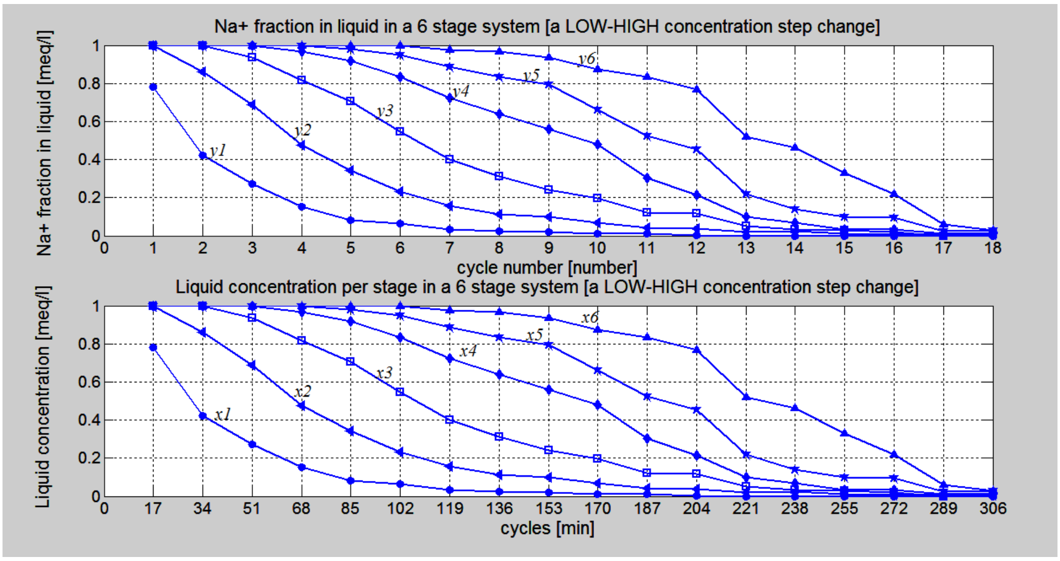


Figure 5.6. Na^+ fractional change values of the 6 stage system for a LOW to HIGH concentration step change

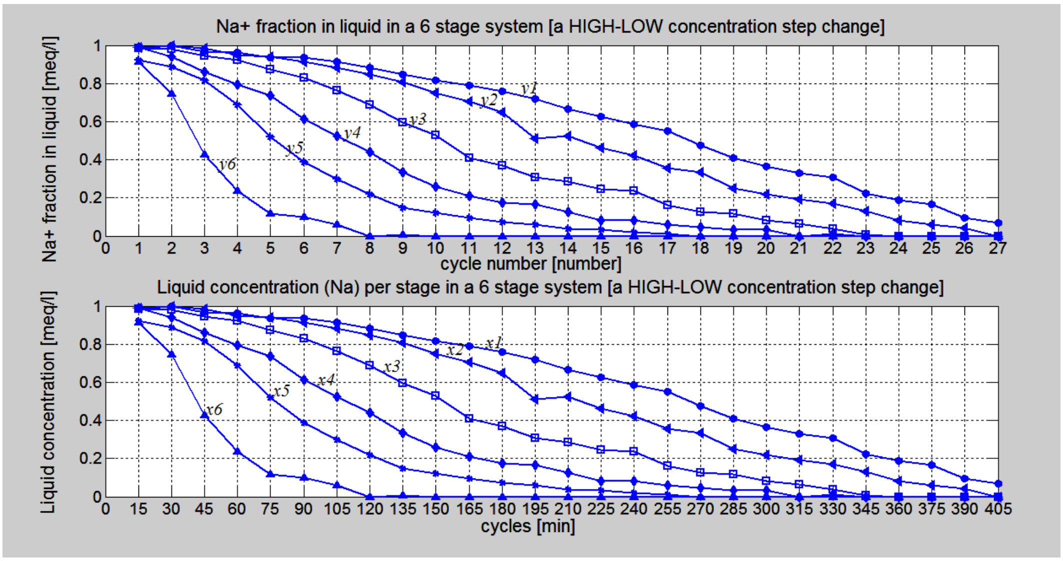


Figure 5.7. Na^+ fractional change values of the 6 stage system for a HIGH to LOW concentration step change

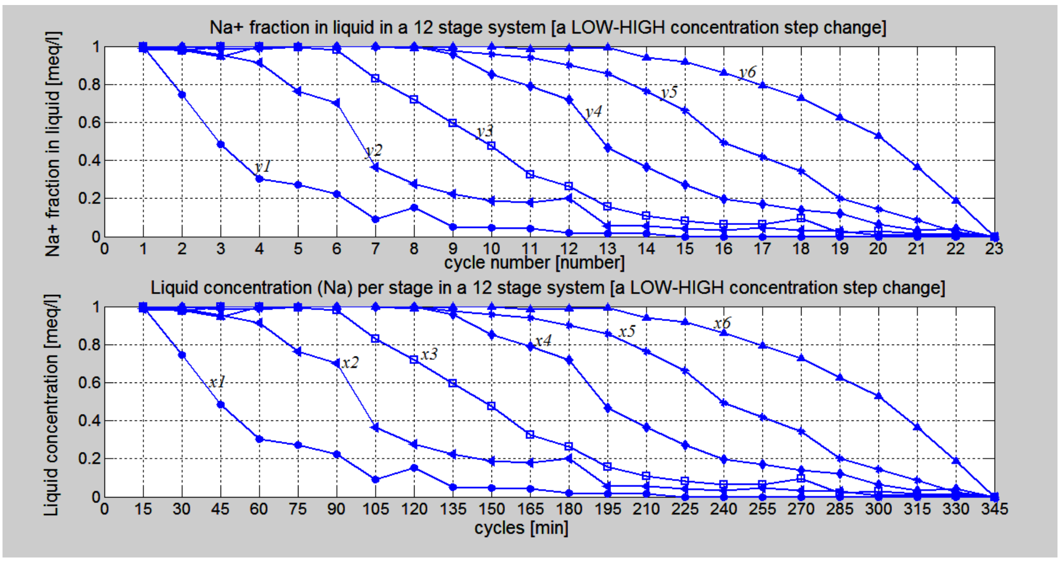


Figure 5.8. Na^+ **fractional change values of the 12 stage system for a LOW to HIGH concentration step change**

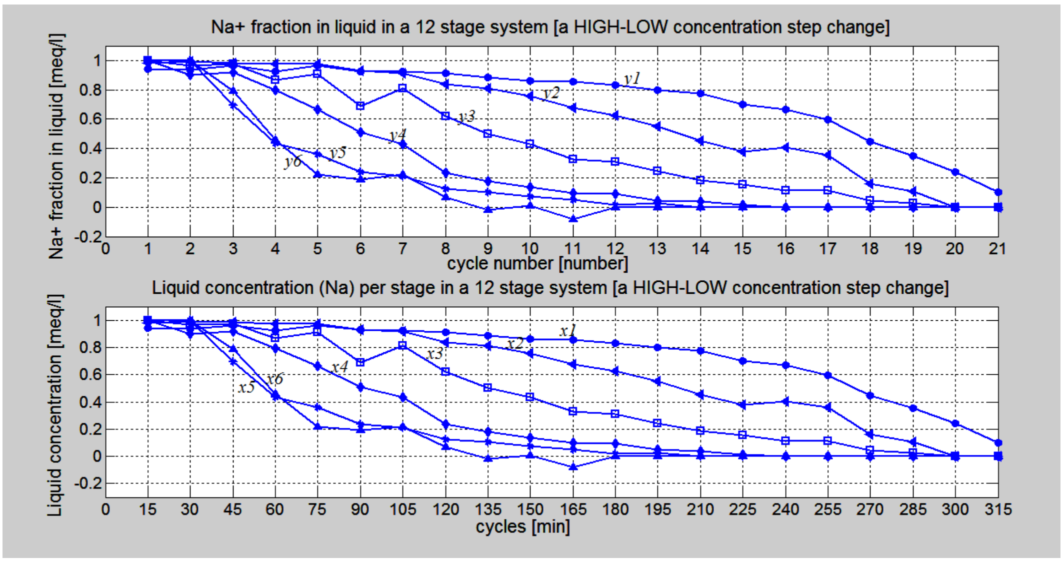


Figure 5.9. Na^+ **fractional change values of the 12 stage system for a HIGH to LOW concentration step change**

The ionic fraction of sodium in liquid data presented from MATLAB software programs (Figures 5.6–5.9) validate the feasibility of using Na^+ ionic fraction in liquid for optimal control purposes. The system trajectories do match the process model design trajectories based on calculations. It is therefore possible to use this data for both parameter and state estimation purposes for validation of the developed model.

5.5. Model reformulation for parameter estimation problem based on process ionic fraction data

The data presented above has been identified as that of either 1) *sodium in the resin* (y_{Na^+}) or 2) *hydrogen in the liquid* (x_{H^+}), given that throughout the column from the bottom up, the concentrations of counter ions are decreasing from one stage to the next and increasing in each stage throughout the number of cycles to reach steady state. From this observation it was also concluded that this data can never be that of 1) *sodium in liquid* (x_{Na^+}) or hydrogen in resin (y_{H^+}).

The flow direction of each phase is very important in presenting these dynamic behavioural structures. Firstly, starting with simpler presentations, *for the resins going down the column*, and its hydrogen content at the top stage; resin has highest content of hydrogen. At the top stage the hydrogen content of resin is highest and if normalized, it can be presented to having a value of 1 and if resin is well exchanged, it would have a lowest value of hydrogen, close to 0 in the bottom stage Figure 5.10.

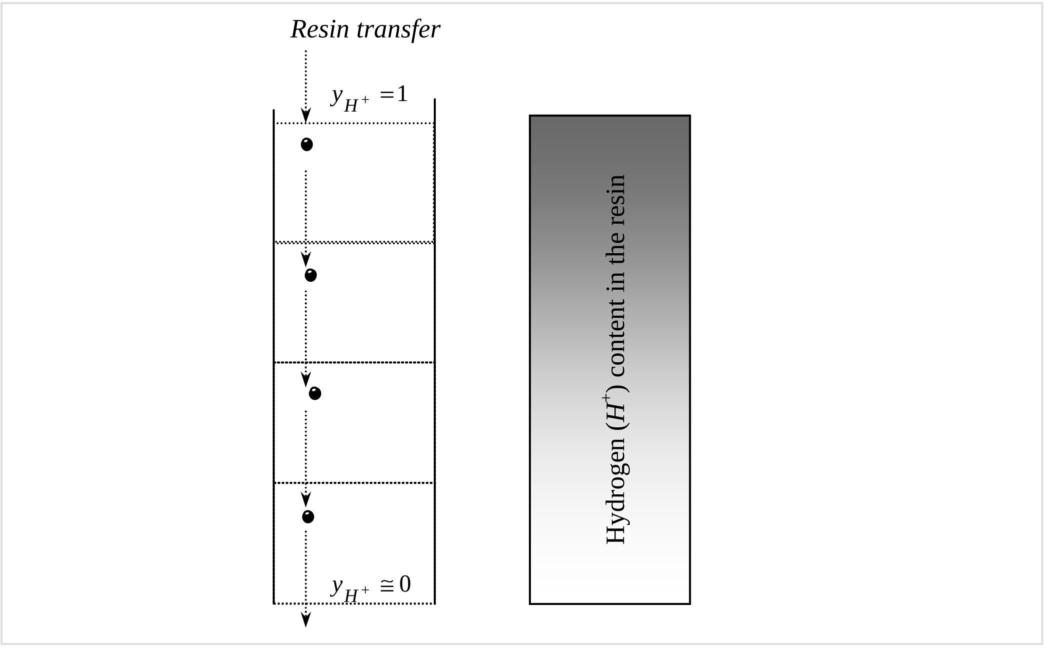


Figure 5.10. **The basic ion exchange resin hydrogen ionic fraction**

If the sodium behaviour is considered throughout the column in the *resin phase*, the *ionic content of sodium* is *lower* at the most top stage (last stage) compared to the bottom stages, Figure 5.11. More sodium ions are absorbed by the resin at the top stage since there is *more hydrogen to exchange with sodium ions* compared to the more saturated resin at the bottom, take note not to confuse with sodium in the liquid phase.

If the *liquid phase* is considered, the liquid going up the column, *the sodium at the* most *bottom stage* (first stage) is at its highest throughout the column compared to the highest stage.

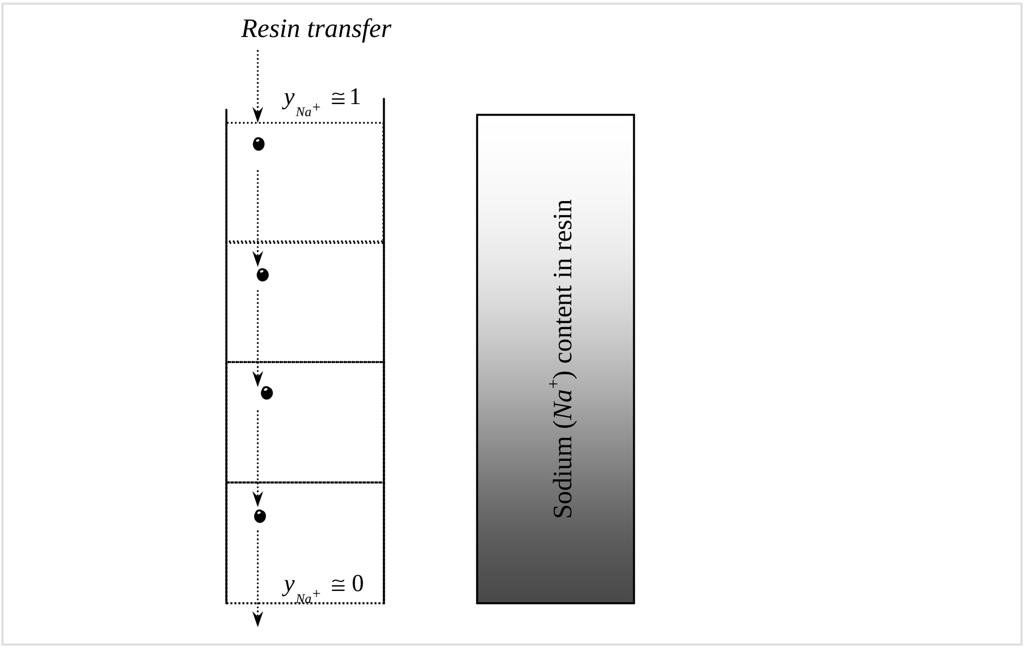


Figure 5.11. **The basic ion exchange resin hydrogen ionic fraction**

This is shown by the graphical representation in Figure 5.12. The *hydrogen in the liquid phase* can be seen to be the lowest in the highest (top) stage since

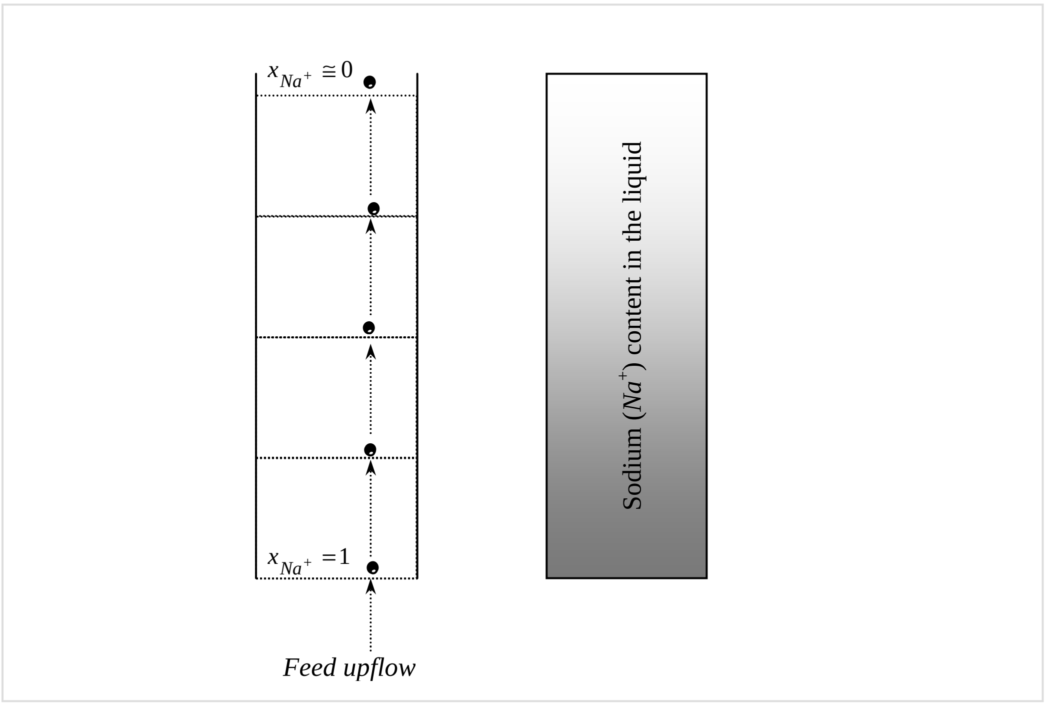


Figure 5.12. **The basic ion exchange sodium ionic fraction in liquid phase**

most hydrogen is in the resin. More *hydrogen ionic content* (H^+) is found in the liquid at the most bottom stage (first stage) due to more sodium ionic content being exchanged with the more sodium coming in with the feed.

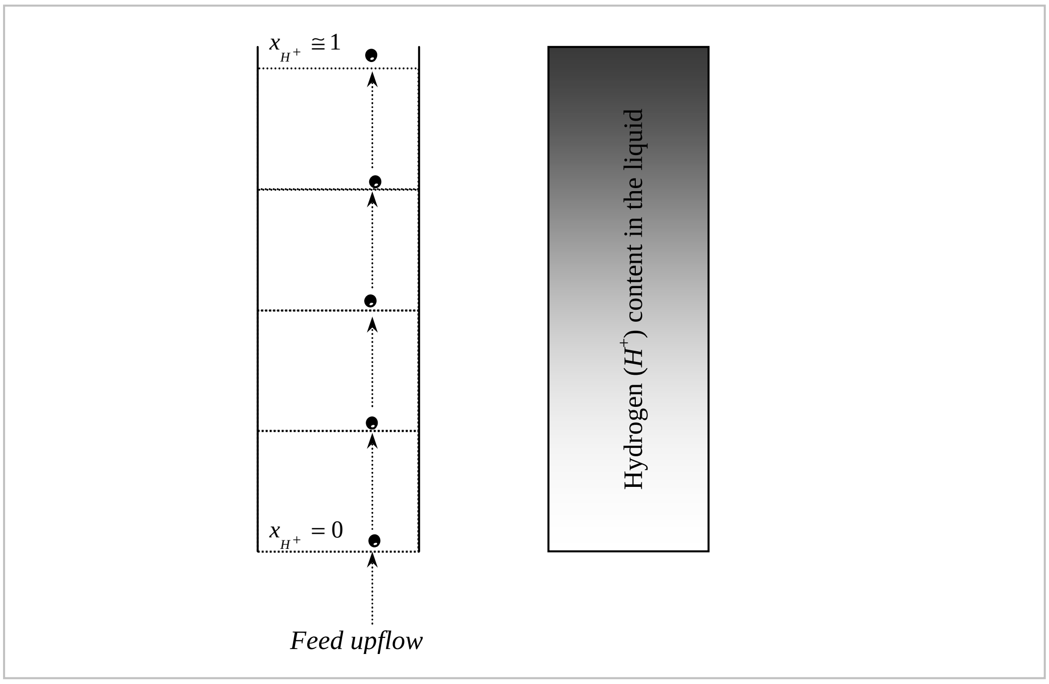


Figure 5.13. **The basic ion exchange hydrogen ionic fraction in liquid phase**

This means that the resin will release more hydrogen at the first stage (bottom stage) compared to the last stage (top stage) looking at it in terms of the liquid phase, Figure 5.13.

Considering the behaviour of the counter ions instantaneously in each stage and the ionic fraction in each flow against time, the behaviour of each ionic species within a stage is the same in time whether at the top stage or bottom stage; the only difference is the behaviour throughout the column for each flow, going up the column (liquid phase) or down the column (resin phase). This ionic behaviour throughout the column is presented in the flow chart in Figure 5.13. This presentation can be supplemented by the flow chart representing the behaviour of the counter ions species as seen throughout the load column (from bottom up, for liquid flow and top to bottom for resin flow), Figure 5.14. The darker shades mean more ionic species of the relevant ion.

Based on this ionic dynamic behaviour, the parameter estimation problem can be formulated and solved. Using the load column exchanging cations between in liquid (sodium) and resin (hydrogen), the exchange dynamics are represented by the equation,

$$R - H + NaCl \leftrightarrow R - Na + HCl \tag{5.4}$$

According to Equation (5.4) for the ion exchange to occur, the cation resin is in H^+ form and the liquid contains a strong electrolyte, in this case the sodium chloride, $NaCl$. If enough time is allowed for the exchanging ions to react, the reaction will proceed until some equilibrium is reached.

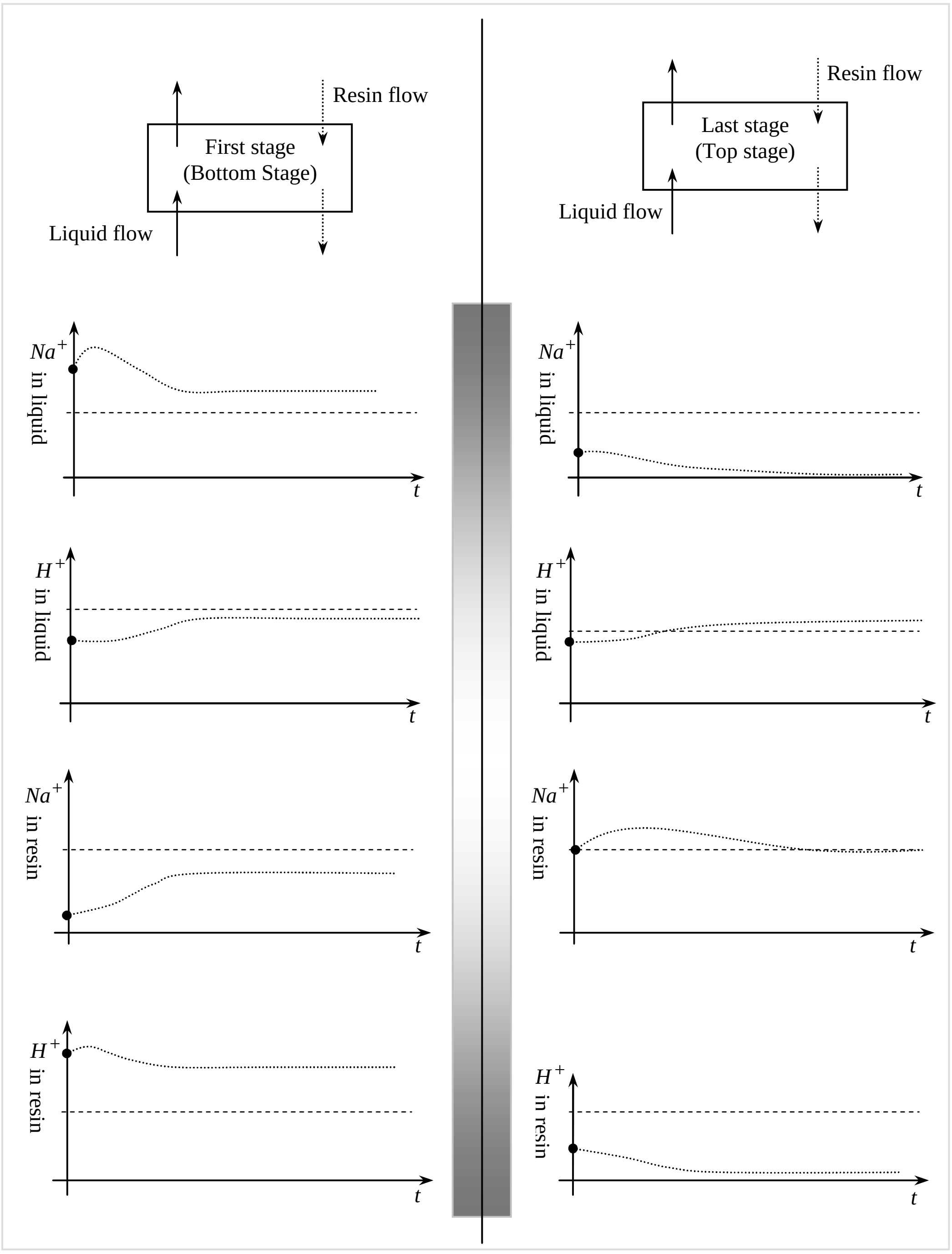


Figure 5.14. The basic ion exchange sodium and hydrogen ionic fraction relationship in each stage of the CCIX column for both the liquid and resin phases

According to Hendry (1982b:19) at this equilibrium point there will be competing counter ions (Na^+ and H^+) in both the liquid and the resin, i.e., for every movement of one counter ion from one phase to the other, there will be another movement of the same counter ion but in the opposite direction, Figure 5.15.

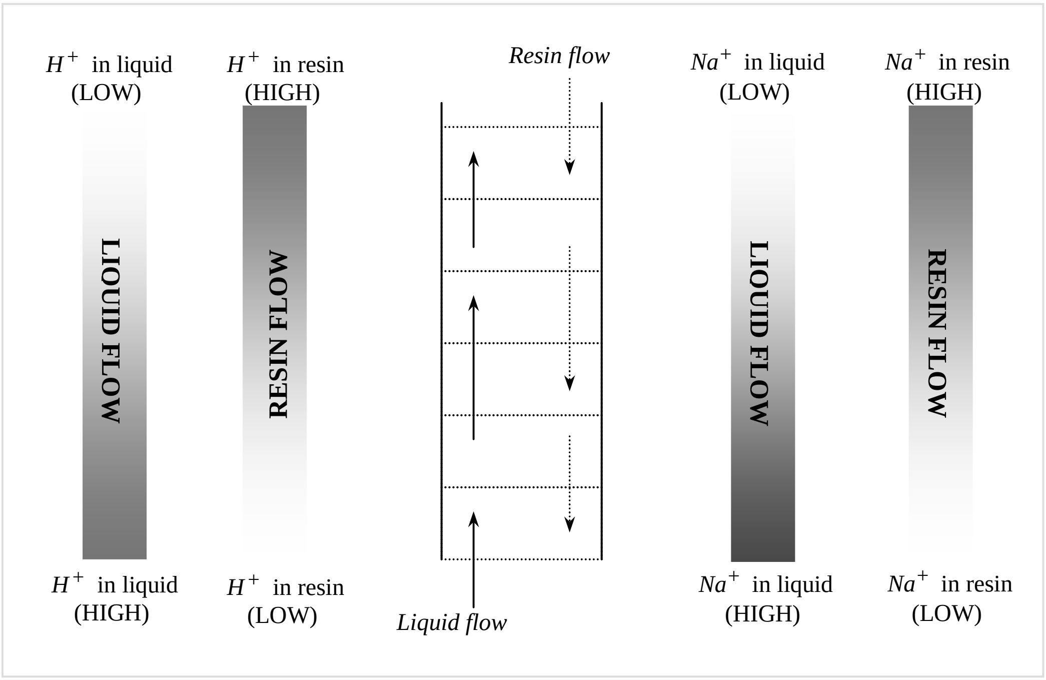


Figure 5.15. **Behaviour of hydrogen and sodium ionic fraction throughout the cation column**

5.5.1. Model parameters selection

The resin phase is represented by y_n and the liquid phase is represented by x_n ionic fractions in the resin and liquid phases. The ionic balance in each phase can be defined as follows:

$$y_{H+} = \frac{m_{H+}\overline{C}_{H+}}{m_{H+}\overline{C}_{H+} + m_{Na+}\overline{C}_{Na+}} \tag{5.5}$$

$$y_{Na+} = \frac{m_{Na+}\overline{C}_{Na+}}{m_{H+}\overline{C}_{H+} + m_{Na+}\overline{C}_{H+}}$$

$$x_{H+} = \frac{m_{H+}C_{H+}}{m_{H+}C_{H+} + m_{Na+}C_{Na+}}$$

$$x_{Na+} = \frac{m_{Na+}C_{Na+}}{m_{H+}C_{H+} + m_{Na+}C_{Na+}}$$

where

y_{H+} – is the equivalent ionic fraction of hydrogen ions (H^+) in the resin phase,

y_{Na+} – is the equivalent ionic fraction of sodium ions (Na^+) in the resin phase,

x_{H+} – is the equivalent ionic fraction of hydrogen ions (H^{+}) in the liquid phase,

x_{Na+} – is the equivalent ionic fraction of sodium ions (Na^{+}) in the liquid phase,

C_{H+} – the concentration of hydrogen (H) in the liquid phase,

C_{Na+} – the concentration of sodium (Na) in the liquid phase,

$\overline{C}_{H+}$ – the concentration of hydrogen (H) in the resin phase,

$\overline{C}_{Na+}$ – the concentration of sodium (Na) in the resin phase,

m_{H+} – is the absolute value of the electrochemical valence of hydrogen in a stage,

m_{Na+} – is the absolute value of the electrochemical valence of sodium in a stage.

For each phase (resin or liquid), the total ionic content is defined by (Hendry, 1982a; Hendry, 1982e):

$$\underset{(resin)}{y_H} + \underset{(resin)}{y_{Na}} = 1 \tag{5.6}$$

$$\underset{(liquid)}{x_H} + \underset{(liquid)}{x_{Na}} = 1 \tag{5.7}$$

On the bases of these equilibria equations, Equations (5.6)–(5.7) and the general state space model from Chapter 4, given by Equation (5.8), a reformulated state space model can be developed with the aim of solving the parameter estimation problem using the linear relationship that exists between the phases, Equations (5.9) and (5.10). The general model is given by:

$$H\dot{x}_n(t) + h\dot{y}_n(t) = F_L(x_{(n-1)}(t) - x_n(t)) + F_R(y_{(n-1)}(t) - y_n(t)),\ n = \overline{1, N} \tag{5.8}$$

The linear relationship between resin and liquid phase can be given by:

$$\underset{(liquid)}{x_n(t)} = \overline{a}_n \underset{(resin)}{y_n(t)} + \overline{b}_n \tag{5.9}$$

$$\underset{(resin)}{y_n(t)} = a_n \underset{(liquid)}{x_n(t)} + b_n \tag{5.10}$$

Coefficients in Equations (5.9) depend on the value of the coefficients in Equation (5.10). If the liquid ionic content in Equation (5.10) is represented by the resin ionic content it can be written in the following manner:

$$x_n(t) = 1/(a_n)y_n(t) - b/a_n = \overline{a}_n y_n(t) + \overline{b}_n \tag{5.11}$$

From the Equation (5.11), the unknown coefficients can be solved as,

$$\overline{a}_n = 1/a_n\,;\ \overline{b}_n = b_n/a_n \tag{5.12}$$

Considering the model using concentration in the liquid phase to express the concentration in the resin phase, Equations (5.8) and (5.10), and following the derivations in Chapter 4, the model given by Equation (4.24) is obtained as follows:

$$\begin{aligned}\dot{x}_n(t) &= \frac{F_L}{(H+a_n h)} x_{(n-1)}(t) - \frac{F_L}{(H+a_n h)} x_n(t) + \\ &+ \frac{a_{(n+1)}}{(H+a_n h)} x_{(n+1)}(t) F_R(t) - \frac{a_n}{(H+a_n h)} x_n(t) F_R(t) + \frac{b_{(n+1)} - b_n}{(H+a_n h)} F_R\end{aligned}, \quad n = \overline{1,N}$$

(5.13)

Considering the concentration in resin phase, Equations (5.8) and (5.11),

$$\begin{aligned}H\frac{d(\bar{a}_n y_n(t) + \bar{b}_n)}{dt} + h\frac{dy_n(t)}{dt} &= F_L(\bar{a}_{(n-1)} y_{(n-1)}(t) + \bar{b}_{(n-1)}) + \\ &+ F_L(\bar{a}_n y_n(t) + \bar{b}_n) + y_{(n+1)}(t) F_R(t) - y_n(t) F_R(t)\end{aligned}, \quad n = \overline{1,N} \qquad (5.14)$$

$$\begin{aligned}\dot{y}(t) &= \frac{\bar{a}_{(n-1)}}{(\bar{a}_n H + h)} F_L y_{(n-1)}(t) - \frac{\bar{a}_n}{(\bar{a}_n H + h)} F_L y_n(t) + \\ &+ \frac{1}{(\bar{a}_n H + h)} y_{(n+1)}(t) F_R(t) - \frac{1}{(\bar{a}_n H + h)} y_n(t) F_R(t) - \frac{\bar{b}_n + \bar{b}_{(n-1)}}{(\bar{a}_n H + h)} F_R(t)\end{aligned}, n = \overline{1,N}$$

(5.15)

Depending on measurements data available, the relevant model (variable or variables of interest in resin or liquid phase) will be considered in solving the estimation problems. In this case, the model Equation (5.17) cannot be used with the developed model for parameter estimation given that the available data from Hendry (1982d) is presented only for hydrogen content in liquid. There are a couple of important factors to consider before choosing a relevant model to use. The closeness in the presentation of a variable of interest may render the model unusable if incorrect combination of model and data are used. The following are a few points that confronted the author for the model reformulation for parameter estimation problem:

- Measurements data presented were the ionic fraction of hydrogen in liquid (x_{H^+}) and the initial model developed (see Chapter 4) is based on sodium content in the liquid (x_{Na^+}),
- In the project documents of Hendry (1982) the reference to hydrogen H^+ content was always based on the resin (y_{H^+}),
- The behaviour of each counter ion H^+ or Na^+ is very different when considered in 1) *a different phase*, *resin or liquid*, 2) *different stage* and 3) *throughout a column*,

- Decision on which variable of interest to choose and where it is considered in the three points mentioned above will make or break the estimation solution; and therefore needs to match the design requirements,
- The dynamic behaviour of the counter ions is the same in some of the possibilities from the three points above but with different values,
- The model may become too complex when trying to directly related resin phase and liquid phase using a single counter ion, e.g., Na^+ in resin and liquid, Equations (5.6) and (5.7). On the basis of these equations such a property called the *separation factor* can be considered as follows:

$$\alpha = \frac{\overset{\text{(liquid)}}{x_{Na}} \; \overset{\text{(resin)}}{y_H}}{\underset{\text{(liquid)}}{x_H} \; \underset{\text{(resin)}}{y_{Na}}} \tag{5.16}$$

The separation factor describes the exchanging counter ions at equilibrium. Separation factor for sodium Na^+ and hydrogen H^+ ions in this case, can be described using Equations (5.17) and (5.18).

$$\alpha_H^{Na} = \frac{y_H x_{Na}}{y_{Na} x_H} = \frac{x_{Na}(1 - y_{Na})}{y_{Na}(1 - x_{Na})} \tag{5.17}$$

$$\alpha_{Na}^{H} = \frac{y_H x_{Na}}{y_{Na} x_H} = \frac{y_H(1 - x_H)}{x_H(1 - y_H)} \tag{5.18}$$

From Equation (5.17) the sodium ionic fraction can be related to hydrogen ionic fraction through separation factor in this way,

$$y_{Na} = \frac{x_{Na}(1 - y_{Na})}{\alpha_H^{Na}(1 - x_{Na})} \tag{5.19}$$

$$y_{Na}\alpha_H^{Na} - y_{Na}\alpha_H^{Na} x_{Na} = x_{Na} - x_{Na} y_{Na} \tag{5.20}$$

$$y_{Na} = \frac{x_{Na}}{\alpha_H^{Na} - x_{Na}\alpha_H^{Na} + x_{Na}} \tag{5.21}$$

Attempting to use the model in Equation (5.21) direct with the general model will result in a very complex model that may be almost impossible to be used for parameter and state estimation problems. This model is then used as a nonlinear case, and it is considered in Chapter 7.

There were two possibilities in choosing the model parameters to be determined with the available measurements data – that of *hydrogen ionic fraction in liquid* as presented. The model parameters could be estimated using the *sodium in the resin* (y_{Na^+}) or by using *hydrogen in the liquid* (x_{H^+}). There are two possible scenarios from these options to use: 1) *the model based on measured hydrogen in resin*, or 2) *the model based on measured hydrogen in liquid*.

5.5.2. Model based on measured hydrogen in the resin

For a model where the solution is based on sodium in resin with available data as hydrogen in the resin, the ionic fraction of the two counter ions from Equation (5.26) will be presented by

$$\underset{(\text{resin})}{y_{Na}} = 1 - \underset{(\text{resin})}{y_H} \tag{5.22}$$

where

y_H – is the available measured data, hydrogen content in the resin,

$y_H \overset{1}{\underset{0}{\downarrow}}$ – means hydrogen ionic fraction in resin is decreasing from the top to the bottom of the column; the arrow indicates the direction of resin flow and

$y_{Na} \overset{0}{\underset{1}{\downarrow}}$ – is the sodium ionic fraction in resin and it is increasing from the top to the bottom of the column.

This presentation is possible since the hydrogen in resin phase is a *known variable*. If the sodium Na in the resin y_{Na} is a variable of interest, then the state space variable in the model has to be y_H. It will be considered as the measured variable. The substitution has to be $y_{Na} = 1 - y_H$, if the existing model is written for y_{Na}. y_{Na} and y_H does not refer to the process output in this case, but represent the ionic fractions of Na^+ and H^+ in the resin.

5.5.3. Model based on measured hydrogen in the liquid

This model is based on the sodium content $x_{Na^+}(t)$ of liquid phase. The dynamic behaviour of this variable in the process can be presented by Figure 5.16. Based on the fact that the hydrogen ionic fraction is the available measurements data, the model solution must contain x_{Na^+} presented in the form of x_{H^+} or solved using the Equation (5.25) and this brings about two possible solutions as discussed below,

$$\underset{(\text{liquid})}{x_{Na}} = 1 - \underset{(\text{liquid})}{x_H} \tag{5.23}$$

where

x_{Na} – is the sodium ionic fraction in the liquid,

$x_{Na} \overset{0}{\underset{1}{\uparrow}}$ – sodium ionic fraction is decreasing through the column going from the bottom to the top, the arrow indicates the direction of liquid flow,

$x_H \overset{1}{\underset{0}{\uparrow}}$ – is the hydrogen ionic fraction and its ionic content increases going up the column from the bottom to the top.

x_H – is the hydrogen ionic fraction in the liquid obtained from measured data.

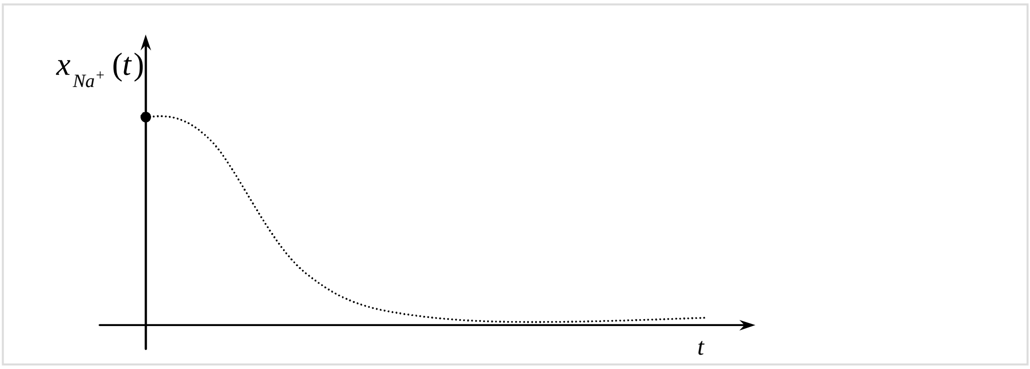


Figure 5.16. **Dynamic behaviour of sodium in the liquid phase throughout the process column stages**

Using the linear relationship between the counter ions given by Equation (5.10), the model equation based on measured x_H can be expressed as Equation (5.13). Alternatively, another possibility to use one ionic fraction equation to determine the unknown cation in the model based on the available measured data.

From the main model equation based on known data measurements that of *hydrogen in the liquid* (x_{H^+}), Equation (10.15), the solution for the model can be presented as follows:

$$\frac{d[1-x_{n,H}(t)]}{dt} = \frac{F_L}{(H+a_n h)}\left[1-x_{(n-1),H}(t)\right] - \\ -\frac{F_L}{(H+a_n h)}\left[1-x_{n,H}(t)\right] + \frac{a_{(n+1)}}{(H+a_n h)}F_R\left[1-x_{(n+1),H}(t)\right] - \\ -\frac{a_n}{(H+a_n h)}F_R\left[1-x_{n,H}(t)\right] + \frac{b_{(n+1)}-b_n}{(H+a_n h)}F_R \tag{5.24}$$

The final equation can be expressed as, Equation (5.25). This equation is used to with the measured data to calculate the unknown parameters and or states.

$$\frac{dx_n(t)}{dt} = \frac{F_L}{(H + a_n h)} x_{(n-1)}(t) - \frac{F_L}{(H + a_n h)} x_n(t) + \\ + \frac{a_{(n+1)}}{(H + a_n h)} F_R x_{(n+1)}(t) - \frac{a_n}{(H + a_n h)} F_R x_n(t) - \\ - \left[\frac{a_{(n+1)} - a_n + b_{(n+1)} - b_n}{(H + a_n h)} \right] F_R \tag{5.25}$$

The unknown variables are the parameters a_n and b_n.for $n = \overline{1, N}$. The summary of all model equations presented in this chapter is given in Table 5.13 below.

5.6. Parameter estimation problem formulation based on the process ionic fraction content

Based on data formulation in the previous section, two cases can be considered for parameter estimation, that is, 1) problem based on *model's ionic fraction linearization coefficients* and 2) a problem based on the process model's *separation factor*.

Case 1: Linear model

Parameter estimation problem is formulated based on linear connection between the ionic fractions in the liquid and resins. The model contains a total of twenty three parameters to be estimated.

Case 2: Nonlinear model

In this case the parameter estimation problem is formulated based on the process separation factor. In this case, a total of only six parameters are to be estimated. Two types of estimation methods are considered based on the two cases above, the linear parameter estimation case and the nonlinear parameter estimation case. The linear case has three possible methods of solution and these are covered in Chapter 6. The nonlinear case is considered as a separate nonlinear parameter estimation problem, and it is covered in Chapter 7.

The linear parameter estimation problem is solved by three methods, 1) the *direct method* that considers the *state vector*, 2) the *optimization problem* solved by a *gradient method that considers measured data* (the output only and the full state vector), and 3) the *Lagrange method that considers state vector measurements*. The nonlinear parameter estimation problem is solved by considering the state vector measurement using the Lagrange method.

Table 5.13: **Summary of the model types that can be used for the Continuous Countercurrent Ion Exchange Process (CCIX) using the relationship between the liquid and resin phases**

Model Type	Model
General continuous countercurrent ion exchange (CCIX) model showing liquid and resin relationship in each stage:	$H\frac{dx_n(t)}{dt}+h\frac{dy_n(t)}{dt}=F_L(x_{(n-1)}(t)-x_n(t))+F_R(y_{(n-1)}(t)-y_n(t))\,,\ n=\overline{1,N}$ (5.8)
Linear model relationship between liquid and resin defined in terms of liquid content:	$\underset{(liquid)}{x_n(t)}=\bar{a}_n\underset{(resin)}{y_n(t)}+\bar{b}_n$ (5.9)
Linear model relationship between liquid and resin defined in terms of resin content:	$\underset{(resin)}{y_n(t)}=a_n\underset{(liquid)}{x_n(t)}+b_n$ (5.10)
Model representing process concentration in the liquid phase obtained from Equations (5.8) and (5.10):	$\frac{dx_n(t)}{dt}=\frac{F_L}{(H+a_nh)}x_{(n-1)}(t)-\frac{F_L}{(H+a_nh)}x_n(t)+ +\frac{a_{(n+1)}}{(H+a_nh)}x_{(n+1)}(t)F_R(t)-\frac{a_n}{(H+a_nh)}x_n(t)F_R(t)+\frac{b_{(n+1)}-b_n}{(H+a_nh)}F_R\,,\ n=\overline{1,N}$ (5.13)
Model representing process concentration in the resin phase:	$\frac{dy_n(t)}{dt}=\frac{\bar{a}_{(n-1)}}{(\bar{a}_nH+h)}F_Ly_{(n-1)}(t)-\frac{\bar{a}_n}{(\bar{a}_nH+h)}F_Ly_n(t)+ +\frac{1}{(\bar{a}_nH+h)}y_{(n+1)}(t)F_R(t)-\frac{1}{(\bar{a}_nH+h)}y_n(t)F_R(t)-\frac{\bar{b}_n+\bar{b}_{(n-1)}}{(\bar{a}_nH+h)}F_R(t)\,,n=\overline{1,N}$ (5.15)

Model of separation factor for sodium ions in the solution:	$\alpha_H^{Na} = \frac{y_H x_{Na}}{y_{Na} x_H} = \frac{x_{Na}(1-y_{Na})}{y_{Na}(1-x_{Na})}$	(5.18)
Model for separation factor for hydrogen ions in the solution:	$\alpha_{Na}^{H} = \frac{y_H x_{Na}}{y_{Na} x_H} = \frac{y_H(1-x_H)}{x_H(1-y_H)}$	(5.19)
Sodium ionic fraction model in relation to hydrogen ionic fraction:	$y_{Na} = \frac{x_{Na}(1-y_{Na})}{\alpha_H^{Na}(1-x_{Na})}$ $y_{Na} = \frac{x_{Na}}{\alpha_H^{Na} - x_{Na}\alpha_H^{Na} + x_{Na}}$	(5.21) (5.22)
Model based on measured hydrogen in the resin:	$\underset{(\text{resin})}{y_{Na}} = 1 - \underset{(\text{resin})}{y_H}$	(5.23)
Model based on measured sodium in the liquid:	$\underset{(\text{liquid})}{x_{Na}} = 1 - \underset{(\text{liquid})}{x_H}$	(5.24)
CCIX model based measured sodium in liquid:	$\frac{dx_n(t)}{dt} = \frac{F_L}{(H+a_n h)} x_{(n-1)}(t) - \frac{F_L}{(H+a_n h)} x_n(t) + + \frac{a_{(n+1)}}{(H+a_n h)} F_R x_{(n+1)}(t) - \frac{a_n}{(H+a_n h)} F_R x_n(t) - -\left[\frac{a_{(n+1)} - a_n + b_{(n+1)} - b_n}{(H+a_n h)}\right] F_R$	(5.25)

5.7. Conclusion

Data collected from the University of Cape Town, Department of Chemical Engineering project used to validate the performance the methods that were developed in the thesis. The collected data are also used to test the model's validity. Trajectories of sodium concentration that the model is based have been generated using MATLAB software programming and presented. Techniques developed for solving parameter estimation and state estimation problems will be tested against the presented real data.

Model development based on ionic fraction content of the CCIX process columns has been presented. Both the sodium, Na^+ and hydrogen H^+ contents were presented. Data from the UCT project were presented as that of: 1) sodium in the resin (y_{Na^+}), and 2) hydrogen in liquid (x_{H^+}). From these presentations a suitable model more convenient for the thesis' hypothesis has been developed to solve the problem of parameter and state estimation. Two cases have been considered, 1) ionic fraction linearizing coefficients based model and 2) separation factor based model. The first case has four possible methods of solution, the *direct method* solution, the *optimization problem solution based on measured output*, the *optimization problem solution based on full state vector*, and the *state vector measurements based solution using the Lagrange method*. The second case is considered as a separate problem, that of nonlinear parameter estimation and it is solved using MATLAB software program; and the formulation of these problems has been presented.

The next chapter, Chapter 6 considers the first case of the parameter estimation problem solutions; the model linear coefficients based estimation.

CHAPTER SIX

METHOD FOR PARAMETER ESTIMATION (BILINEAR MODEL WITH AFFINE PARAMETERS)

CHAPTER SIX

6. METHOD FOR PARAMETER ESTIMATION (BILINEAR MODEL WITH AFFINE PARAMETERS)

6.1. Introduction

The developed bilinear model for the continuous countercurrent ion exchange (CCIX) process in Chapter 4 allows for the usage of linear techniques in that the model is linear according to its parameters. The first case of linear parameter estimation methods as proposed in Chapter 5 is considered in this chapter.

This case has a number of possible solutions based on the proposed methods of parameter estimation techniques. There are four suggested methods: 1) a gradient method that considers the process output measurements only, 2) a gradient method that considers the full-order state observation); 3) the direct solution using the state vector measurements, and 4) the Lagrange optimization procedure based on state vector measurements. Each of these methods is implemented using MATLAB software and results from simulations are produced, presented and discussed under each subsection of the chapter.

The chapter covers the model reformulation for parameter estimation in discrete time; and the procedures, algorithms and results for the linear model parameter estimation based on all methods mentioned. Concluding remarks are then provided, first for each method and then for all four methods.

6.2. Reformulation of the process model to be linear towards the unknown parameters

From the standard representation of the continuous countercurrent ion exchange (CCIX) process model developed in Chapter 4,

$$\begin{aligned}\dot{x}_n(t) = \frac{F_L}{(H+a_n h)}x_{n-1}(t) - \frac{F_L}{(H+a_n h)}x_n(t) - \frac{a_n}{(H+a_n h)}x_n(t)F_R(t) + \\ , \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad , \quad n=\overline{1,N} \\ + \frac{a_{n+1}}{(H+a_n h)}x_{n+1}(t)F_R(t) + \frac{b_{n+1}-b_n}{(H+a_n h)}F_R(t)\end{aligned} \tag{6.1}$$

the following model equations can be generated:

$$\dot{x}_1(t) = \frac{F_L}{H+a_1h}x_f(t) - \frac{F_L}{H+a_1h}x_1(t) + \frac{a_2}{H+a_1h}x_2(t)F_R(t) -$$
$$- \frac{a_1}{H+a_1h}x_1(t)F_R(t) + \frac{(b_2-b_1)}{H+a_1h}F_R(t)$$
$$\dot{x}_2(t) = \frac{F_L}{H+a_2h}x_1(t) - \frac{F_L}{H+a_2h}x_2(t) + \frac{a_3}{H+a_2h}x_3(t)F_R(t) -$$
$$- \frac{a_2}{H+a_2h}x_2(t)F_R(t) + \frac{(b_3-b_2)}{H+a_2h}$$
$$\vdots \qquad = \qquad \cdots \qquad \cdots \qquad \cdots$$
$$\dot{x}_n(t) = \frac{F_L}{(H+a_nh)}x_{n-1}(t) - \frac{F_L}{(H+a_nh)}x_n(t) - \frac{a_n}{(H+a_nh)}x_n(t)F_R(t) +$$
$$+ \frac{a_{n+1}}{(H+a_nh)}x_{n+1}(t)F_R(t) + \frac{b_{n+1}-b_n}{(H+a_nh)}F_R(t)$$
$$\cdots \qquad = \qquad \cdots \qquad \cdots \qquad \cdots$$
$$\dot{x}_N(t) = \frac{F_L}{H+a_Nh}x_{N-1}(t) - \frac{F_L}{H+a_Nh}x_N(t) + \frac{a_{N+1}}{H+a_Nh}x_{N+1}(t)F_R(t) -$$
$$- \frac{a_N}{H+a_Nh}x_N(t)F_R(t) + \frac{b_{N+1}-b_N}{H+a_Nh}$$

(6.2)

where $x_{n-1}(t) = w(t)$ for $n = 1$ is the disturbance input variable.

Using the above representation the following parameter notations are introduced

$$\frac{F_L}{H+a_1h} = l_1,\ \frac{F_L}{H+a_2h} = l_2,\ \cdots\frac{F_L}{H+a_nh} = l_n,\ \cdots\frac{F_L}{H+a_Nh} = l_N$$

$$\frac{a_2}{H+a_1h} = m_{12},\ \frac{a_3}{H+a_2h} = m_{23},\ \cdots\frac{a_{n+1}}{H+a_nh} = m_{n(n+1)},\ \cdots\frac{a_{N+1}}{H+a_Nh} = 0$$

$$\frac{a_1}{H+a_1h} = m_1,\ \frac{a_2}{H+a_2h} = m_2,\ \cdots\frac{a_n}{H+a_nh} = m_n,\ \cdots\frac{a_N}{H+a_Nh} = m_N$$

$$\frac{(b_2-b_1)}{H+a_1h} = k_1, \frac{(b_3-b_2)}{H+a_2h} = k_2,\ \cdots\frac{(b_{n+1}-b_n)}{H+a_nh} = k_n,\ \cdots\frac{(b_{N+1}-b_N)}{H+a_Nh} = \frac{-b_N}{H+a_Nh} = k_N$$

(6.3)

In common notation the above model parameters can be expressed by

$$\frac{F_L}{H+a_nh} = l_n,\ \frac{a_{n+1}}{H+a_nh} = m_{n,n+1},\ \frac{a_n}{H+a_nh} = m_n,\ \frac{(b_{n+1}-b_n)}{H+a_nh} = k_n,\ n = \overline{1,N}$$

(6.4).

The model Equations (6.2) can be generalized as follows:

$$\dot{x}_1(t) = l_1x_f(t) - l_1x_1(t) + m_{12}x_2(t)F_R(t) - m_1x_1(t)F_R(t) + k_1F_R(t)$$

$$\dot{x}_2(t) = l_2x_1(t) - l_2x_2(t) + m_{23}x_3(t)F_R(t) - m_2x_2F_R(t) + k_2F_R(t)$$

$$\cdots \qquad \cdots \qquad \cdots \qquad \cdots$$

$$\dot{x}_n(t) = l_n x_{n-1}(t) - l_n x_n(t) + m_{n,n+1} x_{n+1} F_R(t) - m_n x_n(t) F_R(t) + k_n F_R(t)$$

$$\cdots \qquad \cdots \qquad \cdots \qquad \cdots$$

$$x_N(t) = l_N x_{N-1}(t) - l_N x_N(t) + 0 - m_N x_N(t) + k_N F_R(t) \tag{6.5}$$

The output equation is determined by

$$\begin{aligned} z(t) &= C x_N(t) = \\ &= \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_1(t) & x_2(t) & \dots & x_n(t) & \dots & x_N(t) \end{bmatrix}^T = x_N(t) \end{aligned} \tag{6.6}$$

The state variables are the mole fractions of Na (sodium) in the liquid phase, determined by $\begin{bmatrix} x_1(t) & x_2(t) & \cdots & x_n(t) & \cdots & x_N(t) \end{bmatrix}$, the input variable is the flow rate of resin in the column $F_R(t)$, but the process real input considered is the length of the up-flow period T, which is calculated from the mass balance using the volume of the resin in each stage, and it is given by $F_R(t)T = hd$, where d is the fraction of the resin hold up which is moved from one stage to another and $d = 2/3$, h is the resin holdup, calculated or given, H is the liquid holdup, calculated or given. The output variable is given by the mole fraction of Na (sodium) in the liquid of the last stage, $x_N(t)$, Equation (6.6). The disturbance variable is the mole fraction of Na in the liquid solution of the feed water, $x_f(t) = x_{n-1}(t)$ for $n = 1$.

The parameters introduced above represent mathematical expression between known and unknown constants in the process. The known ones are H, h and F_L. The unknown ones are a_n and b_n, $n = \overline{1, N}$, which are introduced by the linear relationship between ionic fractions of the sodium and hydrogen. If the parameter estimation problem considers only a_n and b_n, then the parameters a_n make the problem nonlinear according to the parameters. If the mathematical expressions in Equation (6.3) are considered as parameters then the parameter estimation model becomes linear according to parameters. This allows faster solution of the parameter estimation problem.

6.3. Discrete time model formulation for use with experimental data for parameter estimation

It is necessary to develop a discrete time model for use with the experimental data. Based on the continuous time model, Equations (6.2), (6.5) and (6.6) a corresponding discrete time model can be developed using Euler equation

$\frac{dx(t)}{dt} = \frac{x(k+1) - x(k)}{\Delta t}$, where Δt is the sampling period. The corresponding model difference equations are presented by

$$
\begin{aligned}
x_1(k+1) &= x_1(k) + \Delta t\left[l_1 x_f(k) - l_1 x_1(k) + m_{12} x_2(k) F_R(k) - m_1 x_1(k) F_R(k) + k_1 F_R(k)\right] \\
x_2(k+1) &= x_2(k) + \Delta t\left[l_2 x_1(k) - l_2 x_2(k) + m_{23} x_3(k) F_R(k) - m_2 x_2(k) F_R(k) + k_2 F_R(k)\right] \\
\cdots &= \qquad \cdots \qquad \cdots \qquad \cdots \\
x_n(k+1) &= x_n(k) + \Delta t\left[l_n x_{n-1}(k) - l_n x_n(k) + m_{n,n+1} x_{n+1}(k) - m_n x_n(k) F_R(k) + k_n F_R(k)\right] \\
\cdots &= \qquad \cdots \qquad \cdots \qquad \cdots \\
x_N(k+1) &= x_N(k) + \Delta t\left[l_N x_{N-1}(k) - l_N x_N(k) - m_N x_N(k) F_R(k) + k_N F_R(k)\right] \\
z(k) &= C x_N(k) \qquad (6.7)
\end{aligned}
$$

The developed CCIX process model is bilinear according to the control and the state but linear according to the parameters l_n, $m_{n,n+1}$, m_n and k_n for $n = \overline{1, N}$ as determined by Equation (6.5) or Equation (6.7). Every equation is characterized by four parameters and there are a total of six equations but the common number of parameters is twenty three due to the parameter $m_{N,N+1}$ being equal to zero.

In order to estimate values of the unknown parameters, experimental data for input, states and output measurements is required and is sourced from the previous experiments performed in Hendry (1982a-d). Experimental data is considered for two distinct possibilities; the case where the states (concentration in each stage) are measurable and the case where only input and output data is measurable. This gives rise to four possible solutions as suggested in Section 6.1. Each of these methods is considered in the next sections.

6.4. Linear model parameter estimation using the measured data of the process output

Based on the process model as formulated in Chapter 4 and using the Equations (6.2), (6.4) and (6.7), the following re-parameterized model structure has been generated for determining unknown parameters (for better notation of the parameters), for the considered process data, where:

$$l_1, m_{12}, m_1, k_1, l_2, m_{22}, m_2, k_2, \cdots, = \left[p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, ..., p_{23}\right], \quad (6.8)$$

and for the general case

$$l_1, m_{12}, m_1, k_1, ... = \left[p_1, p_2, ..., p_m, ..., p_q\right] \qquad (6.9).$$

$$
\begin{aligned}
x_1(k+1) &= x_1(k) + p_1\Delta t x_f(k) - p_1\Delta t x_1(k) + p_2\Delta t x_2(k)F_R(k) - \\
&\qquad - p_3\Delta t x_1(k)F_R(k) + p_4\Delta t F_R(k) \\
x_2(k+1) &= x_2(k) + p_5\Delta t x_1(k) - p_5\Delta t x_2(k) + p_6\Delta t x_3(k)F_R(k) - \\
&\qquad - p_7\Delta t x_2(k)F_R(k) + p_8\Delta t F_R(k) \\
\cdots &= \qquad \cdots \qquad\qquad \cdots \qquad\qquad \cdots \\
x_n(k+1) &= x_n(k) + p_m\Delta t x_{n-1}(k) - p_m\Delta t x_n(k) + p_{m+1}\Delta t x_{n+1}(k)F_R(k) - \\
&\qquad - p_{m+2}\Delta t x_n(k)F_R(k) + p_{m+3}\Delta t F_R(k) \\
\cdots &= \qquad \cdots \qquad\qquad \cdots \qquad\qquad \cdots \\
x_N(k+1) &= x_N(k) + p_{q-3}\Delta t x_N(k) - p_{q-3}\Delta t x_N(k)F_R(k) + \\
&\qquad + p_{q-2}\Delta t x_{N+1}(k)F_R(k) - p_{q-1}\Delta t x_N(k)F_R(k) + p_q\Delta t F_R(k) \\
z(k) &= Cx_N(k)
\end{aligned}
\tag{6.10}
$$

The method of determining the parameter values using the process output data has only one possible solution based on the fact that the method considers the case where only the input and output are measurable and the states are unknown. The aim is to find the values of the parameters $p_1, p_2, \ldots p_m, p_{m+1}, \ldots p_q$ such that the error between the measured output values, $\bar{z}(k) = C\bar{x}_N(k)$ and the model output values, $z(k) = Cx_N(k)$ is minimized under the constraints of the model for an expressed number of points $k = \overline{0, K}$ in time which becomes the trajectory period for comparison.

According to this requirement, the error difference needs to be expressed as a computational criterion over the trajectory period. The chosen technique uses the least squares method to determine the error between *measured data* and the model *estimated output* from the estimated set of parameter values as shown in Equation (6.11) below, starting at some initial set of parameter values.

$$
J(p) = \sum_{k=1}^{K} \left\| \bar{z}(k) - z(k) \right\|^2 = \sum_{k=1}^{K} \left\| C\bar{x}_N(k) - Cx_N(k) \right\|^2 = \sum_{k=1}^{K} \left\| e(k) \right\|^2 \tag{6.11}
$$

where

$e(k) = \bar{z}(k) - z(k)$ – is the error difference between the measured output and the model output values,

$C \in R^{1\times N}$ – is the output vector with N dimension depending on the number of states involved,

N – is the total number of stages in the process model,

$\bar{x}_N$, x_N – are the last states of the real process and that of the model respectively.

The proposed procedure is an iterative method that searches for the minimum error between the output values obtained using the current estimates of the parameter vector versus the output values as determined from the measured data. If the parameter set does not agree a step increment of each parameter value in the set is made to improve the set until it agrees, alternatively the stopping procedure is reached, and the calculation is stopped.

The criterion gives the best strategy for fitting the trajectory of the model to that of the real process output. One of the main advantages of this criterion is to give a unique trajectory for every given set of data. This procedure can be represented by the structure in Figure 6.1 (Li & Ding, 2013; Li, Mu & Zuo, 2016; Olsson & Newell, 1999; Walter & Pronzato, 1995:113).

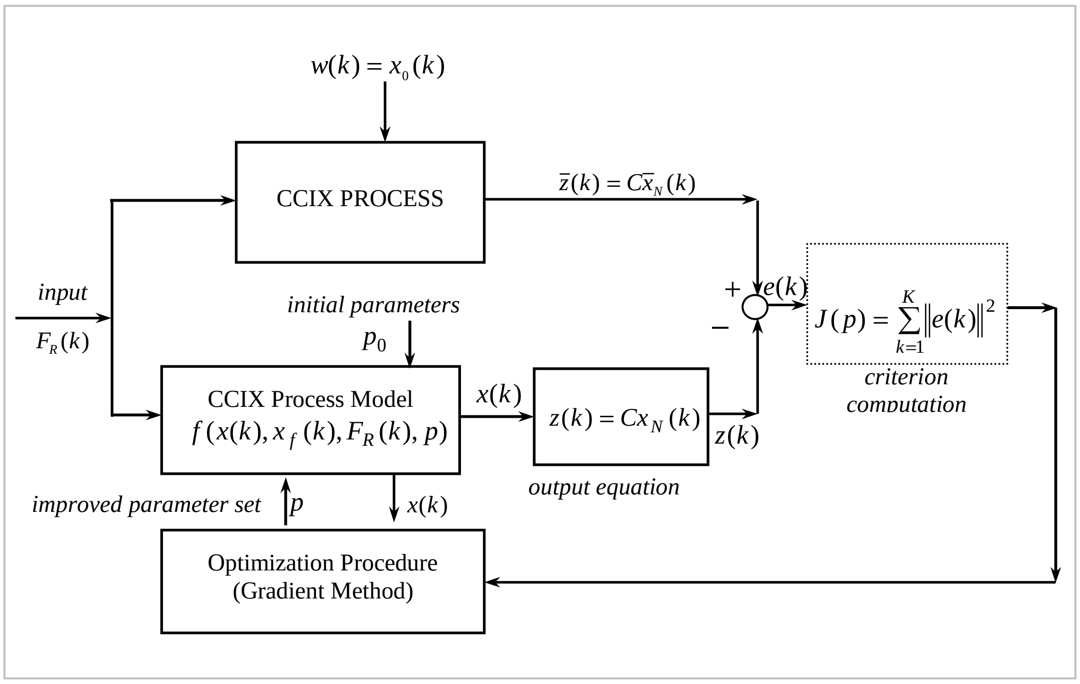


Figure 6.1. **Parameter estimation procedure using optimization techniques based on the output of measured plant data**

(Adapted from Olsson & Newell, 1999; Walter & Pronzato, 1995:113)

The idea is that the solution must give the minimum of the criterion in Equation (6.10). The criterion is the function of the output of the model, and in the same fashion it is compared to that of the state $x(k)$, where $Cx_N(k) = z(k)$. But in each state space equation the state is the function of the parameter from the model Equations (6.1) and (6.5), or the criterion can be thought of as being the function of the parameters. If the error difference is smaller than a preselected small number, the current parameter values are considered suitable for the model.

In searching for a minimum for optimality of a criterion according to the necessary conditions for optimality, the first derivatives towards the parameters of the criterion must be equal to zero.

The gradient search method is used to solve the given optimization problem. This method considers the first derivatives of the criterion towards the parameters such that

$$\frac{\partial J(p)}{\partial p_i} = 0 \qquad i = 1, 2, \ldots, q \tag{6.12}$$

where $p \in R^q$ is the vector of parameters with q dimension.

The vector of parameters does not appear explicitly in the criterion equation and therefore, it becomes necessary to use the criterion dependence shared through the output of the state space representation, Equations (6.1) and (6.2). This is achievable by using an algorithm that will permit the use of this dependency to minimize the criterion. The minimization is possible through different gradient methods for computation. In this case the fastest descent method is chosen due to its *simplicity* and *quick convergence* capabilities.

6.4.1. The procedure for solving the linear model parameter estimation using the measured data (the measurement output data) based on the method of fastest descent

The method of fastest descent used in solving the optimization problem Equations (6.9)–(6.10) is characterized by the simplest *iterative gradient procedure* to reach the required minimum through improving the initial estimates of the vector of the unknown parameters $p \in R^q$. The method can be used to minimize the error difference in a least squares optimization technique. This method is applied here to determine optimal parameter estimates based on the model Equation (6.10) and the optimization criterion, Equation (6.11).

The direction of the fastest descent is opposite to the direction of the gradient. At the initial point, the direction of the gradient coincides with the direction in which the criterion reduces in the fastest way for infinitely small changes of the vector of parameters. The direction of the fastest descent is given by the vector

$$p^{j+1} = p^j + dp \tag{6.13}$$

where

$dp = \left[dp_1, dp_2, \ldots, dp_q \right]$ is the direction of the descent and

p^{j+1} – is the improved value of the parameters,

p^j – is the previous value of the parameters, and

j – is the index of iteration used to determine the best parameter values.

The gradient of the criterion toward the i^{th} component of the vector of parameters p is given by $\frac{\partial J(p)}{\partial p_i}$. A step of the gradient procedure α, is required to determine the amount of change of the parameter value. The direction of the descent for each vector calculation is given by the equation:

$$dp_i = \frac{-\alpha\left[\frac{\partial J(p)}{\partial p_i}\right]}{\left[\sum_{i=1}^{q}\left[\frac{\partial J(p)}{\partial p_i}\right]^2\right]^{1/2}} \tag{6.14}$$

It is not possible to directly calculate the vector of gradients $\frac{\partial J(p)}{\partial p_i}$ by analytical methods using derivatives since the model is dynamical and the criterion is not the explicit function of the state and parameters. The approximation technique is therefore applied. The technique is based on the first principle of the gradient by introducing small perturbation Δ for every parameter, independently.

$$\frac{\partial J(p)}{\partial p_i} = \frac{J(p_1, p_2, ..., p_i + \Delta_p, p_{i+1}, ..., p_q) - J(p_1, p_2, ..., p_i, p_{i+1}, ..., p_q)}{\Delta_p},$$

$$i = 1,\ 2, ..., q \tag{6.15}$$

where

Δ_p – is a small deviation of the each component of the vector of parameters.

On the basis of the above mentioned considerations, the parameter estimation procedure can be formalized.

6.4.2. The algorithm for the calculation procedure of the linear model parameter estimation using the measured data (output based on available data)

1. The following process data values must be initialized:

F_L	the liquid flow rate entering the first stage of the load column,
F_R	the resin flow rate entering the top stage of the load column,
H	the liquid holdups per column diameter,
h	the resin holdups per column diameter,
a	ionic fraction of resin in a stage of the process column,

b	ionic fraction of liquid in a stage of the process column,
d	the resin/liquid ionic fraction ratio
N	total number of stages per column,
T	process upflow period,
Δt	sampling period.

2. The following initial parameters for the gradient method must be set:

x_0	initial values of the state vector,
M	very large number for terminating iterations if no optimal values are reached in time,
K	the number of discrete time steps in the optimization period,
α_θ	the step of the gradient procedure,
ε	the preselected very small number for determining the end of calculation,
j	the number of iterations for the gradient procedure,
$p^j = p_0$	the initial estimates of the parameter vector p,
q	the total number of parameters in the parameter vector, and
Δ_p	the value of deviation for each parameter as tested for optimality.

3. Generate the vector of parameters, in this case from Equations (6.3–6.5) and (6.8)
$$p_0 = \left[p_1, p_2, ..., p_m, p_{m+1}, ..., p_q\right]^T = \\ = \left[l_1, m_{12}, m_1, k_1, l_2, m_{23}, m_2, k_2, ..., l_n, m_{n,n+1}, m_n, k_n, ..., l_N, m_N, k_N\right]$$
For $N = 6$, the number of parameters is $q = 23$, $p \in R^q = p \in R^{23}$.
4. Solve the model parameters, Equation (6.10) using initial values for parameters p_0 and initial state values $x(k) = x_0$ as obtained from the measured data
$$x_n(k+1) = x_n(k) + p_m \Delta t x_{n-1}(k) - p_m \Delta t x_n(k) + p_{m+1} \Delta t x_{n+1}(k) F_R(k) - \\ - p_{m+2} \Delta t x_n(k) F_R(k) + p_{m+3} \Delta t F_R(k)$$
   4.1. Set the optimization period $k = \overline{1, K}$ and solve the model Equations (6.10) for the full trajectory of the optimization period,
   4.2. The solution is an $x \in R^{N \times K}$ matrix.
5. Calculate the value of the criterion based on the measured output data $\bar{z}(k)$ and obtained output solution determined by $z(k) = Cx_N(k) = x_N(k)$ from the

state values calculated in step 4., i.e., $J(p)=\sum_{k=1}^{K}\left\|e(k)\right\|^2=\sum_{k=1}^{K}\left\|\bar{z}(k)-z(k)\right\|^2$ for the full optimization period.

6. Initialize the gradient procedure, and $while \quad j<M$:
   6.1. Declare the $istates()$ function arguments, $states(k)$, $F_R(k)$, $states\_in(k)$, x_0, p, dt for full trajectory $k=\overline{1,K}$
   6.2. And then call the $istates()$ function to determine the current state values,
   6.3. Set the deviated value of the parameter for the i^{th} component starting with the first parameter value p_1, where $\Delta p_i^j = p_i^j + \Delta_p$,
   6.4. Solve the model equations (6.9) for every i^{th} component and separately for every p_i^j component,
   6.5. Calculate the deviated value of the criterion for the i^{th} component, $J(p_1, p_2, ..., \Delta p_i^j, p_{i+1}, ..., p_q$,
   6.6. Calculate the gradients $\dfrac{\partial J(p)}{\partial p_i}$ using the Equation (6.13) for every parameter, $i=\overline{1,q}$.
7. Calculation of the gradients for the fastest descent dp_i
   7.1. Calculate the weighted sum $S=\left[\sum_{k=1}^{K}\left(\dfrac{\partial J(p)}{\partial p_i^j}\right)^2\right]^{1/2}$,
   7.2. Calculate the direction of the gradients $dp_i^j=\dfrac{-\left[\alpha\dfrac{\partial J(p)}{\partial p_i^j}\right]}{S}$, $i=1,2,...,q$, Equation (6.11).
8. Calculate the (improved) estimates of the parameter vector using $p_i^{j+1}=p_i^j+dp_i^j$, $i=1,2,...,q$.
9. Determine the error difference between the previous parameters and the (improved) latest calculated parameters for termination of the calculation $e_{p_i}=\left|p_i^{j+1}-p_i^j\right|=\left|dp_i^j\right|$, $i=1,2,...,q$.
10. The criterion for termination of the calculation:

10.1. if the error difference is smaller than or equal to the preselected very small number, ε, i.e., $e_{p_i} \le \varepsilon$ for all parameters $i = 1,2,...,q$, the optimal parameters have been obtained, and the calculation is terminated,

10.2. if the error difference is still greater than the preselected very small number, $e_{p_i} = \left| dp_i^j \right| > \varepsilon$, the calculations are restarted from step 4 using the current parameters of the parameter vector p_i^j until the maximum number of iterations M is reached,

10.3. if at any point within the procedure, $j \ge M$, i.e., the procedure step has reached the maximum number of iterations required, the calculations must be stopped.

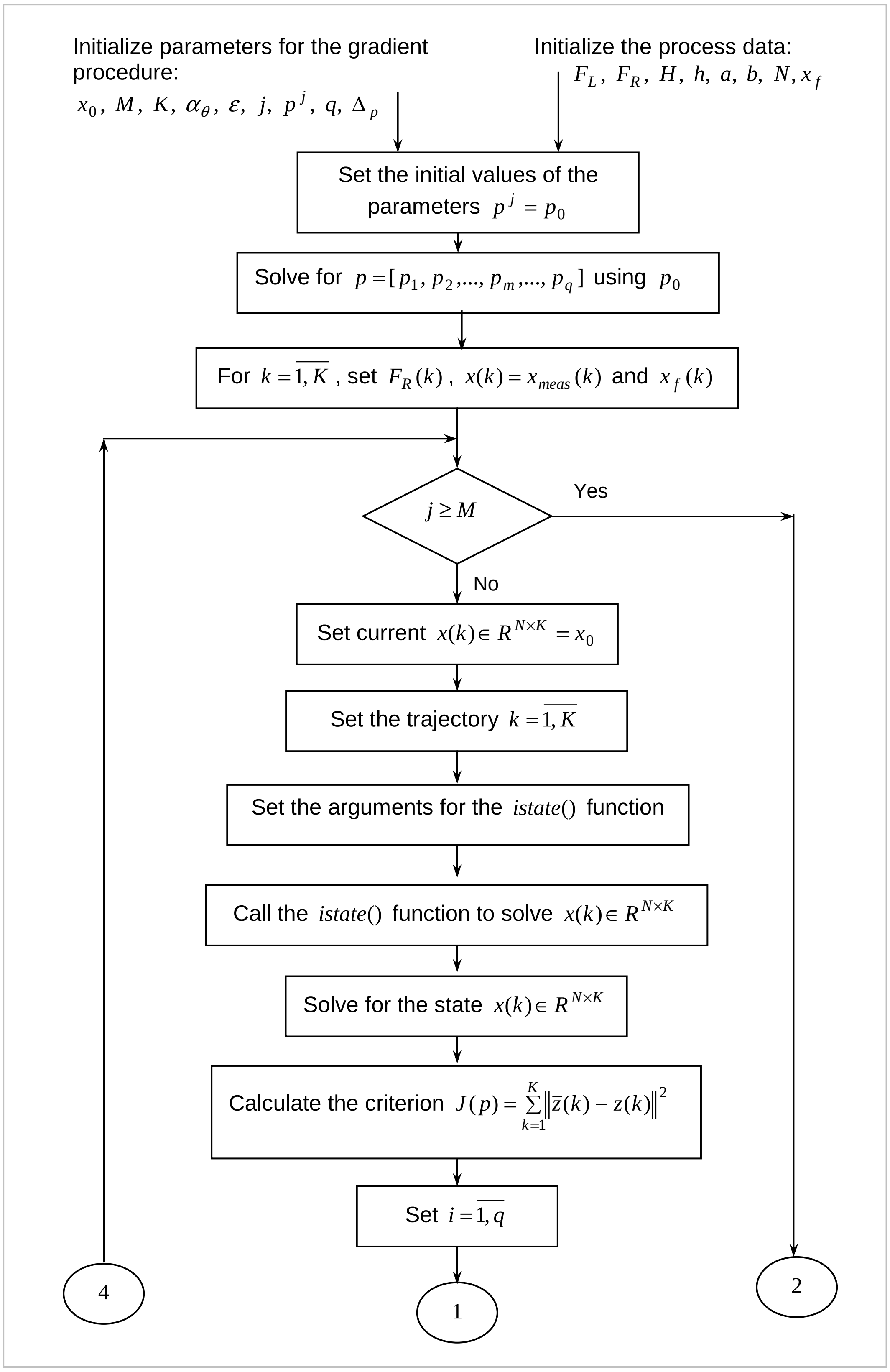
The flowchart of the output measurement-based parameter estimation algorithm is presented in Figure 6.2.

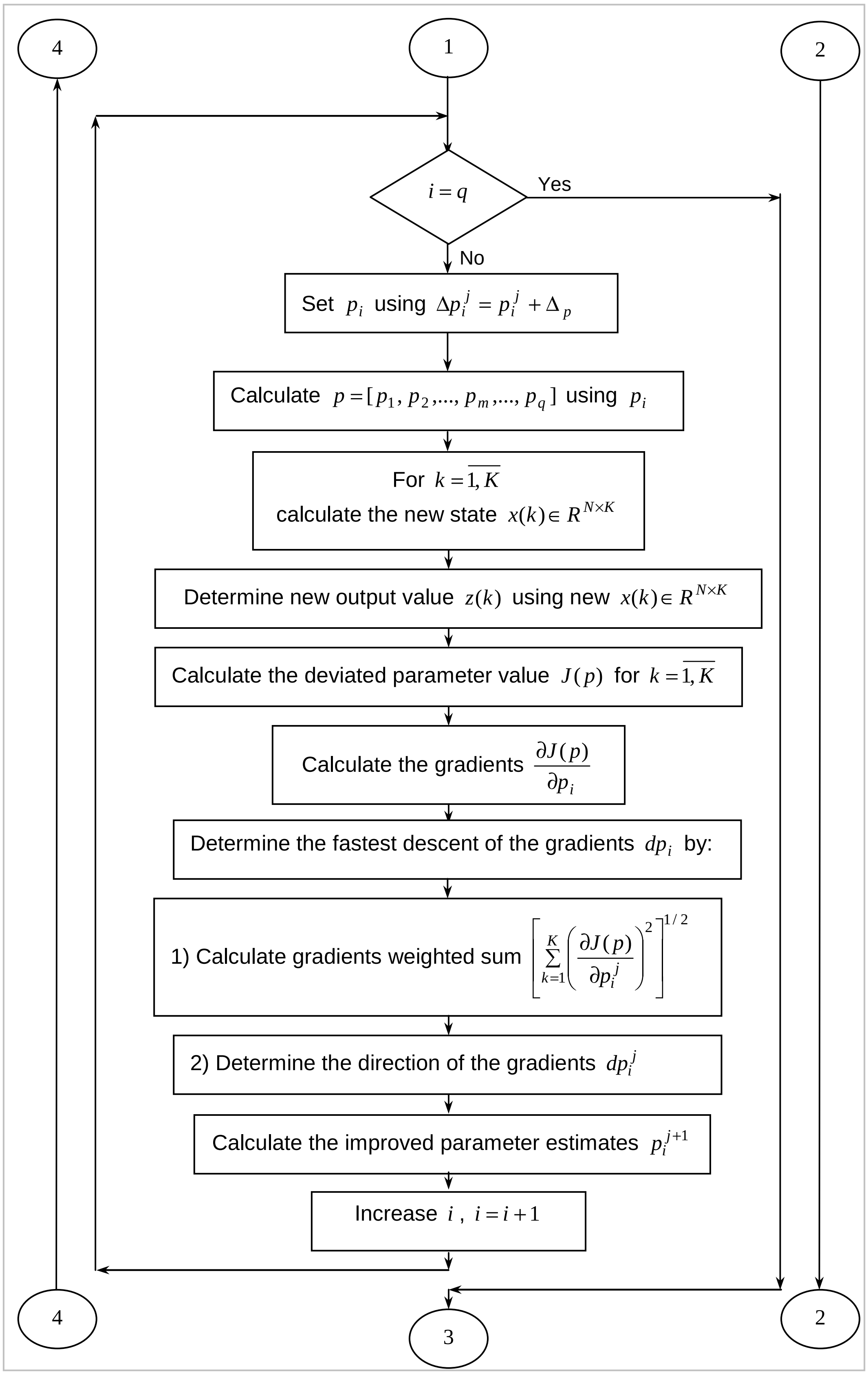
6.4.3. The experiments and results of the linear model parameter estimation problem using the process output (output measured data)

Experiments were run for the calculation of process dependencies on: the *sampling period* Δt, *model coefficient vectors* a and b, *process holdups* and *iteration gradient step* α. Once the best values of these parameters are obtained, they are then applied on analysing the process behaviour under *input disturbance changes* and *changes of initial model parameters*.

The strategy for investigation of Δt is to select its best value and apply it in the rest of the experiments. This best value of Δt is also used to identify the best model parameters based on a and b coefficients. The criterion for evaluating the best results is based on two parameters: the *least squares error* $J(p) = \sum_{k=1}^{K} e(k)e(k)^T$ and the *method's calculation run time* calculated from MATLAB processor time. In all experiments' results $x_n bar = \overline{x}_n$, $x_n^{\sim} = x_n(k)$ for $n = \overline{1,N}$, Equation (6.11).

6.4.4. The flowchart for implementing the procedure


Initialize parameters for the gradient procedure:
$x_0,\ M,\ K,\ \alpha_\theta,\ \varepsilon,\ j,\ p^j,\ q,\ \Delta_p$
Initialize the process data:
$F_L,\ F_R,\ H,\ h,\ a,\ b,\ N, x_f$
Set the initial values of the parameters $p^j = p_0$
Solve for $p = [p_1, p_2, ..., p_m, ..., p_q]$ using p_0
For $k = \overline{1, K}$, set $F_R(k)$, $x(k) = x_{meas}(k)$ and $x_f(k)$
$j \geq M$
Yes
No
Set current $x(k) \in R^{N \times K} = x_0$
Set the trajectory $k = \overline{1, K}$
Set the arguments for the $istate()$ function
Call the $istate()$ function to solve $x(k) \in R^{N \times K}$
Solve for the state $x(k) \in R^{N \times K}$
Calculate the criterion $J(p) = \sum_{k=1}^{K} \|\bar{z}(k) - z(k)\|^2$
Set $i = \overline{1, q}$
4
1
2


4
1
2
$i = q$
Yes
No
Set p_i using $\Delta p_i^j = p_i^j + \Delta_p$
Calculate $p = [p_1, p_2, ..., p_m, ..., p_q]$ using p_i
For $k = \overline{1, K}$
calculate the new state $x(k) \in R^{N \times K}$
Determine new output value $z(k)$ using new $x(k) \in R^{N \times K}$
Calculate the deviated parameter value $J(p)$ for $k = \overline{1, K}$
Calculate the gradients $\frac{\partial J(p)}{\partial p_i}$
Determine the fastest descent of the gradients dp_i by:
1) Calculate gradients weighted sum $\left[\sum_{k=1}^{K} \left(\frac{\partial J(p)}{\partial p_i^j} \right)^2 \right]^{1/2}$
2) Determine the direction of the gradients dp_i^j
Calculate the improved parameter estimates p_i^{j+1}
Increase i, $i = i + 1$
4
3
2

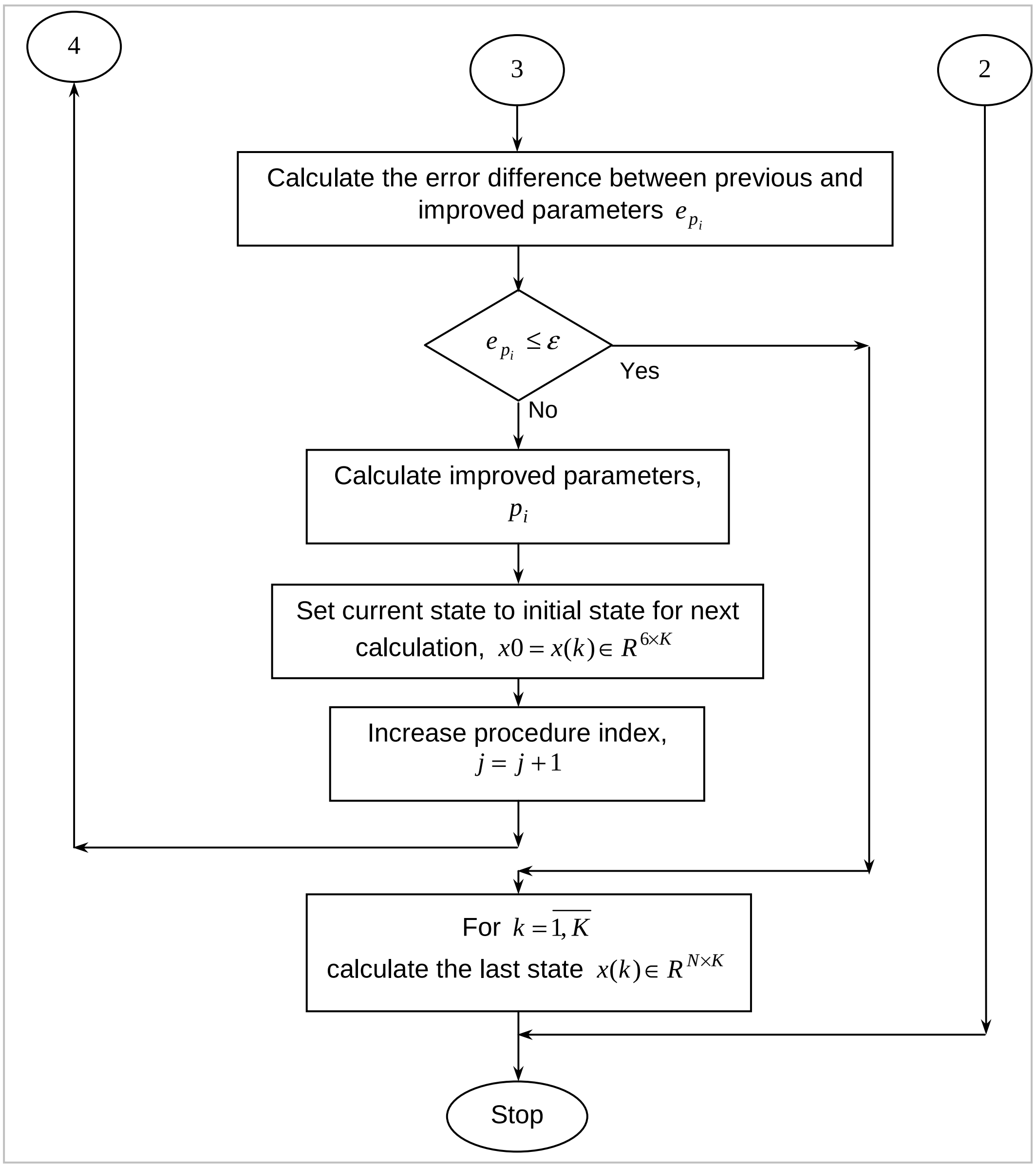

4
3
2
Calculate the error difference between previous and improved parameters e_{p_i}
$e_{p_i} \le \varepsilon$
Yes
No
Calculate improved parameters, p_i
Set current state to initial state for next calculation, $x0 = x(k) \in R^{6 \times K}$
Increase procedure index, $j = j + 1$
For $k = \overline{1, K}$
calculate the last state $x(k) \in R^{N \times K}$
Stop

Figure 6.2. **The procedure for linear model parameter estimation based on process output measurement**

The first set of experiments for the measured output method

The first set experiments and the corresponding sets of results are presented in Table 6.1 and Figures 6.3–6.8. The aim of this experiment is to evaluate the influence of the *sampling period* on the overall process behaviour. The initial sampling period is set to $\Delta t = 2.35\,\text{min}$

Table 6.1: First set of experiments showing the changing sampling period Δt

The changing sampling period Δt					
Process parameter (units)	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 32.93\ l$	✓	✓	✓	✓	✓
$H = 42.809\ l$	✓	✓	✓	✓	✓
$F_L = 2000/60\ (l/\min)$	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓
$a = [0.8\ 0.6\ 0.5\ 0.4\ 0.3\ 0.25]$	$a = 230 \times a$	✓	✓	✓	✓
$b = [0.01\ 0.02\ 0.05\ 0.08\ 0.1\ 0.2]$	$b = 60 \times b$	✓	✓	✓	✓
M	1000	2000	✓	✓	✓
dt, sampling period (Δt)	$\Delta t = 2.35 \min$	$\Delta t = \Delta t$	$\Delta t = \frac{1}{2}\Delta t$	$\Delta t = \frac{1}{4}\Delta t$	$\Delta t = \frac{1}{32}\Delta t$
Δ_p, parameter deviation	0.001	✓	✓	✓	✓
$grad$, iteration gradient step (α)	0.82	✓	✓	✓	✓
ε, epsilon (very small number)	0.0001	✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

The table shows the changing sampling period with all other process parameters fixed. The aim is to obtain the best sampling period and to apply it in the rest of the experiments that follow. The best sampling period refers to the sampling period that would produces the best estimation values (i.e., the estimated output tracks the real output very closesly).

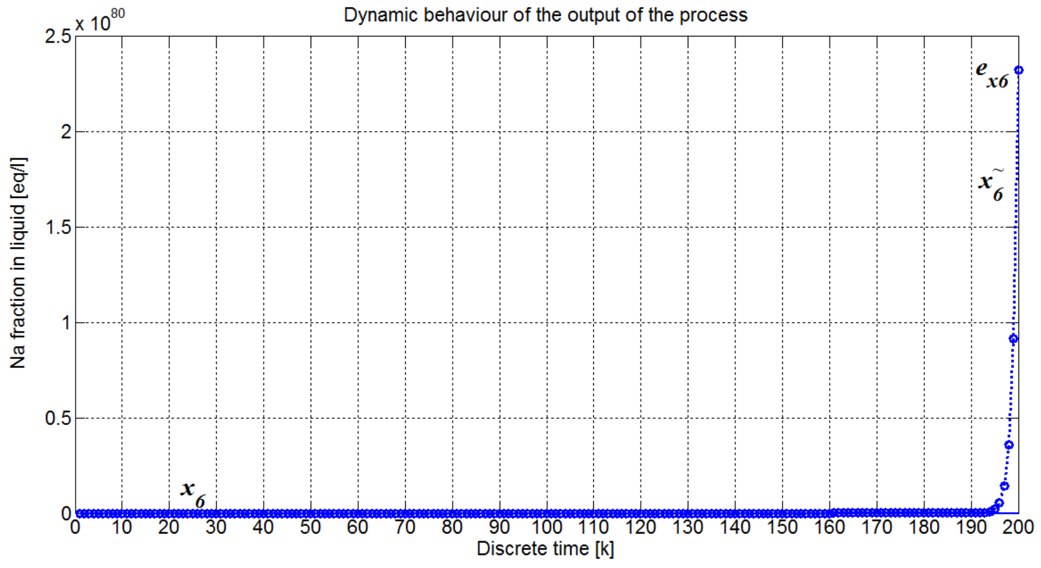


Figure 6.3. The measured and estimated output and the output error difference from the worse case results – Run 1 of Table 6.1

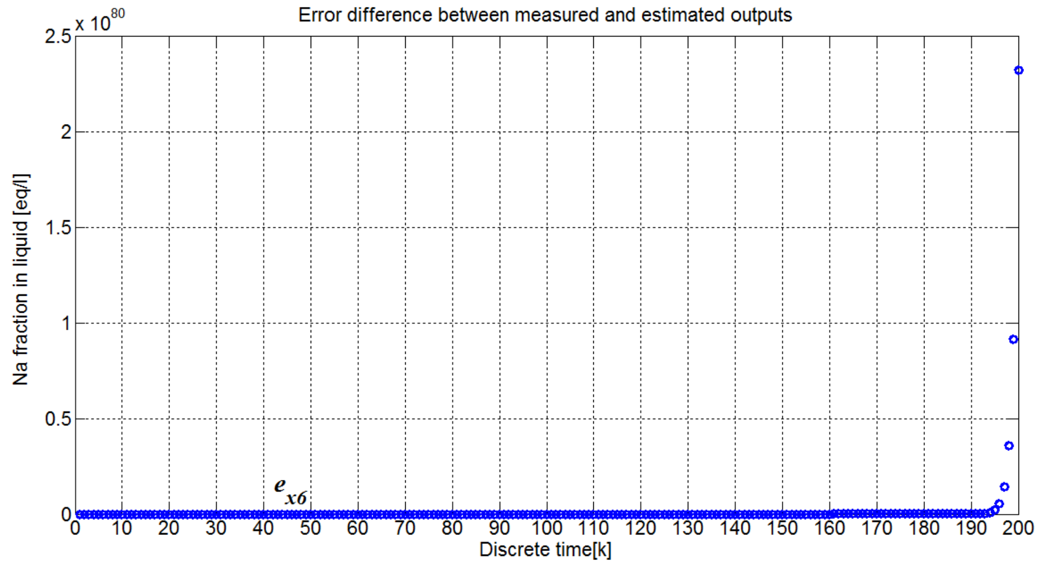


Figure 6.4. The output error signal in detail for the worst case results – Run 1 of Table 6.1

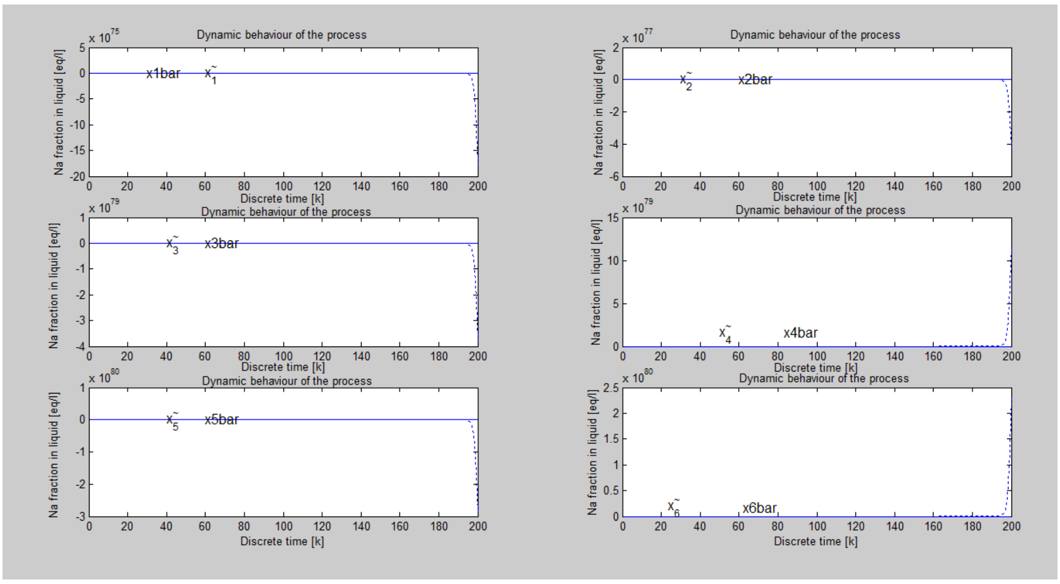


Figure 6.5. Model dynamic behaviour showing the real and estimated states of the worst case results – Run 1 Table 6.1

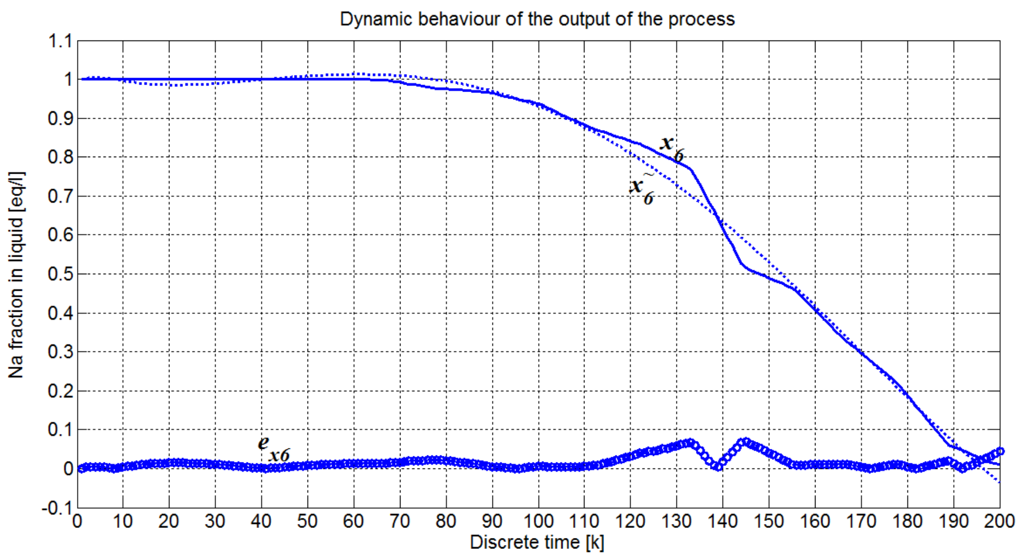


Figure 6.6. The measured output and estimated output and the output error for the best results – Run 5 of Table 6.1

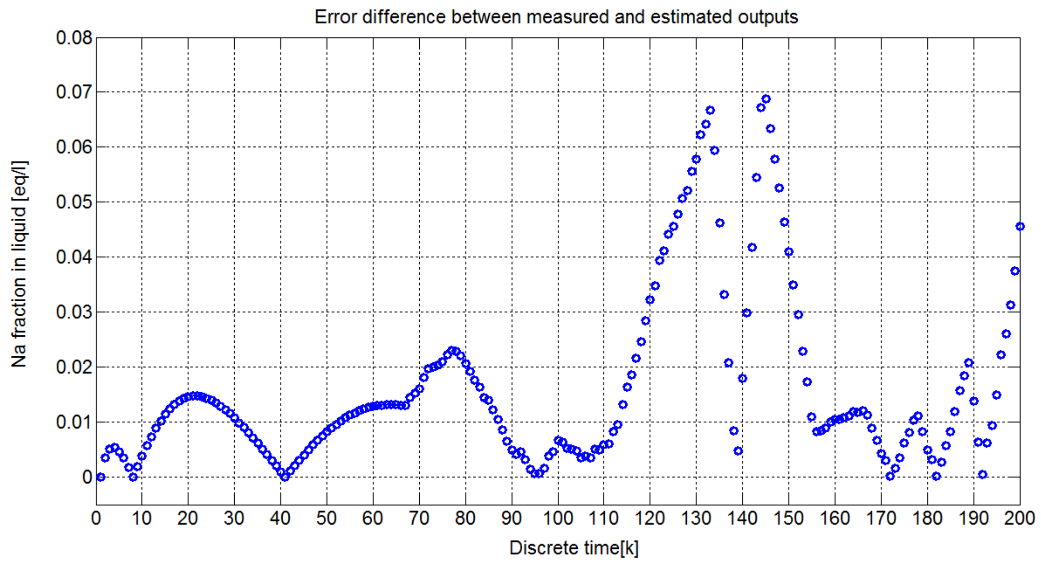


Figure 6.7. The output error signal in detail for the best results – Run 5 of Table 6.1

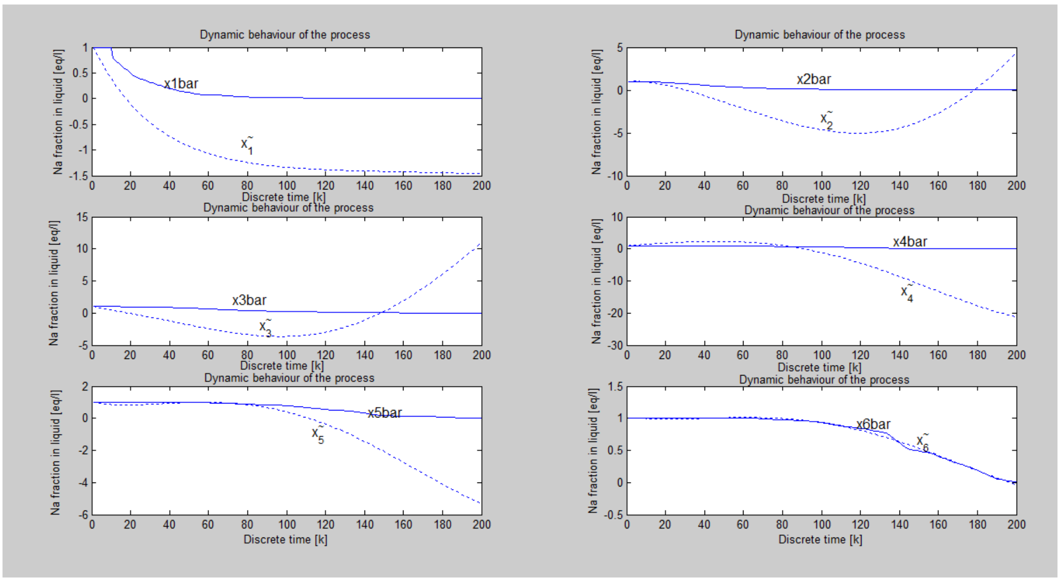


Figure 6.8. Model dynamic behaviour showing the real and estimated states for the best results – Run5 Table 6.1

Table 6.2: Results of first set of experiments for changing the sampling period Δt showing the error behaviour and calculated run time

Estimation criterion the least squares error and the estimation run time for the first set of experiments		
Case 1: (First set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	1.2787e+163	3.7900
Run 2	1.2787e+163	3.8060
Run 3	8.3037e+009	225.9510
Run 4	673.1209	230.6180
Run 5	20.2932	225.6700

Results and discussion of the first set of experiments of the measured output method

The first set of results from the changing sampling period, produced no estimates of the process output for the first three runs up to $\Delta t = \frac{1}{2}\Delta t$. From $\Delta t = \frac{1}{8}\Delta t$ the estimation results are acceptable though the error is relatively high. At $\Delta t = \frac{1}{32}\Delta t$, the error improves quite drastically. The results of this run are displayed in Figures 6.3–6.8, where x_6 is the measured output, $\tilde{x}_6$ is the estimated output and e_{x_6} is the error between the measured output and simulated output using the estimated parameters. Table 6.2 demonstrates the estimation criterion value and the processing time taken for each run. The process is faster for the poor error, this could be associated with reaching of the stopping procedure faster than in the improved error case. As the sampling period decreases, the error also becomes smaller.

The second set of experiments of the measured output method

The second set of experiments show different combinations of a and b model coefficients, Table 6.3. The aim of this experiment is to get the best combination between model coefficients and the best smallest number for stopping the estimation procedure. The original a and b coefficients are $a = [0.8\ 0.6\ 0.5\ 0.4\ 0.3\ 0.25]$ and $b = [0.01\ 0.02\ 0.5\ 0.08\ 0.1\ 0.20]$. The same vector coefficients have been used in the first sets of experiments. Results are presented in Figures 6.9–6.14. The vectors a and b are multiplied by the factors of 23 and 6 correspondingly, Table 6.3.

Table 6.3: **The second set of experiments to investigate the influence of a and b model coefficients with respect to stopping time**

Changing different sets of model coefficients					
Process parameter (units)	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 32.93\ l$	✓	✓	✓	✓	✓
$H = 42.809\ l$	✓	✓	✓	✓	✓
$F_L = 2000/60\,(l/\min)$	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓
$a = [0.8\ 0.6\ 0.5\ 0.4\ 0.3\ 0.25]$	$a = 23 \times a$	$a = 23 \times a$	$a = 230 \times a$	$a = 230 \times a$	$a = 60 \times a$
$b = [0.01\ 0.02\ 0.05\ 0.08\ 0.1\ 0.2]$	$b = 6 \times a$	$b = 6 \times a$	$b = 60 \times b$	$b = 60 \times b$	$b = 230 \times b$
M	2000	✓	✓	✓	✓
dt , sampling period (Δt)	0.012	✓	✓	✓	✓
Δ_p , parameter deviation	0.001	✓	✓	✓	✓
$grad$, iteration gradient step (α)	0.82	✓	✓	✓	✓
ε , epsilon (very small number)	0.001	0.0001	0.001	0.0001	0.0001

✓ means the value stays the same as in the previous mention in the table

Different sets of a and b coefficients are changed with respect to changing very small positive number ε for stopping the estimation procedure; the aim is to obtain the best combination of these two process parameters and used them for the rest of the experiments.

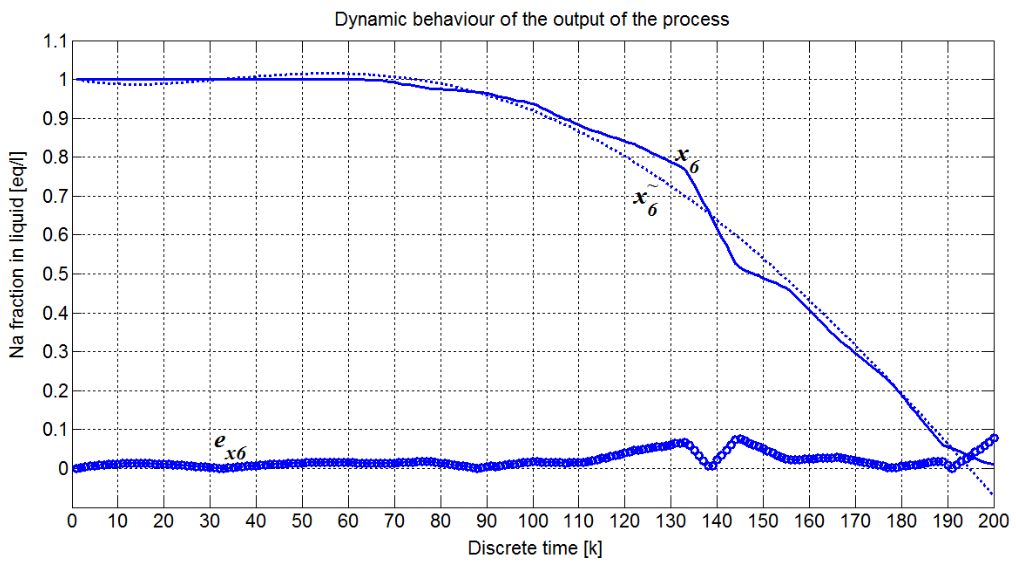


Figure 6.9. The process output, estimated output and the output error from the worst case results – Run 1 of Table 6.3

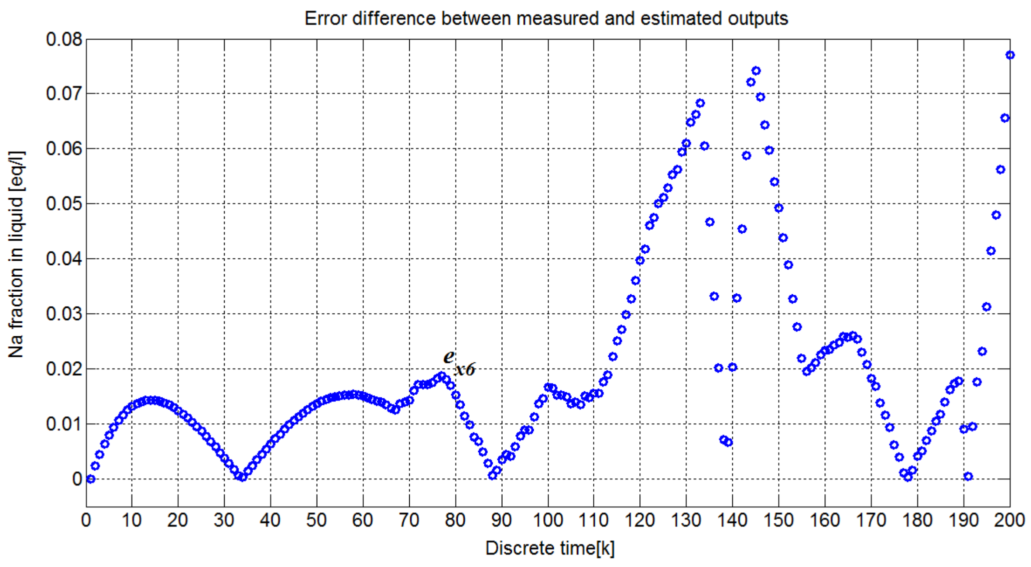


Figure 6.10. The output error signal in detail for the worst case results – Run 1 of Table 6.3

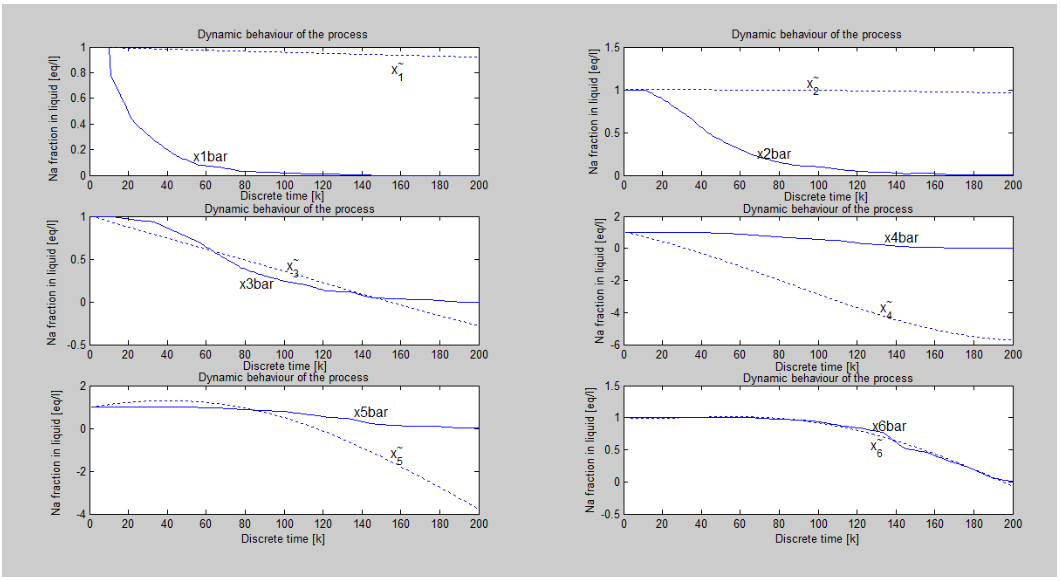


Figure 6.11. Model dynamic behaviour showing the real and estimated states for the worst case scenario – Run1 Table 6.3

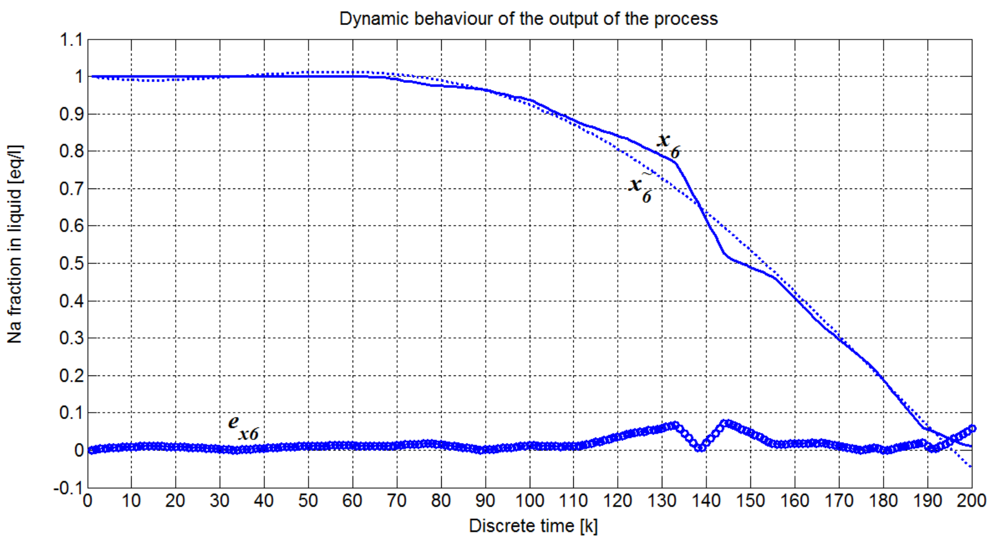


Figure 6.12. The process output, estimated output and the output error from the best results – Run 4 of Table 6.3

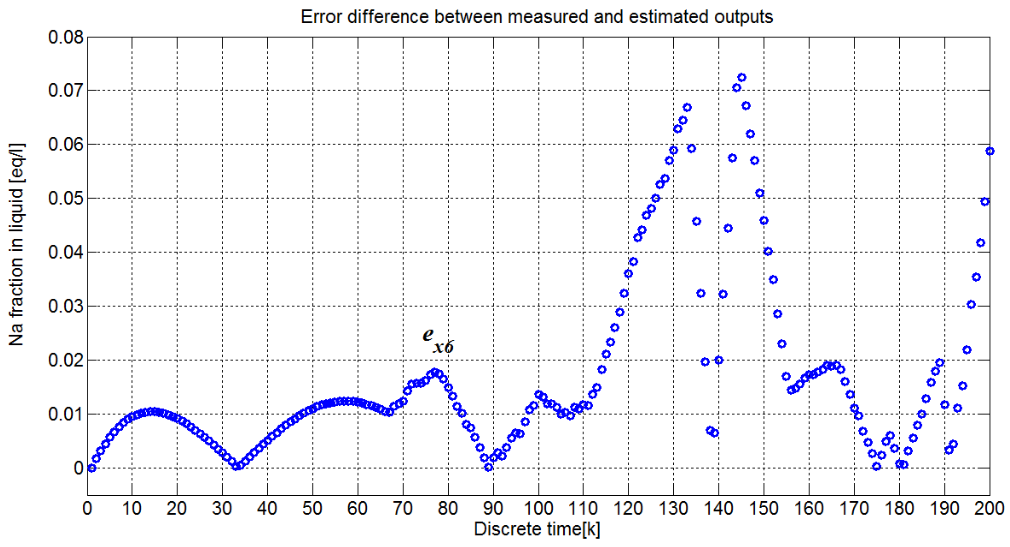


Figure 6.13. The output error signal in detail for the best results – Run 4 of Table 6.3

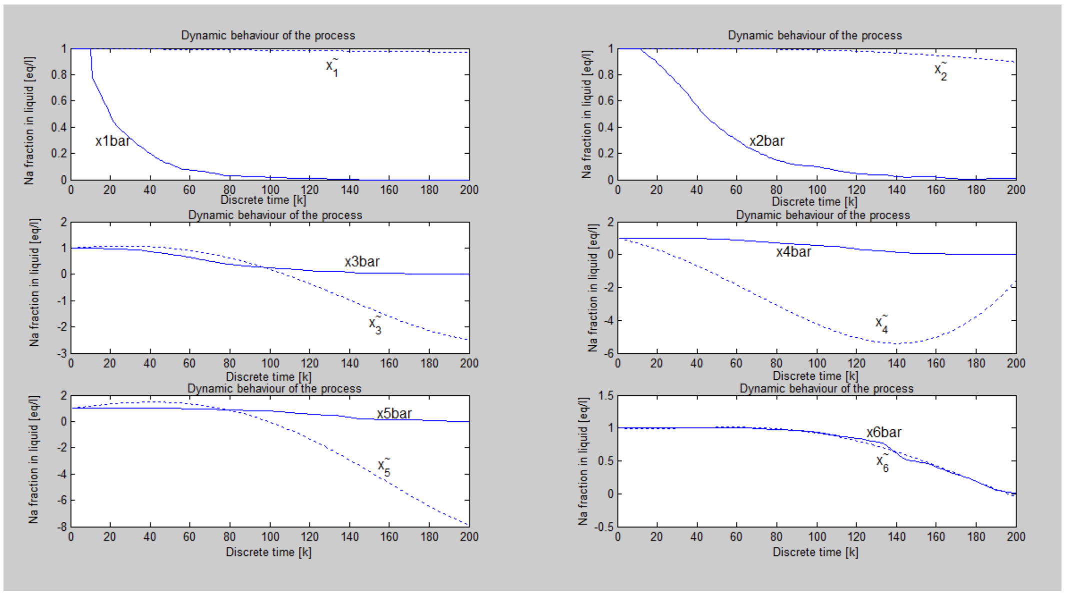


Figure 6.14. Model dynamic behaviour showing the real and estimated states from the best results – Run 4 Table 6.3

Table 6.4: Results of the second set of experiments for changing sampling period Δt showing the error behaviour and calculated run time

Estimation criterion the least squares error and the estimation run time for the second set of experiments		
Case 2: (Second set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	29.4149	168.8080
Run 2	28.5245	224.8900
Run 3	23.7122	173.3330
Run 4	22.9868	233.2180
Run 5	28.0460	230.4770

Results and discussion of the second set of experiments of the first method

The fourth run of *Case 2* at $a = 230 \times a$ and $b = 60 \times b$ of the given vectors and at the smallest number of stopping the procedure $\varepsilon = 0.0001$ produces the best error results in comparison to the rest of vectors sets. The rest of the results are also not far from these results except for the improved error. Run 2 shows that at reduced stopping number $\varepsilon = 0.0001$, the error is improved compared to higher stopping number $\varepsilon = 0.001$; but the calculation time is longer as expected.

The third set of experiments of the measured output method

The third set of experiments shows the effect of model coefficient vectors at reduced holdups values, Table 6.5 and Figures 6.15–6.22. The aim of this experiment is to observe the influence of reduced holdups values at different model coefficients. A new multiplication factor for a is also introduced at the last run of these experiments. These set of coefficients were chosen from a few different sets after a number of runs; they carry the most influence on the process dynamic behaviour. The rest of the experiments parameters used in the second set are maintained through out this set of experiments, the sampling period, the gradient and the procedure stopping number and the parameters deviation value.

Table 6.5: **The third set of experiments for the changing a and b model coefficients**

Changing model coefficients at reduced holdup values					
Process parameter (units)	Run 1	Run 2	Run 3	Run 4	Run 5
$h \ = 0.3293 \ l$	✓	✓	✓	✓	✓
$H \ = 0.42809 \ l$	✓	✓	✓	✓	✓
$F_L = 2000/60 \ (l/\text{min})$	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓
a (see Run number below)	$a = 230\times a$	$a = 230\times a$	$a = 230\times a$	$a = 230\times a$	$a = 480\times a$
b (see Run number below)	$b = 30\times b$	$b = 30\times b$	$b = 5.2\times b$	$b = 30\times b$	$b = 30\times b$
M	2000	✓	✓	✓	✓
dt, sampling period (Δt)	0.012	✓	✓	✓	✓
Δ_p, parameter deviation	0.001	✓	✓	✓	✓
$grad$, iteration gradient step (α)	0.076	✓	✓	✓	✓
ε, epsilon (very small number)	0.0001	✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

Each set of the different model coefficients used is shown below.

Run 1	$a = [0.8 \ 0.6 \ 0.5 \ 0.4 \ 0.3 \ 0.1]$	$b = [0.1 \ 0.25 \ 0.3 \ 0.4 \ 0.5 \ 0.6]$
Run 2	$a = [0.8 \ 0.6 \ 0.5 \ 0.4 \ 0.3 \ 0.2]$	$b = [0.1 \ 0.25 \ 0.3 \ 0.4 \ 0.5 \ 0.6]$
Run 3	$a = [0.8 \ 0.6 \ 0.5 \ 0.4 \ 0.3 \ 0.2]$	$b = [0.025 \ 0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.6]$
Run 4	$a = [0.8 \ 0.7 \ 0.5 \ 0.4 \ 0.3 \ 0.2]$	$b = [0.08 \ 0.2 \ 0.25 \ 0.7 \ 0.8 \ 0.9]$
Run 5	$a = [0.8 \ 0.7 \ 0.5 \ 0.4 \ 0.3 \ 0.2]$	$b = [0.025 \ 0.1 \ 0.25 \ 0.5 \ 0.6 \ 0.7]$

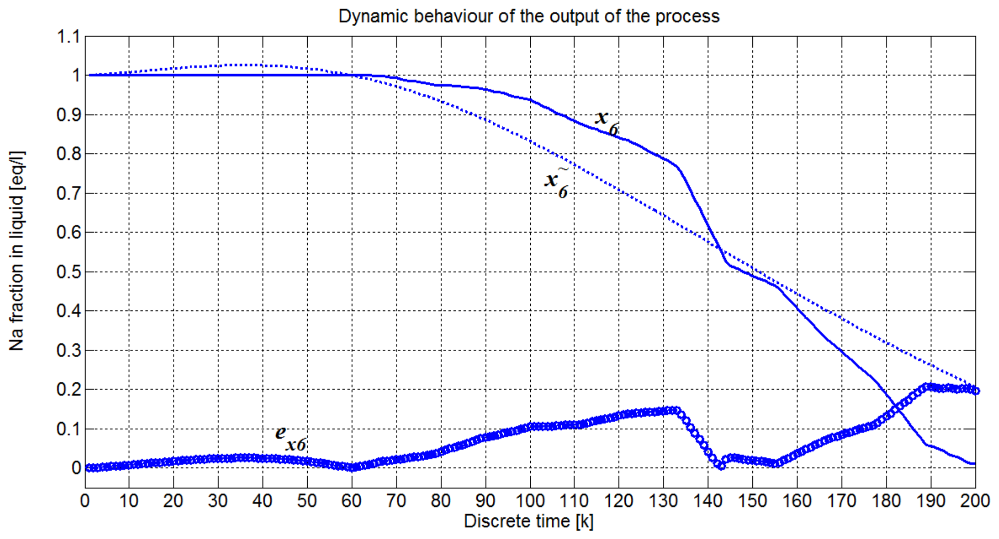


Figure 6.15. The process output, estimated output and the output error from the results of Run 1 of Table 6.5

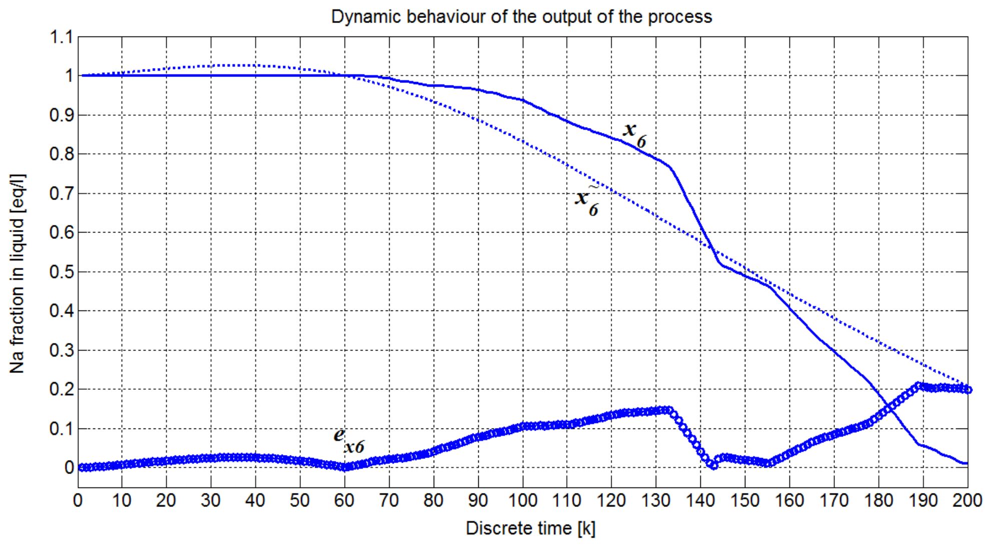


Figure 6.16. The process output, estimated output and the output error from the results of Run 2 for Table 6.5

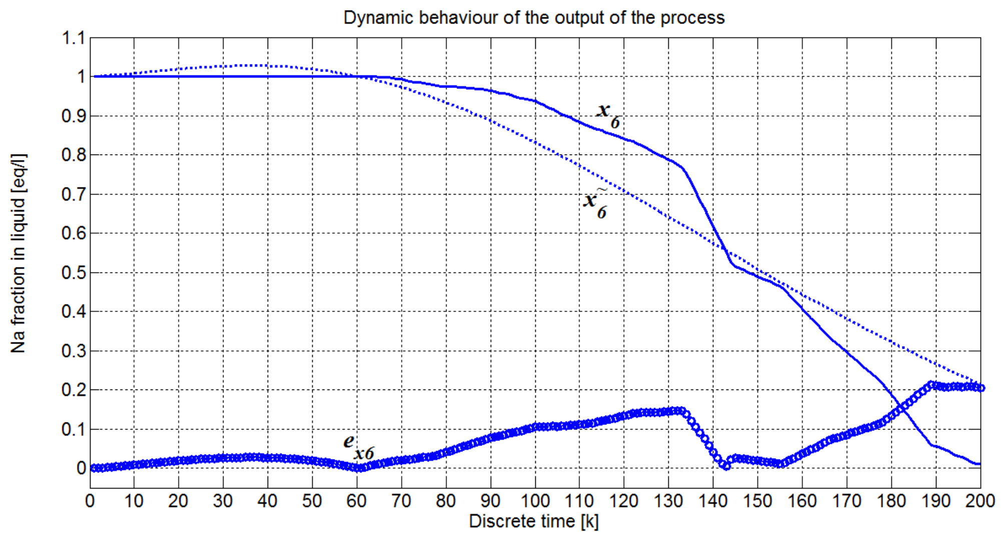


Figure 6.17. The process output, estimated output and the output error from the results of Run 3 for Table 6.5

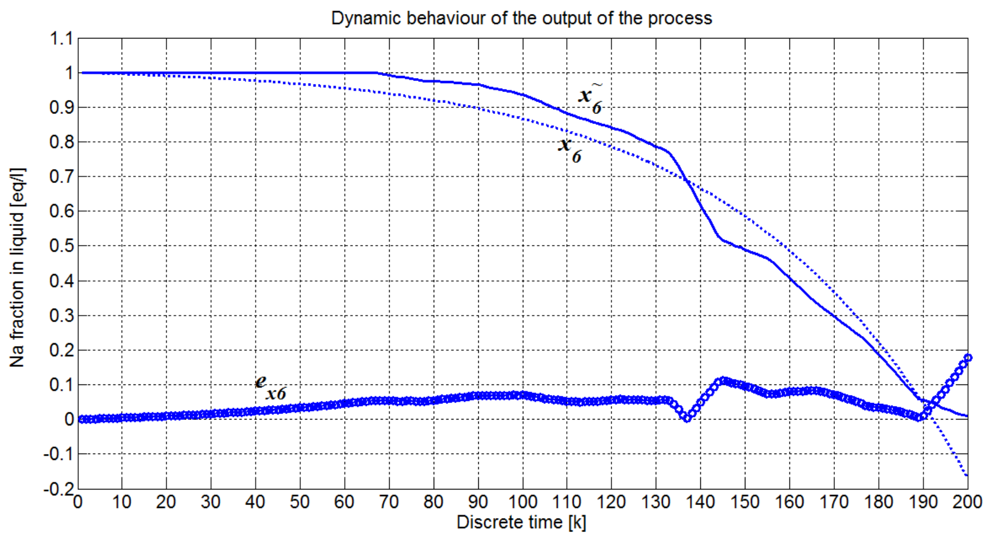


Figure 6.18. The process output, estimated output and the output error from the results of Run 4 of Table 6.5

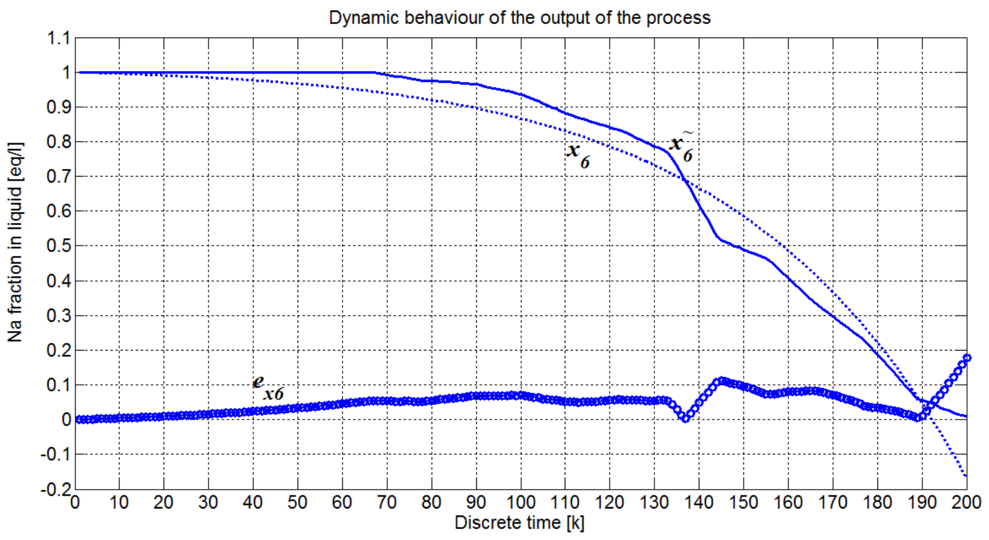


Figure 6.19. The process output, estimated output and the output error from the results of Run 5 of Table 6.5

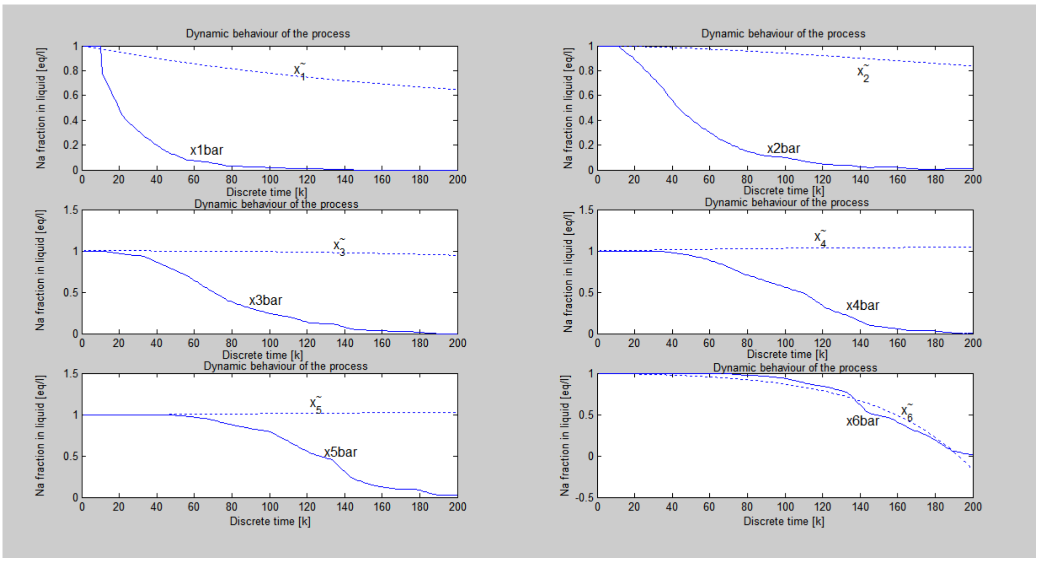


Figure 6.20. Model dynamic behaviour showing the real and estimated states – the best results – Run 5 Table 6.5

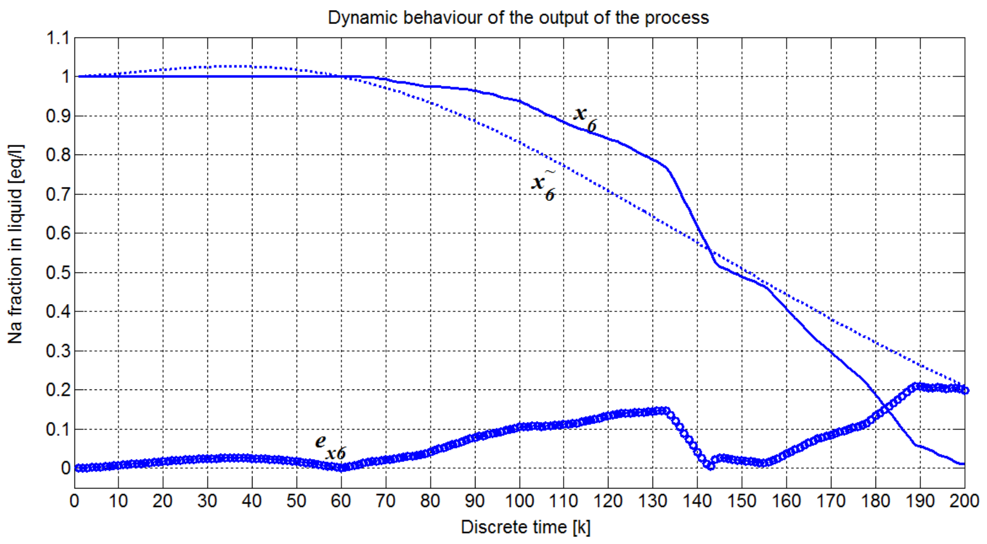


Figure 6.21. The process output, estimated output and the output error from the best results of Case 2 Table 6.3

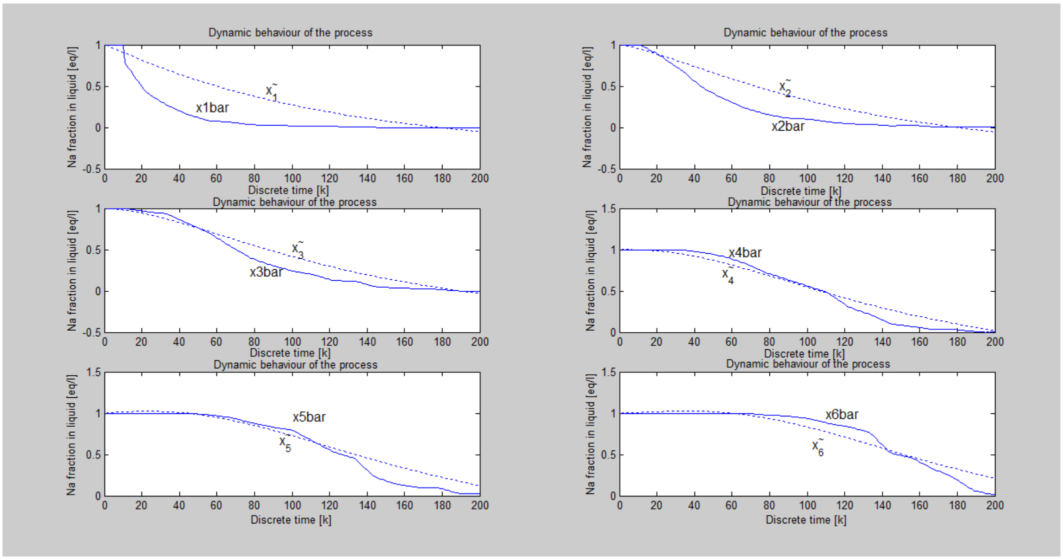


Figure 6.22. Model dynamic behaviour showing the real and estimated states – the best results of Case 2 Table 6.3

Table 6.6: **Results of the third set of experiments for changing a and b model coefficients showing the error behaviour and calculation run time**

Estimation criterion the least squares error and the estimation run time for the third set of experiments		
Case 3: (Third set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	327.1028	233.6670
Run 2	329.7488	266.0760
Run 3	338.1485	229.3940
Run 4	340.1815	233.6500
Run 5	130.9912	232.8440
Run with best values from Case 2	332.1313	233.2540

Results and discussion of the third set of experiments of the measured output method

All the combinations of given coefficients produced the same process behaviour. They also produced the same value for estimation error criterion and run time. In comparison to the best coefficients from the second set of experiments, the influence of holdups is the improvement of all the state estimates, Figures 6.20 and 6.22. The error stayed the same as the rest of runs. Run 5 produced the best criterion error but the overall states were not close in comparison to best values from *Case* 2.

The fourth set of experiments for the measured output method

The fourth set of experiments considers the influence of the *gradient step* using the best coefficients and sampling period from second and third sets of experiments. The aim of this experiment is to understand what influence the gradient step has in the estimation process. This would help to identify the best value to use for the gradient procedure that would produce the best estimate error. The results from this set of experiments are given in Figures 6.23–6.28.

Table 6.7: The fourth set of experiments showing the changing values of iteration gradient step α

The changing values of iteration gradient step ε						
Process parameter (units)	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6
$h = 32.93\ l$	✓	✓	✓	✓	✓	✓
$H = 42.809\ l$	✓	✓	✓	✓	✓	✓
$F_L = 2000/60\ (l/\text{min})$	✓	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓	✓
$a = [0.8\ 0.6\ 0.5\ 0.4\ 0.3\ 0.25]$	$a = 230 \times a$	✓	✓	✓	✓	✓
$b = [0.01\ 0.02\ 0.05\ 0.08\ 0.1\ 0.2]$	$b = 60 \times b$	✓	✓	✓	✓	✓
M	2000	✓	✓	✓	✓	✓
dt , sampling period (Δt)	0.012	✓	✓	✓	✓	✓
Δ_p , parameter deviation	0.001	✓	✓	✓	✓	✓
$grad$, iteration *gradient step* (α)	0.01	0.076	0.1	0.56	0.82	2.0
ε , epsilon (very small number)	0.0001	✓	✓	✓	✓	

✓ means the value stays the same as in the previous mention in the table

The table shows the set of experiments for the changing *gradient step* value, starting from the smallest of 0.01 up to the last value of 2.0 in no specific order. The best model coefficients from set of experiments of Case 3 of the measured output method.

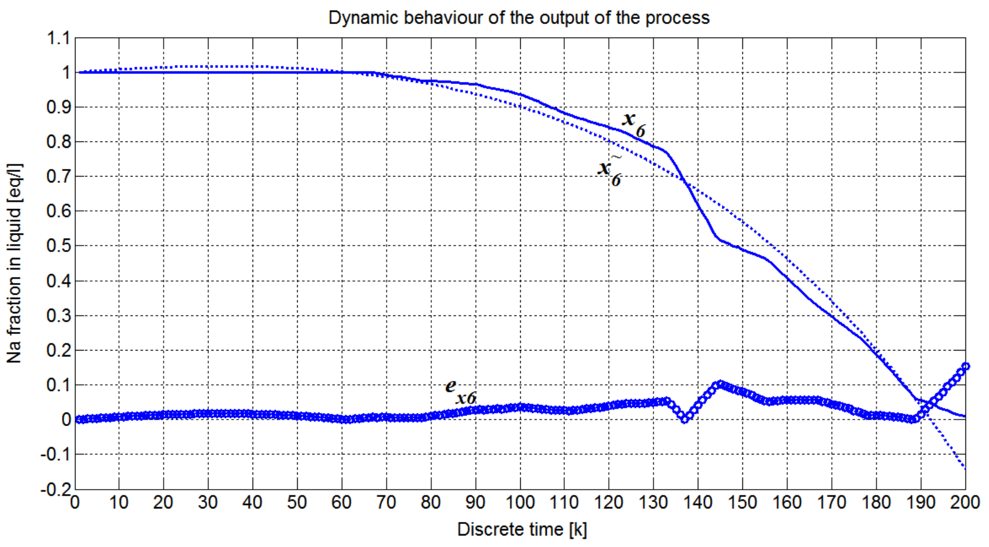


Figure 6.23. The process output, estimated output and the output error of the worst case results – Run 1 of Table 6.7

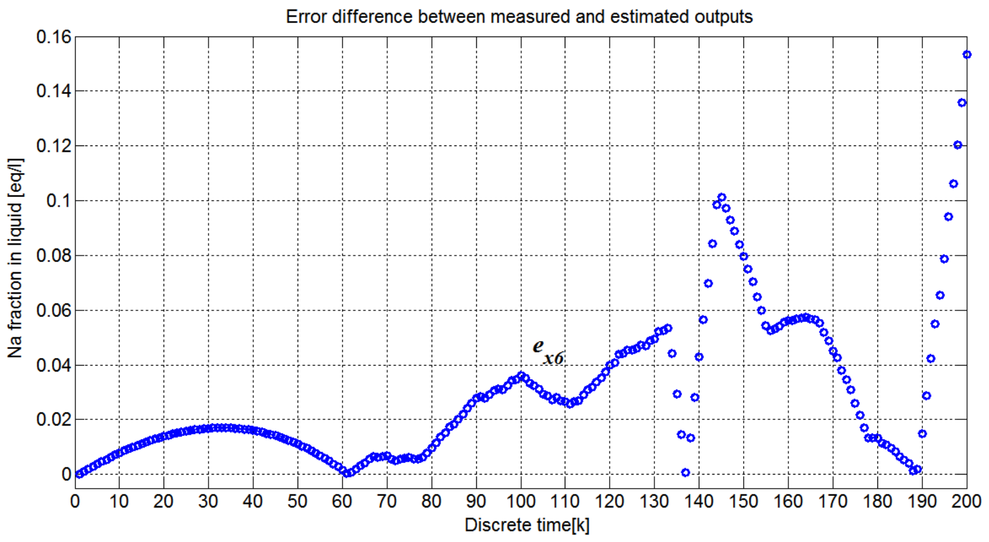


Figure 6.24. The output error signal in detail for the worst case results – Run 1 of Table 6.7

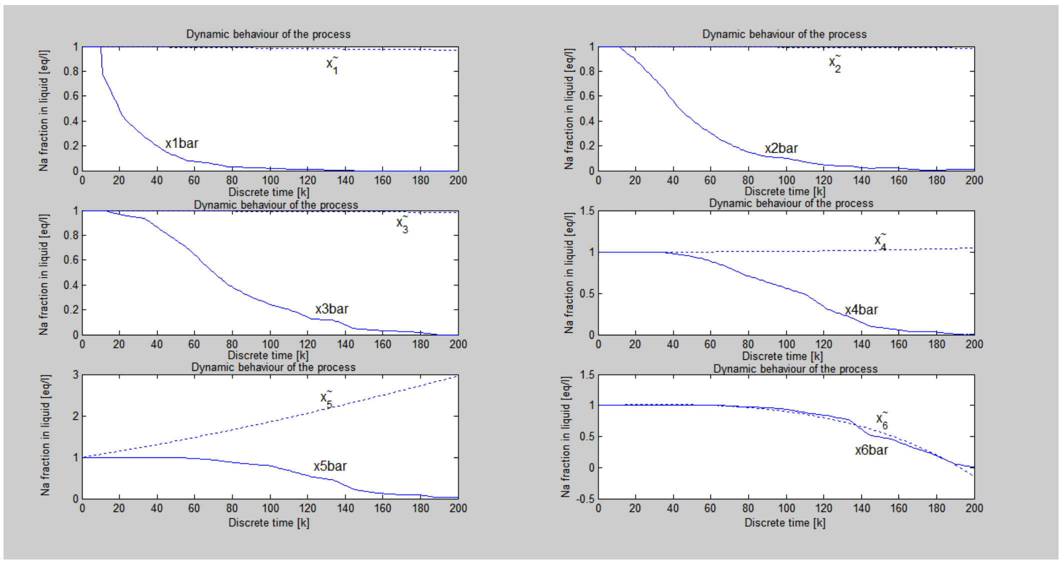


Figure 6.25. Model dynamic behaviour showing the real and estimated states of the worst case results – Run1 Table 6.7

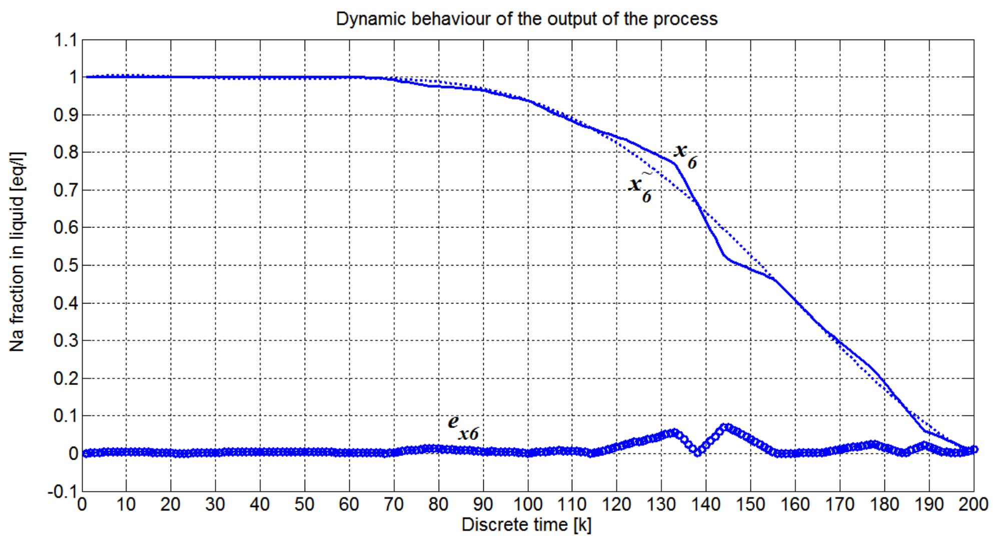


Figure 6.26. The process output, estimated output and the output error of the best results – Run 6 of Table 6.7

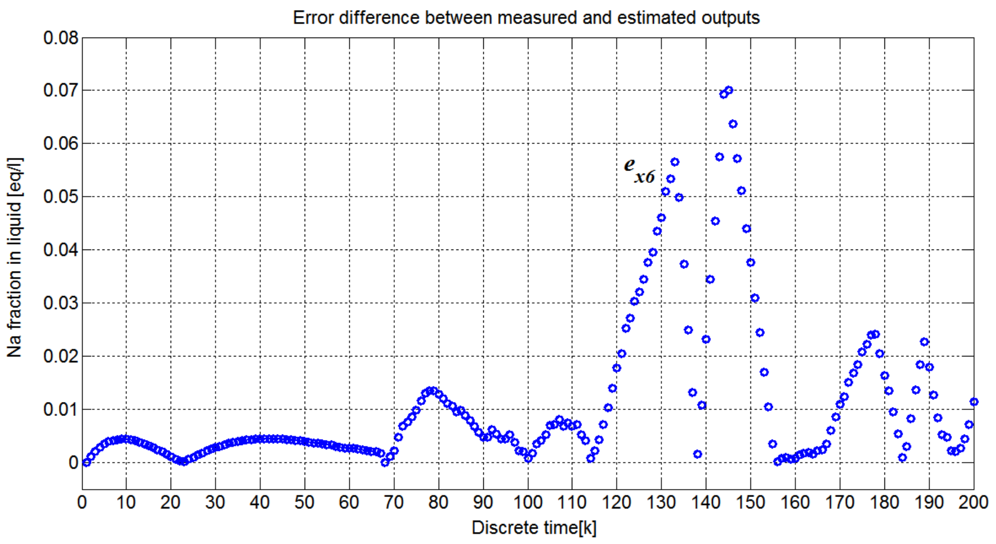


Figure 6.27. The output error difference in detail for the best results – Run 6 of Table 6.7

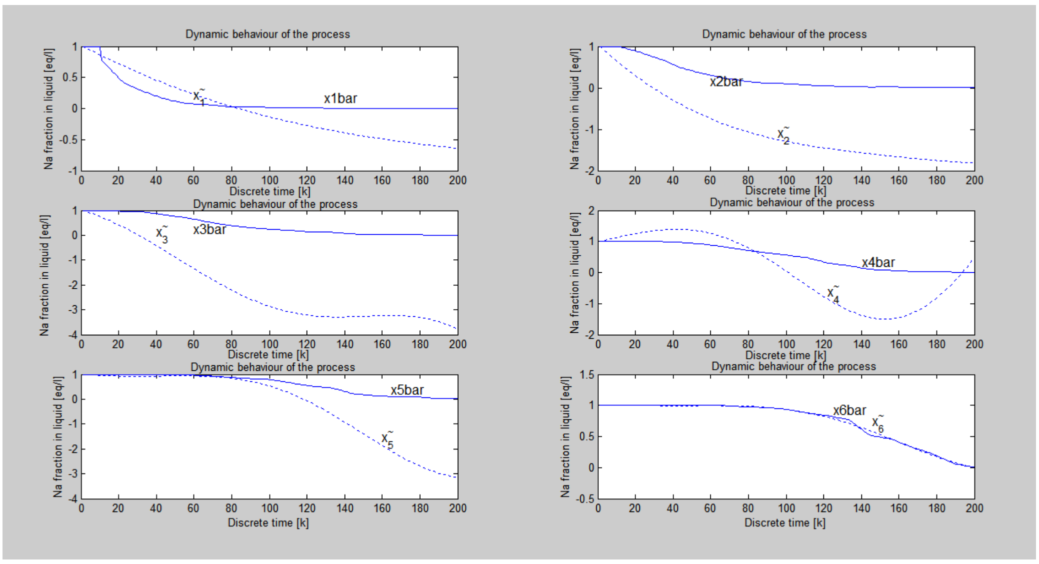


Figure 6.28. Model dynamic behaviour showing the real and estimated states from the best results – Run 6 Table 6.7

Table 6.8: **Results of fourth set of experiments for changing values of iteration gradient step α showing the error behaviour and calculated run time**

Estimation criterion the least squares error and the estimation run time for the fourth set of experiments		
Case 4: (Fourth set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	64.2635	138.7950
Run 2	64.7528	207.9740
Run 3	64.5122	180.7280
Run 4	33.9885	224.7270
Run 5	22.9868	233.9430
Run 6	13.5571	257.0940

Results and discussion of the fourth set of experiments of the measured output method

Though the best estimation error result is produced at a highest run time, the smallest error is considered best because all the other run times are relatively high. The best criterion is obtained at the highest value of the gradient of the given values 0.01 to 2.0, Table 6.7. The highest gradient step value is 2.0.

The fifth set of experiments for the measured output method

The experiment considers the *changing disturbance input* at the best sampling period, model coefficients, and best gradient step as calculated in the earlier sets of experiments, Table 6.9. Figures 6.29–6.34 show the best and the worse case scenarios of this set of experiments.

Table 6.9: **The fifth set of experiments of the changing disturbance input x_f at best sampling period**

Changing disturbance input at best sampling period						
Process parameter (units)	Values/ (Reference)	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 32.93\ l$	✓	✓	✓	✓	✓	✓
$H = 42.809\,l$	✓	✓	✓	✓	✓	✓
$F_L = 2000/60\ (l/\min)$	✓	✓	✓	✓	✓	✓
$a = [0.8\ 0.6\ 0.5\ 0.4\ 0.3\ 0.25]$	$a = 230 \times a$	✓	✓	✓	✓	✓
$b = [0.01\ 0.02\ 0.05\ 0.08\ 0.1\ 0.2]$	$b = 60 \times b$	✓	✓	✓	✓	✓
x_f, disturbance input	$x_f = 0.435$	$x_f = 0.055$	$x_f = 0.110$	$x_f = 0.550$	$x_f = 0.755$	$x_f = 1.00$
M	2000	✓	✓	✓	✓	✓
dt, sampling period (Δt)	0.012 min	✓	✓	✓	✓	✓
Δ_p, parameter deviation	0.001	✓	✓	✓	✓	✓
$grad$, iteration *gradient step* (α)	2.0					
ε, epsilon (very small number)	0.0001	✓	✓	✓	✓	

✓ means the value stays the same as in the previous mention in the table

In this set of experiments the best sampling period and constant control input are used. The investigation is to observe the influence of the disturbance signal on the process and to identify the worse and optimum cases of process dynamics based on the disturbance values.

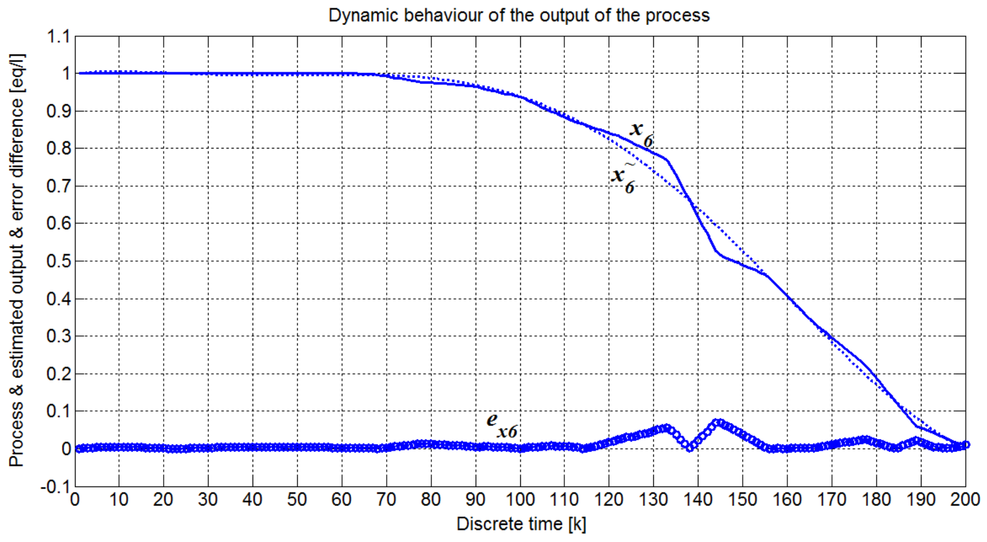


Figure 6.29. The process output, estimated output and the output error of the best results – Run 2 of Table 6.10

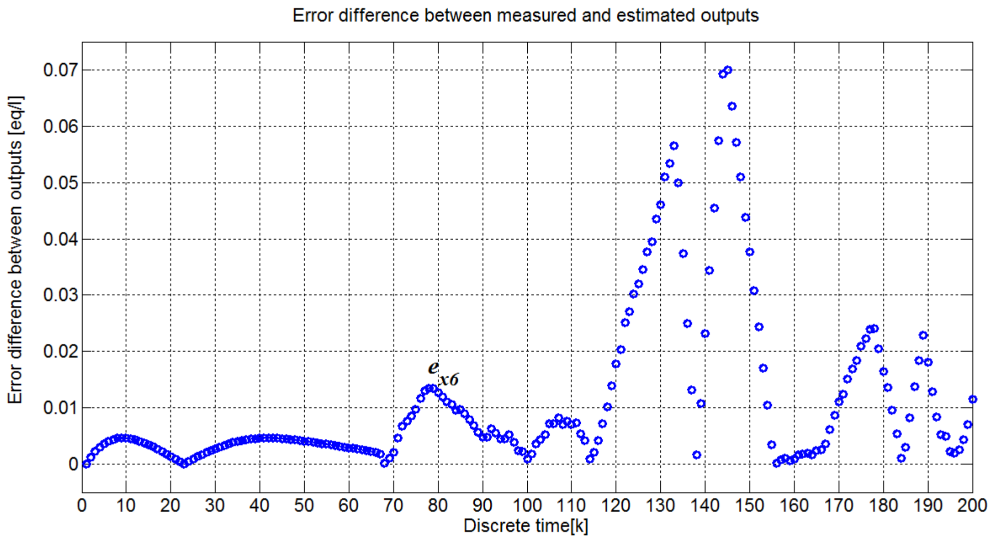


Figure 6.30. The output error difference in detail for the best results – Run 2 of Table 6.10

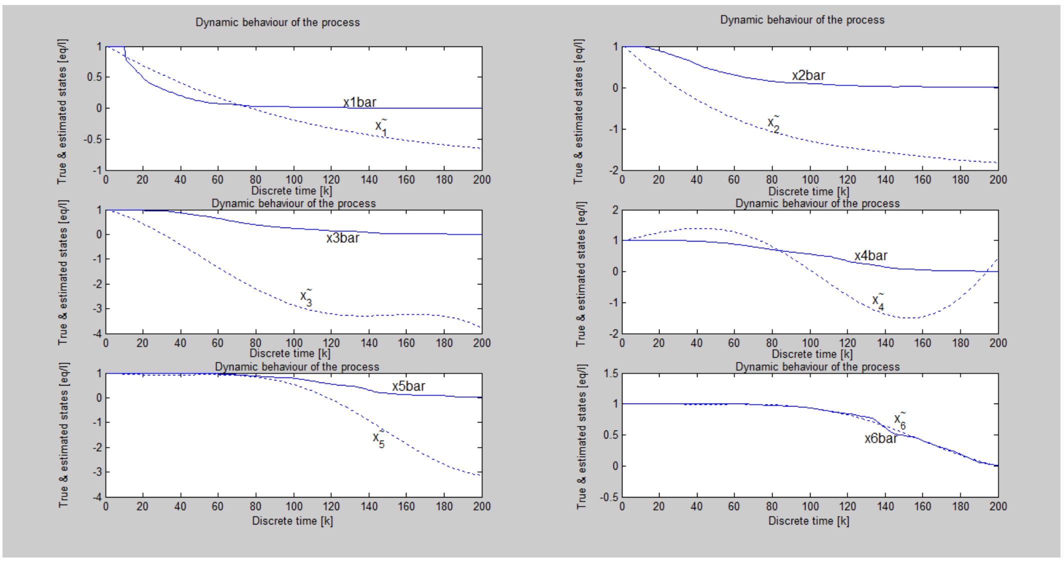


Figure 6.31. **Model dynamic behaviour showing the real and estimated states of the best results – Run 2 Table 6.10**

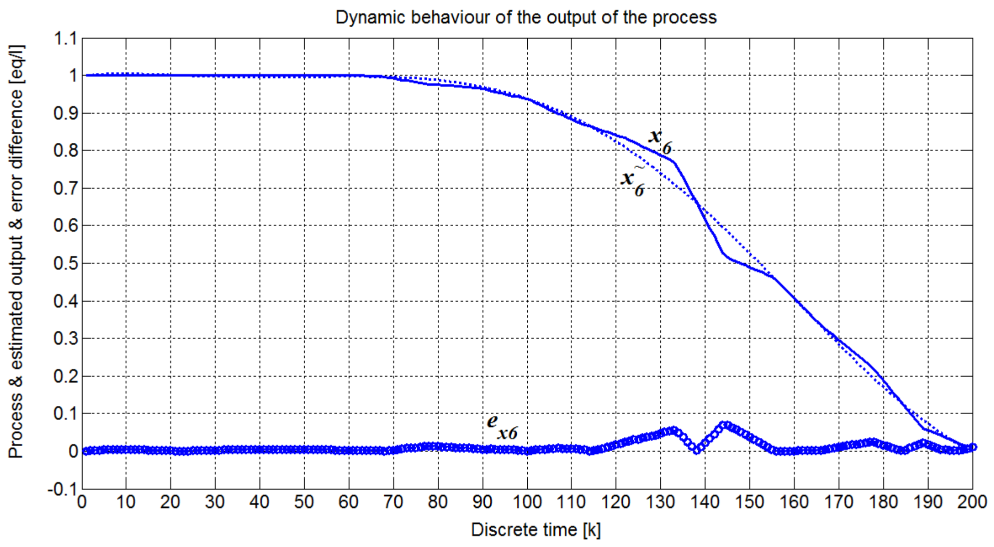


Figure 6.32. **The process output, estimated output and the output error of the worst case – Run 4 of Table 6.10**

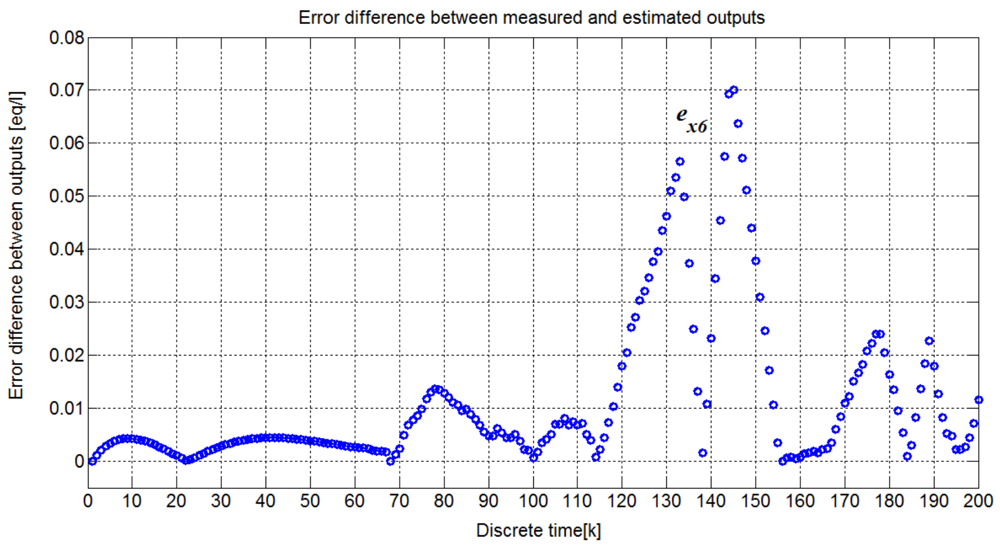


Figure 6.33. The output error difference in detail for the worst case – Run 4 of Table 6.10

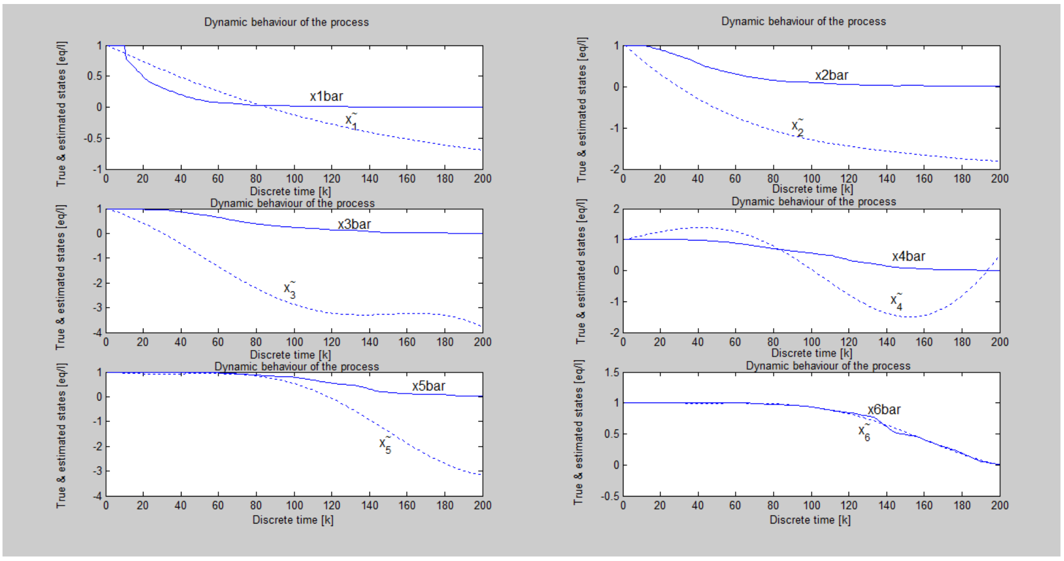


Figure 6.34. Model dynamic behaviour showing the real and estimated states for the worst case – Run 4 Table 6.10

Table 6.10: Results of the fifth set of experiments for the changing disturbance input x_f showing the error behaviour and calculated run time

Estimation criterion the least squares error and the estimation run time for the fifth set of experiments		
Case 4: (Fourth set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	13.5529	254.2890
Run 2	13.5492	253.3760
Run 3	13.5574	254.5300
Run 4	13.5618	255.8380
Run 5	13.5596	253.2850

Results and discussion of the fifth set of experiments of the measured output method

The best criterion is obtained at the disturbance input of $x_f = 0.110$; this is on the lower scale of the reference value, Table 6.10. It can also be concluded that though there is some change in the criterion, the difference is insignificantly small; which means that the method reacts well to all the possible range of disturbances that may occur. The algorithm for all other given parameter values produces best result at the lower disturbance, though not the lowest of all the tested values.

The sixth set of experiments for the measured output method

The experiment considers the changing *initial model parameter values* at the best sampling period, model coefficients, and best gradient step as calculated in the earlier sets of experiments. Figures 6.35–6.40 show the best case and the worst case scenario of this set of experiments. The criterion results are demonstrated on the Table 6.12.

Table 6.11: **The sixth set of experiments of the changing initial parameter values at best sampling period**

Changing initial parameter values of the process model						
Process parameter (units)	Values	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 32.93\ l$	✓	✓	✓	✓	✓	✓
$H = 42.809\ l$	✓	✓	✓	✓	✓	✓
$x_f = 0.110$	✓	✓	✓	✓	✓	✓
$a = [0.8\ 0.6\ 0.5\ 0.4\ 0.3\ 0.25]$	$a = 230 \times a$	✓	✓	✓	✓	✓
$b = [0.01\ 0.02\ 0.05\ 0.08\ 0.1\ 0.2]$	$b = 60 \times b$	✓	✓	✓	✓	✓
Parameters (p)	$p = p$	$p = 0.5p$	$p = 1.5p$	$p = 2.5p$	$p = 10p$	$p = 25p$
M	2000	✓	✓	✓	✓	✓
dt , sampling period (Δt)	0.012 min	✓	✓	✓	✓	✓
Δ_p , parameter deviation	0.001	✓	✓	✓	✓	✓
$grad$, iteration *gradient step* (α)	2.0					
ε , epsilon (very small number)	0.0001	✓	✓	✓	✓	

✓ means the value stays the same as in the previous mention in the table

The table shows the changing initial parameter values in multiples of 0.5 and not in any particular order from 0.5 to 25.

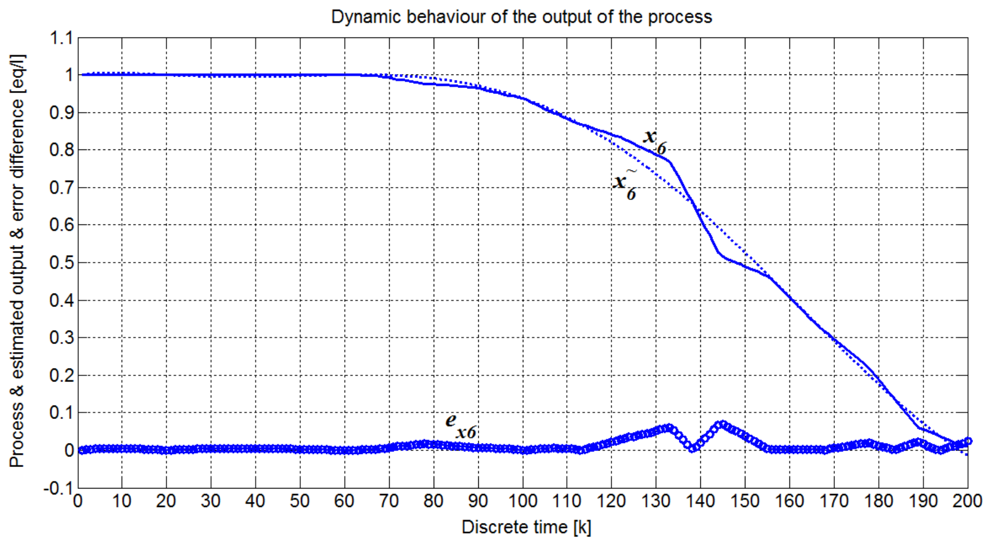


Figure 6.35. The process output, estimated output and the output error of the best results – Run 4 of Table 6.11

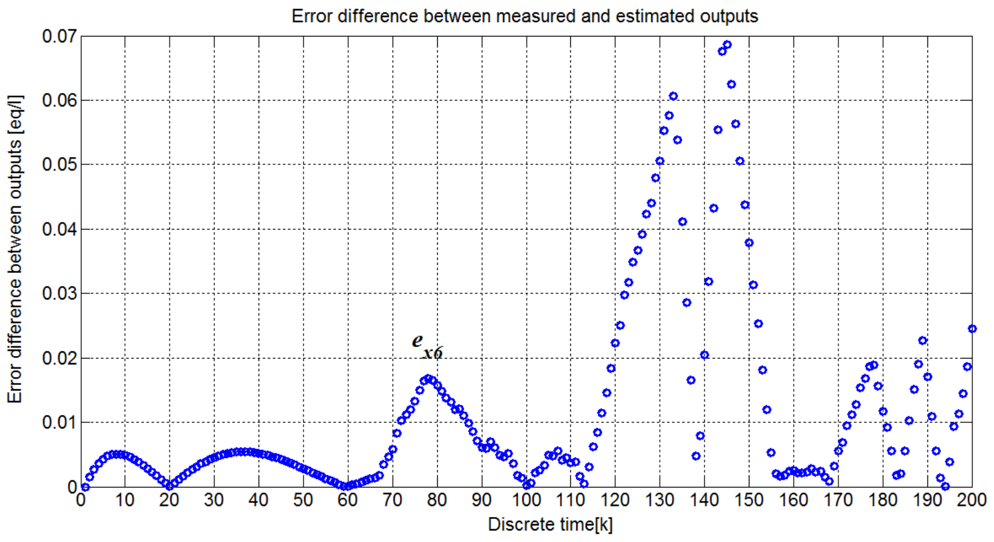


Figure 6.36. The output error difference in detail for the best results – Run 4 of Table 6.11

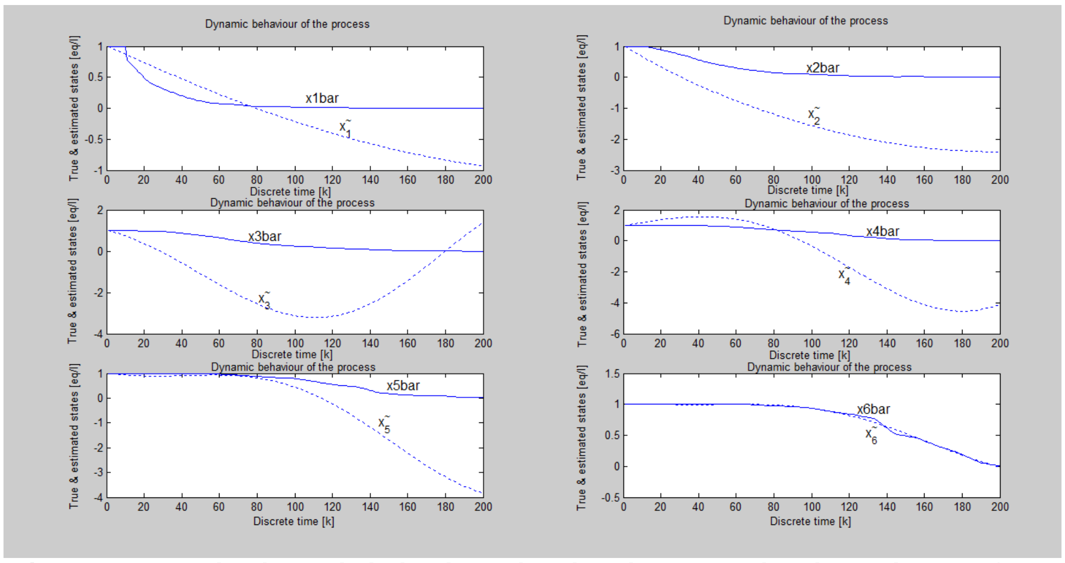


Figure 6.37. Model dynamic behaviour showing the real and estimated states from the best results – Run 4 Table 6.11

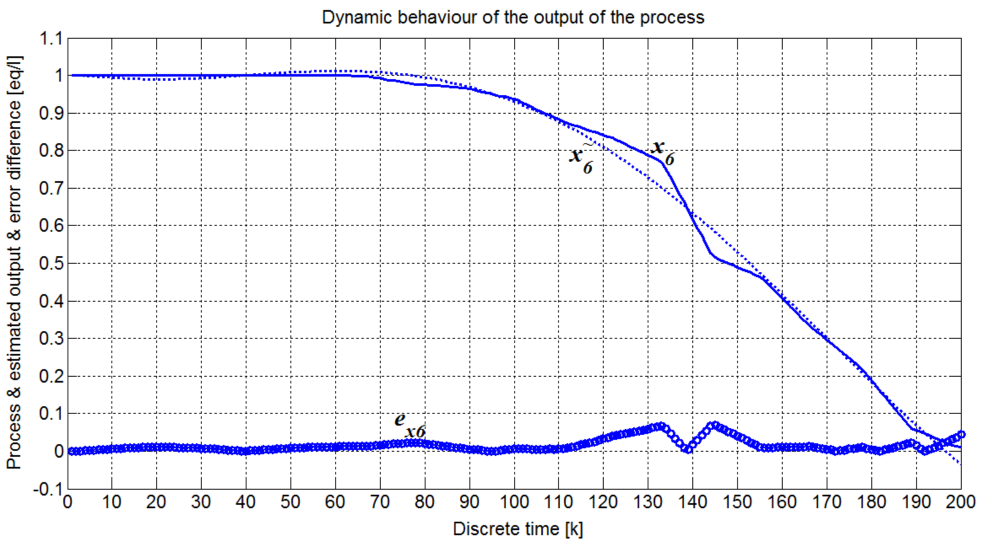


Figure 6.38. The process output, estimated output and the output error of the worst case results – Run 5 of Table 6.11

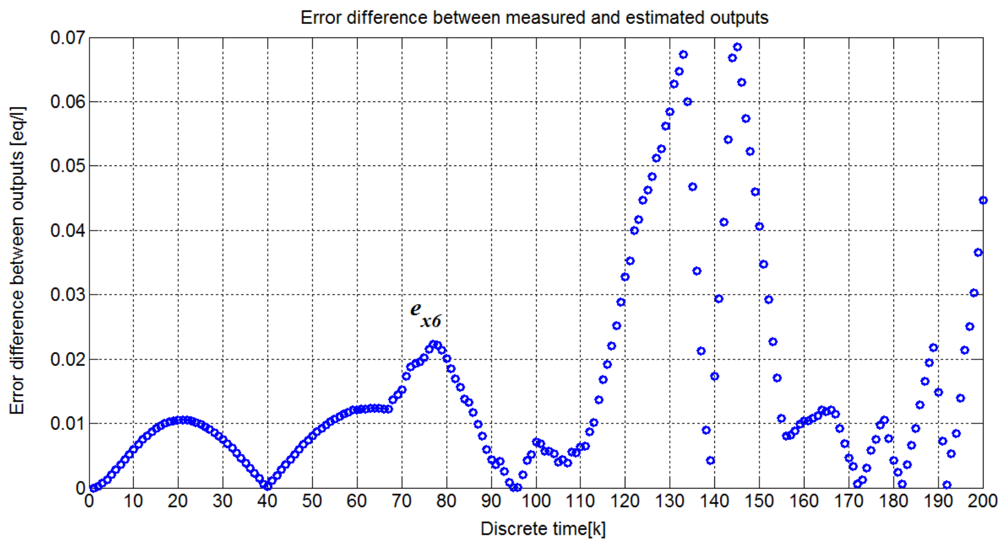


Figure 6.39. The output error difference in detail for the worst case results – Run 5 of Table 6.11

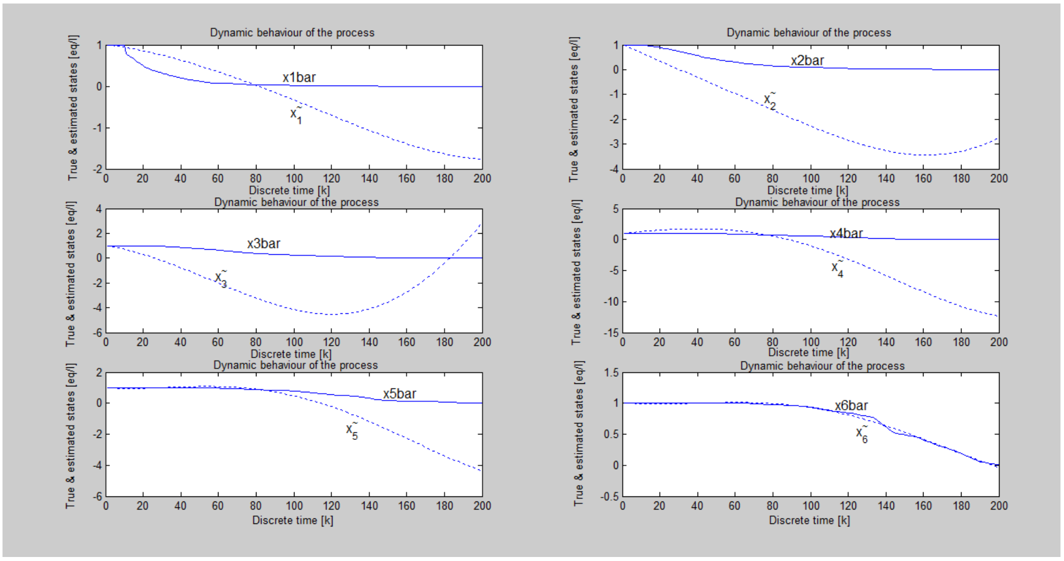


Figure 6.40. Model dynamic behaviour showing the real and estimated states for the worst case results – Run 5 Table 6.11

Table 6.12: Results of the sixth set of experiments for the changing initial parameters showing the error behaviour and calculated run time

Estimation criterion the least squares error and the estimation run time for the sixth set of experiments		
Case 4: (Fourth set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	15.2090	254.9090
Run 2	16.9866	256.3560
Run 3	14.8062	255.7330
Run 4	14.5746	257.1210
Run 5	19.8859	256.1220

Results and discussion of the sixth set of experiments of the measured output method

The best result is obtained at ten times the original values of initial parameters, and the values above that tend to produce higher least squares error, Table 6.12. The best results are obtained at ten times the original value, Figures 6.35–6.37.

6.5. Linear model parameter estimation using the state vector measurements (the *least squares* method)

The method of solution of determining the unknown parameters using the process state vector measurements has two possible solutions, i.e., 1) *the least squares method*, and 2) *the direct method*. This section covers the measured output method of solution and the direct method is covered in the next section.

6.5.1. The procedure for linear model parameter estimation using the state vector measurements

The least squares method can be presented using the flow diagram below, Figure 6.41 (Olsson & Newell, 1999; Walter & Pronzato, 1995:113).

This technique is similar to the measured output method of solution in Section 6.4, except that the criterion is now based on the error difference between the real and estimated state vectors, $e_x(k) = \overline{x}(k) - \hat{x}(k)$. The two state vectors are; the set state values obtained from measured data and the ones calculated from the model equations and therefore the criterion becomes

$$J(p) = \sum_{k=1}^{K} \left\| \overline{x}(k) - x(k) \right\|^2 = \sum_{k=1}^{K} \left\| e_x(k) \right\|^2 \qquad (6.16)$$

where

$\overline{x}(k)$ – are the measured states and

$\hat{x}(k)$ – are the model simulated states.

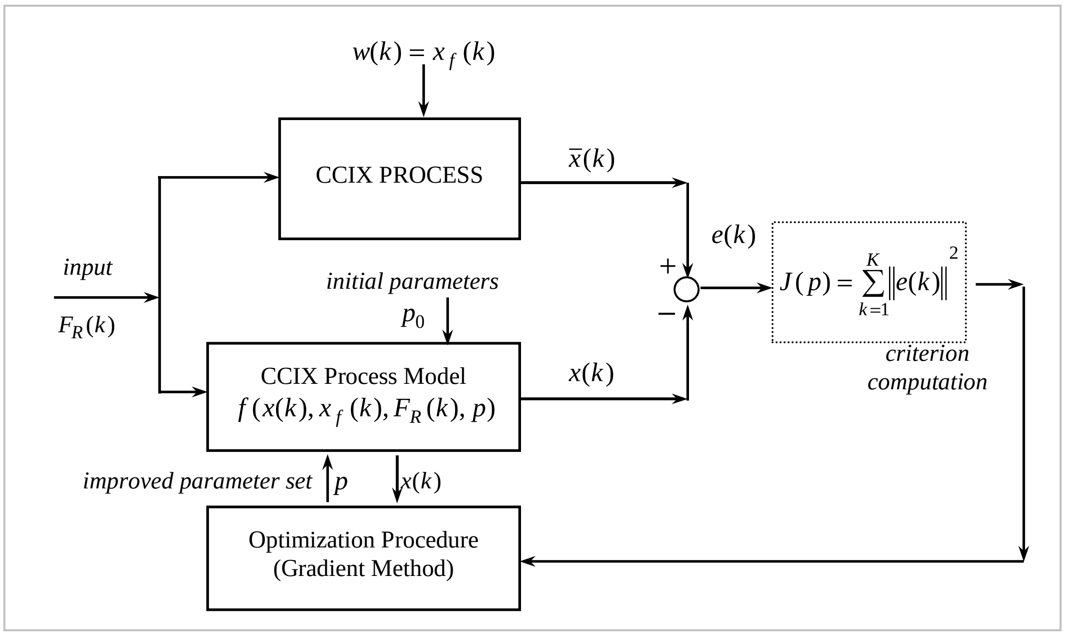


Figure 6.41. **Parameter estimation procedure using optimization techniques based on the state of measured plant data**

(Adapted from Olsson & Newell, 1999; Walter & Pronzato, 1995:113)

The procedure can be summarised as follows:

Using the model Equations (6.9), the initial state values are set to obtain the initial measured state values from the given data, then both the initial state value sets are extended for the full optimization period, $k = \overline{0, K}$. The minimization criterion is determined using error difference between the two sets, Equation (6.13), based on current parameter values $J(p_i) = J(p_1, p_2, ..., p_m, p_{m+1}, ..., p_q)$.

Then the deviated value of the parameters $\Delta p_i = p_i + \Delta_p$, is calculated. From this value, the model Equations (6.9) are determined again using the deviated parameters Δp_i.

The minimization criterion is recalculated, now using the deviated parameters $J(p_i) = J(p_1, p_2, ..., p_i + \Delta_p, p_{i+1}, ..., p_q)$. In this case, it is not possible to determine the values of the parameters analytically and therefore the *gradient approximation method* is used. The gradients are determined by Equation (6.15)

$$\frac{\partial J(p)}{\Delta p_i} = \frac{J(p_1, p_2, ..., p_i, p_{i+1}, ..., p_q) - J(p_1, p_2, ...\Delta p_i, p_{i+1}, ..., p_q)}{\Delta_p}$$

The *method of fastest descent* is then applied to find the direction where the parameters reduce the fastest (for optimality). The direction of the fastest descent is opposite the direction of the gradient and at the initial point it coincides with the direction where the criterion reduces the fastest way for infinitely small changes in the vector of parameters.

The gradients of the fastest descent dp are then calculated, where $dp = \left[dp_1, dp_2, ..., dp_i, ..., dp_q\right]^T$. From the gradients, the direction of the gradient $dp_i = \dfrac{-\alpha\left[\dfrac{\partial J(p)}{\partial p_i}\right]}{S}$ is then determined from Equation (6.14), where S is the weighted sum of parameters $S = \left[\sum_{k=1}^{K}\left[\dfrac{\partial J(p)}{\partial p_i}\right]^2\right]^{1/2}$ over optimization period $k = \overline{1, K}$. The procedure is repeated until optimal values of parameter vector are obtained or the terminating condition is reached.

6.5.2. The algorithm of the linear model parameter estimation using the least squares method

1. Initialize the process data values as in Section 6.4.2:
2. Initial parameters for the gradient procedure as in Section 6.4.2:
3. Generate the vector of parameters, in this case from Equations (6.3–6.5) and (6.8)

$$\begin{aligned} p_0 &= \left[p_1, p_2, ..., p_m, p_{m+1}, ..., p_q\right]^T = \\ &= \left[l_1, m_{12}, m_1, k_1, l_2, m_{23}, m_2, k_2, ..., l_n, m_{n,n+1}, m_n, k_n, ..., l_N, m_N, k_N\right] \end{aligned}$$

   For $N = 6$, the number of parameters is $q = 23$, $p \in R^q = p \in R^{23}$.
4. Solve the model Equations (6.9) using initial values for parameters p_0 and initial state values $x(k) = x_0$ as obtained from the measured data

$$\begin{aligned} x_n(k+1) = x_n(k) + p_m \Delta t x_{n-1}(k) - p_m \Delta t x_n(k) + p_{m+1} \Delta t x_{n+1}(k) F_R(k) - \\ - p_{m+2} \Delta t x_n(k) F_R(k) + p_{m+3} \Delta t F_R(k) \end{aligned}$$

   4.1. Set optimization period $k = \overline{1, K}$ and solve the equations for the full trajectory of the optimization period.
5. Declare the measured states $\bar{x}(k)$ from the measured data.
6. Calculate the value of the criterion based on the measured states $\bar{x}(k)$ and on the calculated state model variables for the full optimization period, i.e.,

$$J(p) = \sum_{k=1}^{K} \left\| e(k) \right\|^2 = \sum_{k=1}^{K} \left\| \bar{x}(k) - x(k) \right\|^2 .$$

7. Initialize the gradient procedure stopping condition $j=1$:

   7.1. Declare the $states()$ function arguments, $states(k)$, $F_R(k)$, $states\_in(k)$, x_0, p, dt for full trajectory $k=\overline{1,K}$,

   7.2. And then call the $states()$ function to determine the current state values,

   7.3. Calculate the error difference between the measured states and the current model states, $e(k)=\bar{x}(k)-x(k)$,

   7.4. Calculate the least squares difference, for the full state trajectory $J(p)=\sum_{k=1}^{K}\|e(k)\|^2$,

   7.5. For total number of parameters, $i=1,2,...,r$ solve for deviated parameter for the i^{th} component of the vector of parameters starting with the first parameter value p_1,

   $\Delta p_i^j = p_i^j + \Delta_p$ and $r=23$,

   7.5.1. Calculate the deviated value of the criterion for the i^{th} component, $J(p_1, p_2, ..., \Delta p_i^j, p_{i+1}, ..., p_r)$,

   7.5.2. Calculate the new state values based on deviated parameters for the full trajectory as in step 7.1,

   7.5.3. Calculate deviated value of the criterion,

   » first clear the previous deviation values, $J(p)=0$

   » for the full trajectory, $k=\overline{1,K}$ calculate the deviated values,

   $J(p)=J(p)+\sum_{k=1}^{K}\|e(k)\|^2$,

   7.5.4. Calculate the gradients $\frac{\partial J(p)}{\partial p_i}$ using the Equation (6.13) for every parameter, $i=\overline{1,r}$,

   7.5.5. Form a vector of the criterion gradients for all newly calculated parameters, $dp=[dp_1,dp_2,...,dp_q]$,

   7.5.6. Calculate the new value of parameters, $p(i)=p(i)+\Delta_p$.

8. Calculation of the gradients for the fastest descent dp_i:

8.1. Calculate the weighted sum $S = \left[\sum_{k=1}^{K} \left(\frac{\partial J(p)}{\partial p_i^j} \right)^2 \right]^{1/2}$,

8.2. Calculate the direction of the gradients $dp_i^j = \frac{-\left[\alpha \frac{\partial J(p)}{\partial p_i^j} \right]}{S}$,

$i = 1, 2, \dots, q$, Equation (6.13).

9. Determine the error difference between the measured states and the calculated states based on estimated parameters.
10. Determine criterion for termination of the calculation, test if the norm with the new parameters is smaller than or equal to the small positive number, ε for all parameters $i = 1, 2, ..., r$,

10.1. first set the current state vector values to be initial values, $x(k) = x_0$,

10.2. determine the current state values based on improved parameters, same as in point 7.1,

10.3. test if the norm condition is reached, $norm(p_i) \le \varepsilon$:

» if true, terminate calculation and end the procedure, the current parameters are considered optimal,

» if the norm is still greater than ε , calculate improved parameters using the current parameters of the parameter vector p_i^j ,

» increase termination count, $j = j + 1$,

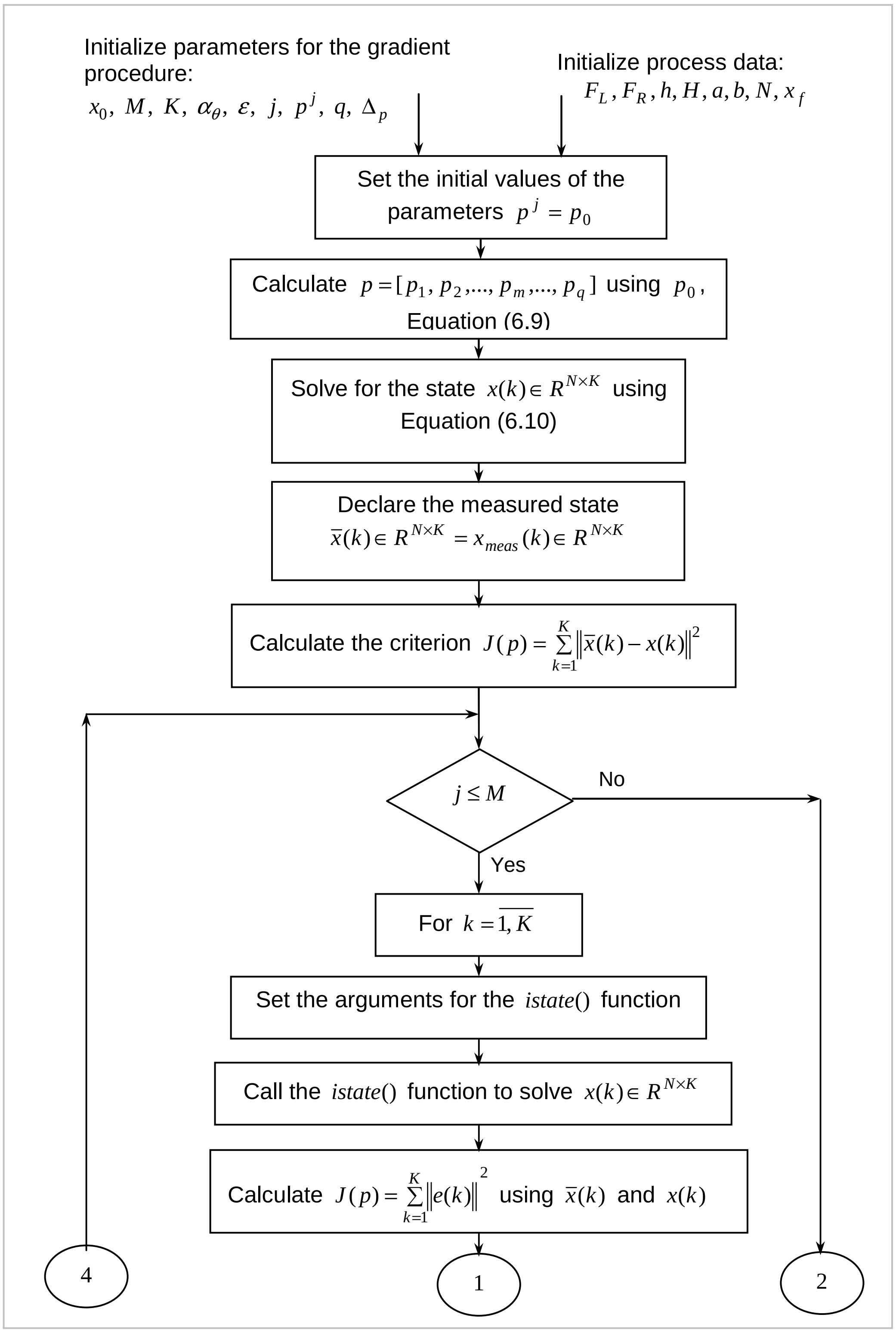
» improve the gradient by reducing delta, $\Delta = \Delta / 2$, and exit test loop,

10.4. repeat from step 7.1,

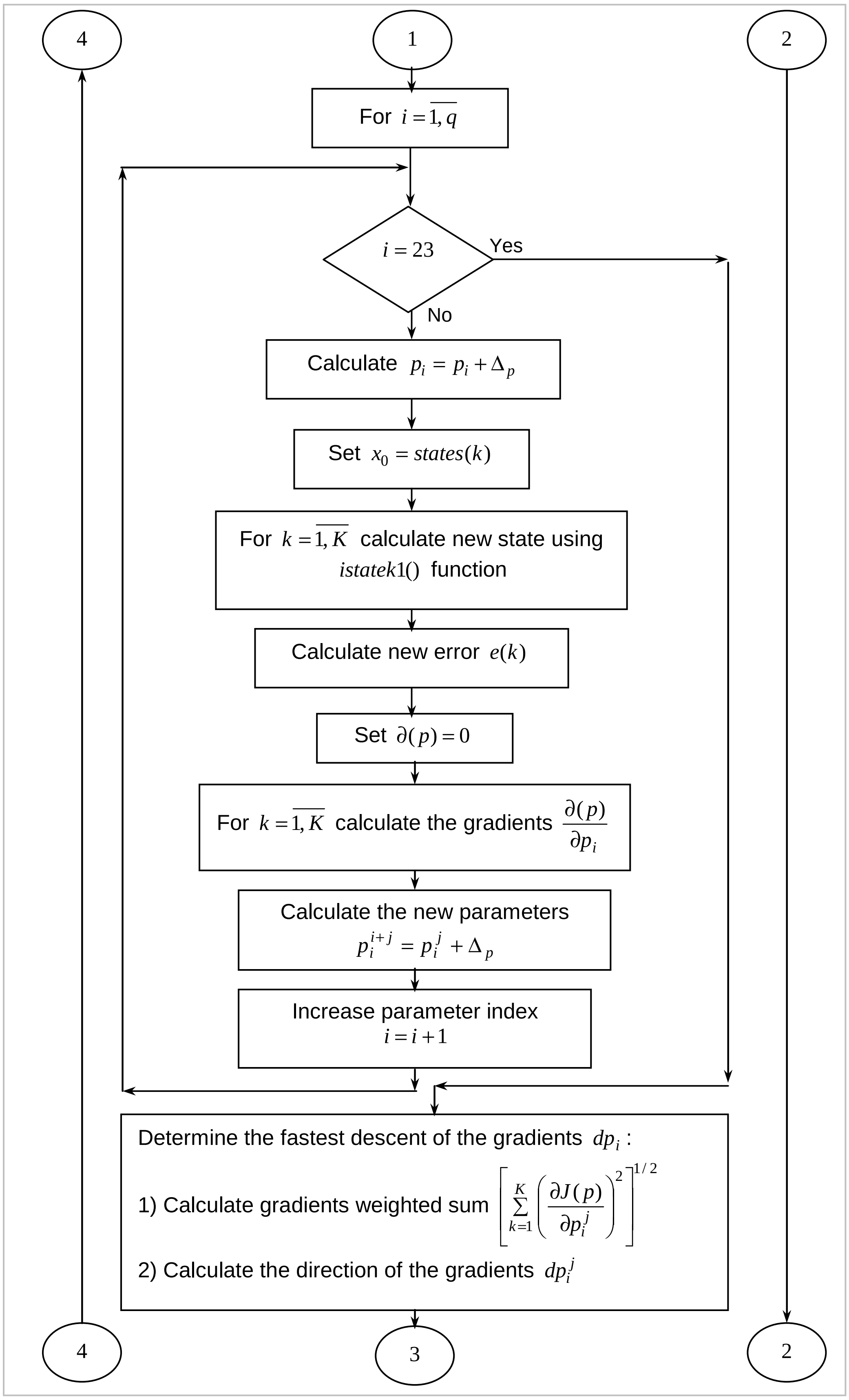
10.5. if at any point within the gradient procedure j , $j > M$, i.e., the procedure step has reached the maximum number of iterations required, the calculations must be stopped and the current values of parameters be used.

6.5.3. The flowchart for implementing the procedure

The flowchart of implementing the algorithm is presented in Figure 6.42.



Initialize parameters for the gradient procedure:
$x_0,\ M,\ K,\ \alpha_\theta,\ \varepsilon,\ j,\ p^j,\ q,\ \Delta_p$
Initialize process data:
$F_L, F_R, h, H, a, b, N, x_f$
Set the initial values of the parameters $p^j = p_0$
Calculate $p = [p_1, p_2, ..., p_m, ..., p_q]$ using p_0, Equation (6.9)
Solve for the state $x(k) \in R^{N\times K}$ using Equation (6.10)
Declare the measured state $\bar{x}(k) \in R^{N\times K} = x_{meas}(k) \in R^{N\times K}$
Calculate the criterion $J(p) = \sum_{k=1}^{K} \|\bar{x}(k) - x(k)\|^2$
$j \le M$
No
Yes
For $k = \overline{1, K}$
Set the arguments for the $istate()$ function
Call the $istate()$ function to solve $x(k) \in R^{N\times K}$
Calculate $J(p) = \sum_{k=1}^{K} \|e(k)\|^2$ using $\bar{x}(k)$ and $x(k)$
4
1
2


4
1
2
For $i=\overline{1,q}$
$i=23$
Yes
No
Calculate $p_i = p_i + \Delta_p$
Set $x_0 = states(k)$
For $k=\overline{1,K}$ calculate new state using $istatek1()$ function
Calculate new error $e(k)$
Set $\partial(p)=0$
For $k=\overline{1,K}$ calculate the gradients $\frac{\partial(p)}{\partial p_i}$
Calculate the new parameters $p_i^{i+j} = p_i^j + \Delta_p$
Increase parameter index $i=i+1$
Determine the fastest descent of the gradients dp_i:
1) Calculate gradients weighted sum $\left[\sum_{k=1}^{K}\left(\frac{\partial J(p)}{\partial p_i^j}\right)^2\right]^{1/2}$
2) Calculate the direction of the gradients dp_i^j
4
3
2

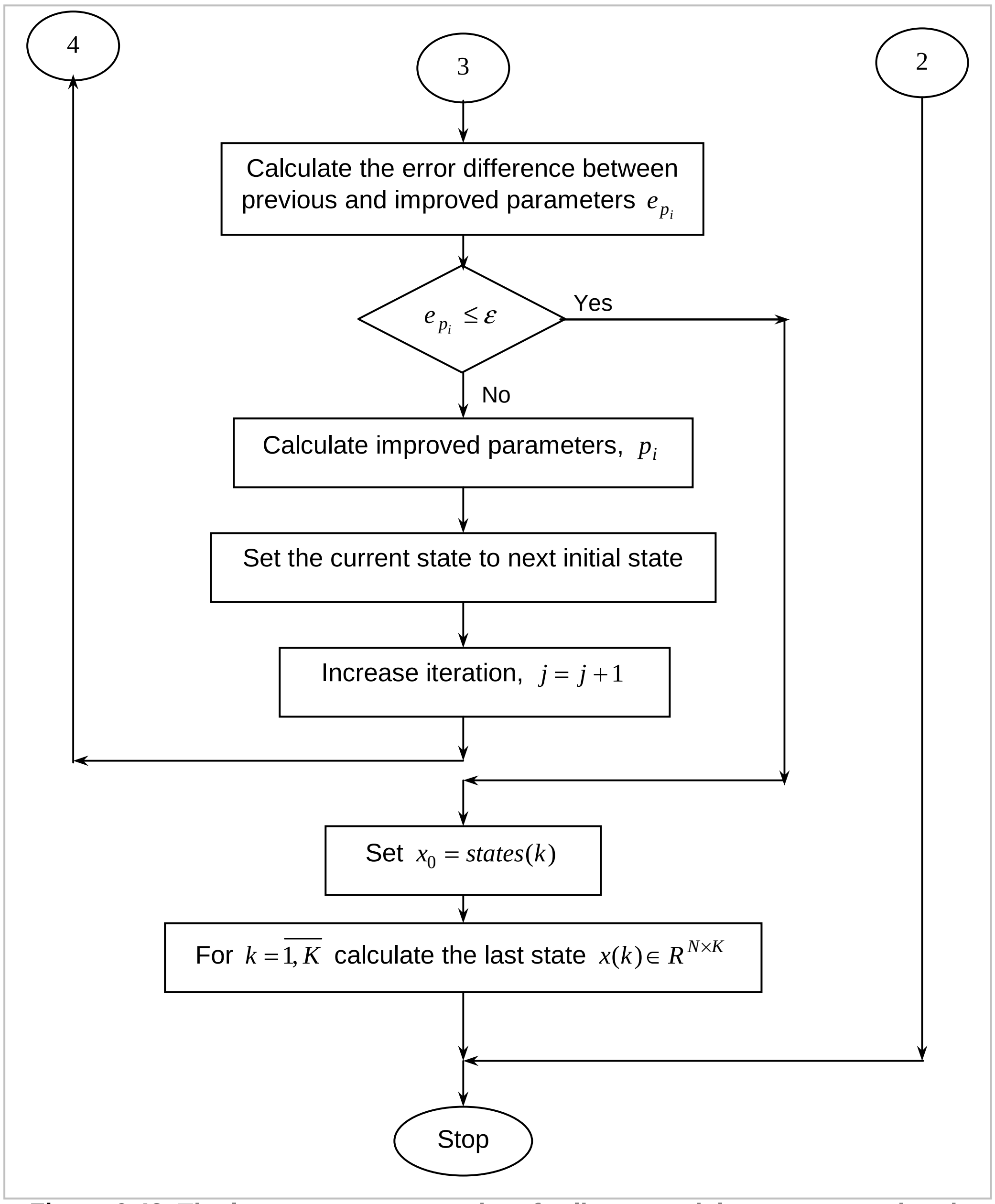

4
3
2
Calculate the error difference between previous and improved parameters e_{p_i}
$e_{p_i} \le \varepsilon$
Yes
No
Calculate improved parameters, p_i
Set the current state to next initial state
Increase iteration, $j = j + 1$
Set $x_0 = states(k)$
For $k = \overline{1, K}$ calculate the last state $x(k) \in R^{N \times K}$
Stop

Figure 6.42. The least squares procedure for linear model parameter estimation based on process state vector

6.5.4. The experiments and results of the linear model parameter estimation problem using the state vector measurements (the least squares method)

Different experiments were run to test the performance of the developed method and the algorithm convergence speed. Experiments monitoring the influence of *the sampling period* Δt, *the error* behaviour $e(k)$, *the state* behaviour $x(k)$, the *disturbance input* and *initial parameter values* p_0 were performed. The criterion for evaluating the success of the different experiment sets is based on the *least* *squares error* $J(p) = \sum_{k=1}^{K} e(k)e(k)^T$ and *convergence speed* of the algorithm, directly calculated from MATLAB during the algorithm processing. In all experiments $x_{\tilde{n}}^{\sim} = \hat{x}_n$, $n = \overline{1, N}$, Equation (6.16).

The first set of experiments of the least squares method

The first set experiments is the observation of the *gradient step* influence on the process using the best sampling period and the best coefficients a and b from the measured output method. The aim of the experiment is to find the best gradient step value for the best error correction in fastest time possible (Table 6.13). The results are presented in Figures 6.43–6.48 and Table 6.14.

Table 6.13: The first set of experiments of the changing value of the iteration gradient step

The changing iteration gradient step values						
Process parameter (units)	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6
$h = 32.93\ l$	✓	✓	✓	✓	✓	✓
$H = 42.809\ l$	✓	✓	✓	✓	✓	✓
$F_L = 2000/60\ l/\min$	✓	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓	✓
$a = [0.8\ 0.7\ 0.5\ 0.4\ 0.3\ 0.02]$	$a = 230 \times a$	✓	✓	✓	✓	✓
$b = [0.025\ 0.1\ 0.2\ 0.3\ 0.4\ 0.6]$	$b = 30 \times b$	✓	✓	✓	✓	✓
M	2000	✓	✓	✓	✓	✓
dt , sampling period (Δt)	0.012 min		✓	✓	✓	✓
Δ_p , parameter deviation	0.001	✓	✓	✓	✓	✓
$grad$, iteration *gradient step* (α)	0.01	0.035	0.076	0.1	1.2	2.5
ε , epsilon (very small number)	0.0001	✓	✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

The table shows the set of experiments for the changing *gradient step* value, starting from the smallest of 0.01 up to the last value of 2.5 in no specific order. The best model coefficients from set of experiments of Case 1 of the measured output method.

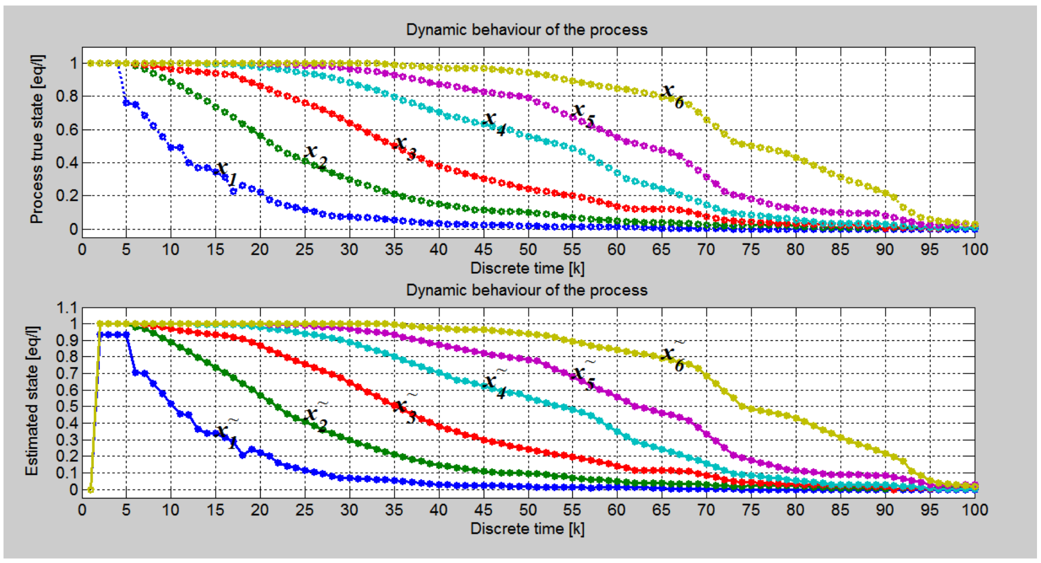


Figure 6.43. Model dynamic behaviour showing the real and estimated states of the best results – Run 1 Table 6.13

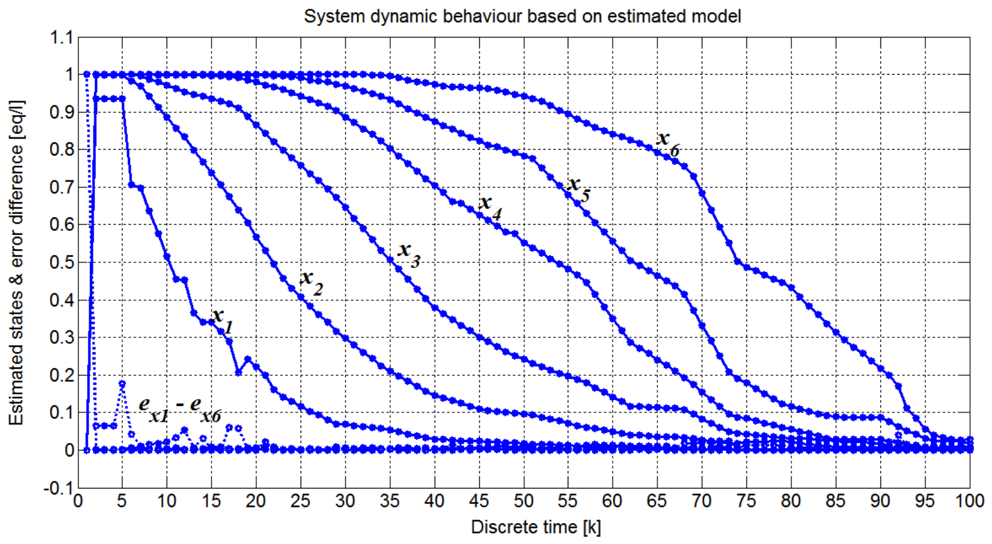


Figure 6.44. Model dynamic behaviour showing the error difference in relation to estimated states – the best case results – Run 1 Table 6.13

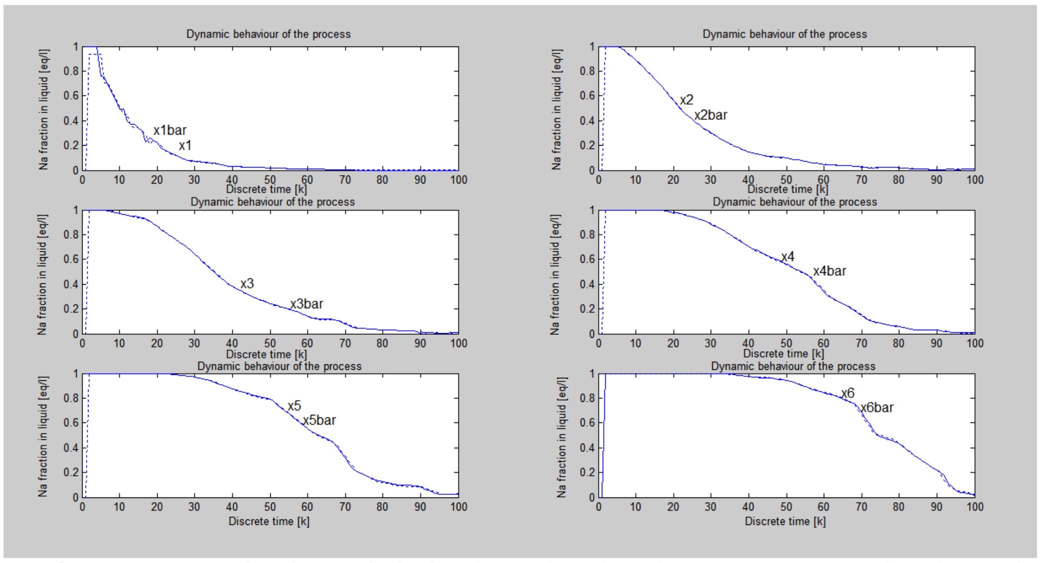


Figure 6.45. Model dynamic behaviour showing the overall real and estimated states – the best results – Run 1 Table 6.13

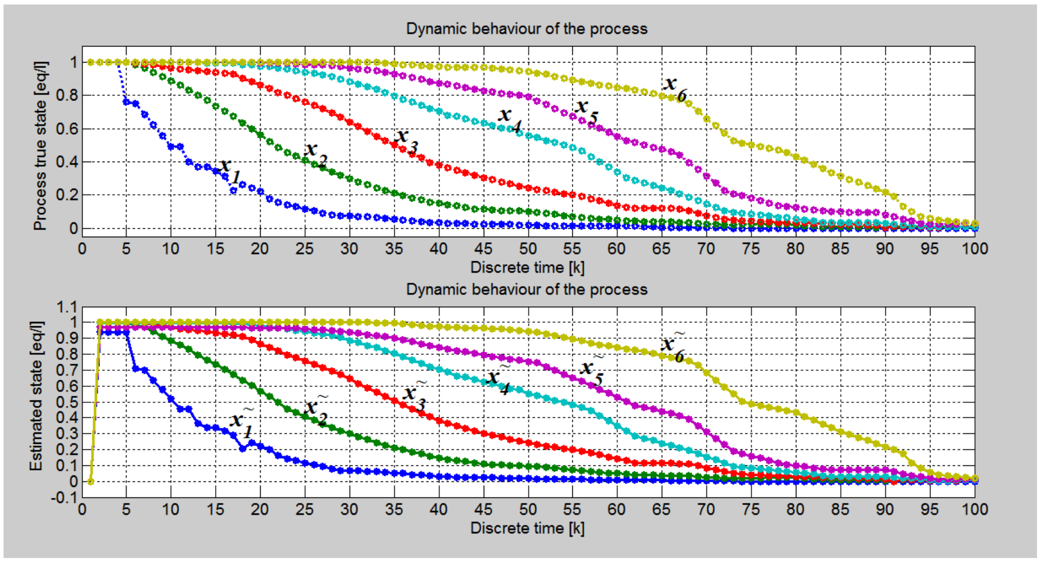


Figure 6.46. Model dynamic behaviour showing the real and estimated states of the worst case results – Run 6 Table 6.13

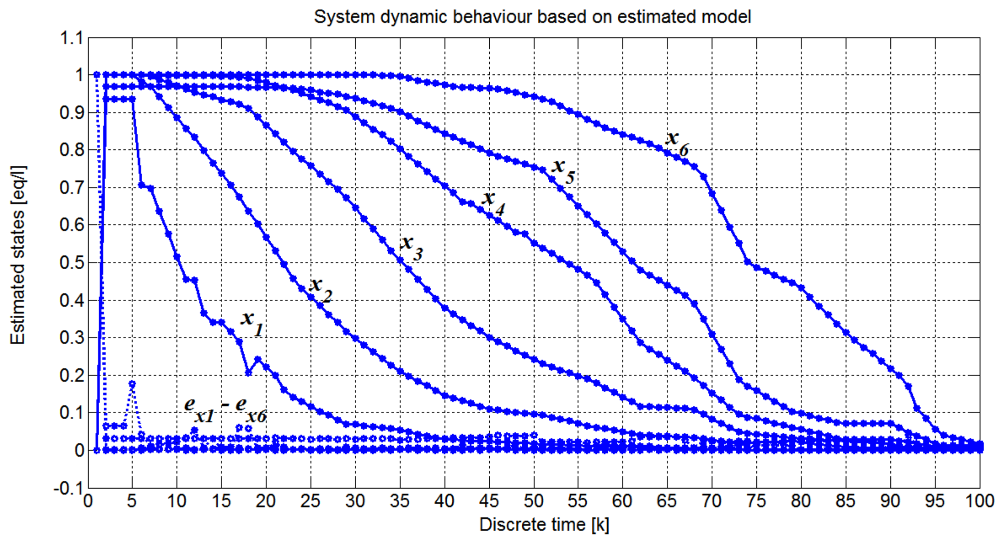


Figure 6.47. Model dynamic behaviour showing the error difference in relation to estimated states – the worst case results – Run 6 Table 6.13

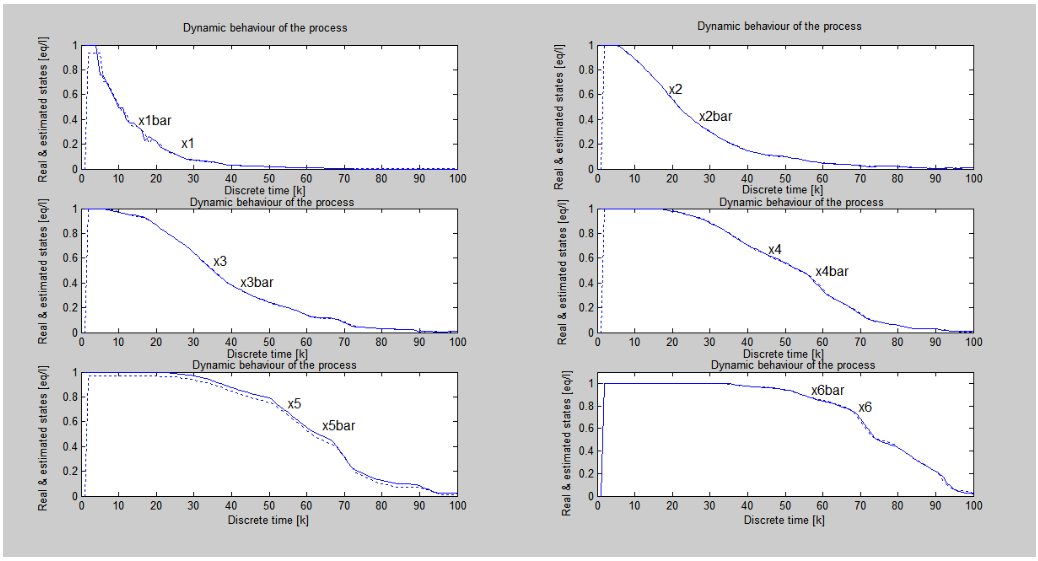


Figure 6.48. Model dynamic behaviour showing the overall real and estimated states – the worst case results – Run 6 Table 6.13

Table 6.14: Results of the first set of experiments of the changing gradient step

Estimation criterion the least squares error and the estimation run time for the first set of experiments		
Case 1: (First set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	6.0759	271.7600
Run 2	6.0758	283.2140
Run 3	6.0759	274.9940
Run 4	6.0759	257.8400
Run 5	6.0948	272.7450
Run 6	6.1459	259.7940

Results and discussion of the first set of experiments of the *least squares* method

The best results are obtained at the *gradient step* value of $\alpha = 0.035$, Figures 6.43–6.48. As the gradient step value goes higher, the error criterion also starts to inrease, Table 6.10. This means that at higher gradient values, the process behaves relatively poor, though for the given values the change is minute.

The second set of experiments of the least squares method

The second set of experiments considers the *changing sample period*, Table 6.15 and the sets of corresponding results are presented in Figures 6.49–6.54. The aim of this experiment is to evaluate the influence of the sampling period on the overall process behaviour and identify its best value for application in the process. The initial sampling period is set to $\Delta t = 2.35\,\text{min}$.

Table 6.15: **The second set of experiments of the changing sampling period**

The changing sampling period					
Process parameter (units)	Run 1	Run 2	Run 3	Run 4	Run 5
$h \ = 32.93 \ l$	✓	✓	✓	✓	✓
$H \ = 42.809 \ l$	✓	✓	✓	✓	✓
$F_L = 2000/60\,(l/\text{min})$	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓
$a = [0.8 \ 0.6 \ 0.5 \ 0.4 \ 0.3 \ 0.25]$	$a = 230 \times a$	✓	✓	✓	✓
$b = [0.01 \ 0.02 \ 0.05 \ 0.08 \ 0.1 \ 0.2]$	$b = 60 \times b$	✓	✓	✓	✓
M	2000	✓	✓	✓	✓
dt, sampling period (Δt)	2.35 min	$\Delta t = \frac{1}{2}\Delta t$	$\Delta t = \frac{1}{4}\Delta t$	$\Delta t = \frac{1}{16}\Delta t$	$\Delta t = \frac{1}{32}\Delta t$
Δ_p, parameter deviation	0.001	✓	✓	✓	✓
$grad$, gradient step (α)	0.076	✓	✓	✓	✓
ε, epsilon (very small number)	0.0001	✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

The table shows the changing sampling period, it is changed at $1/n^2$ for $n = \overline{1,5}$.

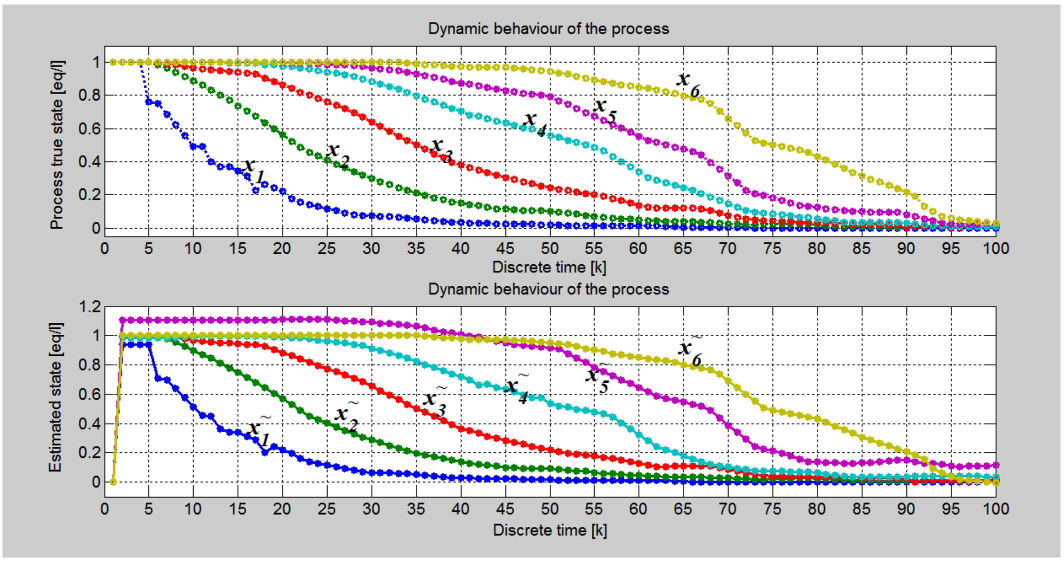


Figure 6.49. Model dynamic behaviour showing the real and estimated states of the worst case results – Run 1 Table 6.15

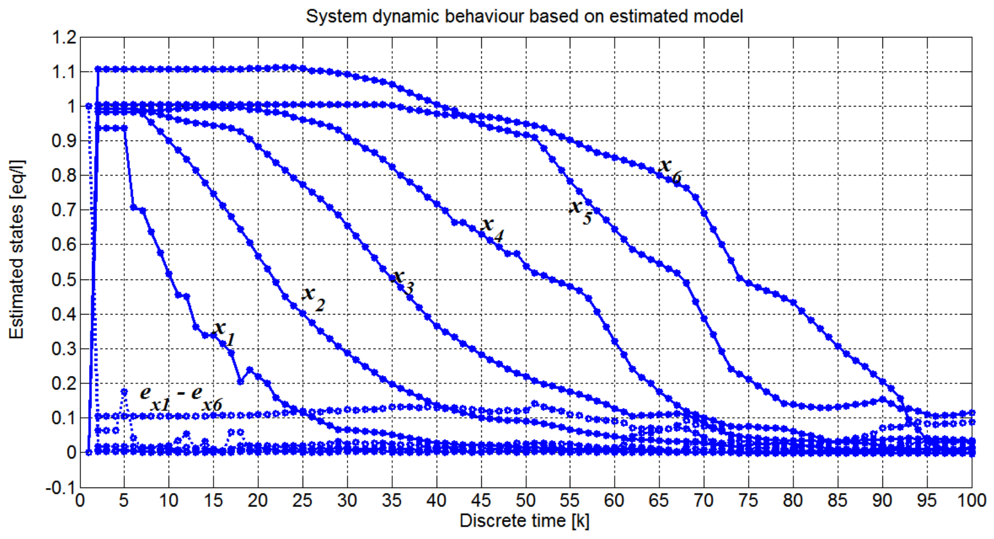


Figure 6.50. Model dynamic behaviour showing the error difference in relation to estimated states – the worst case results – Run 1 Table 6.15

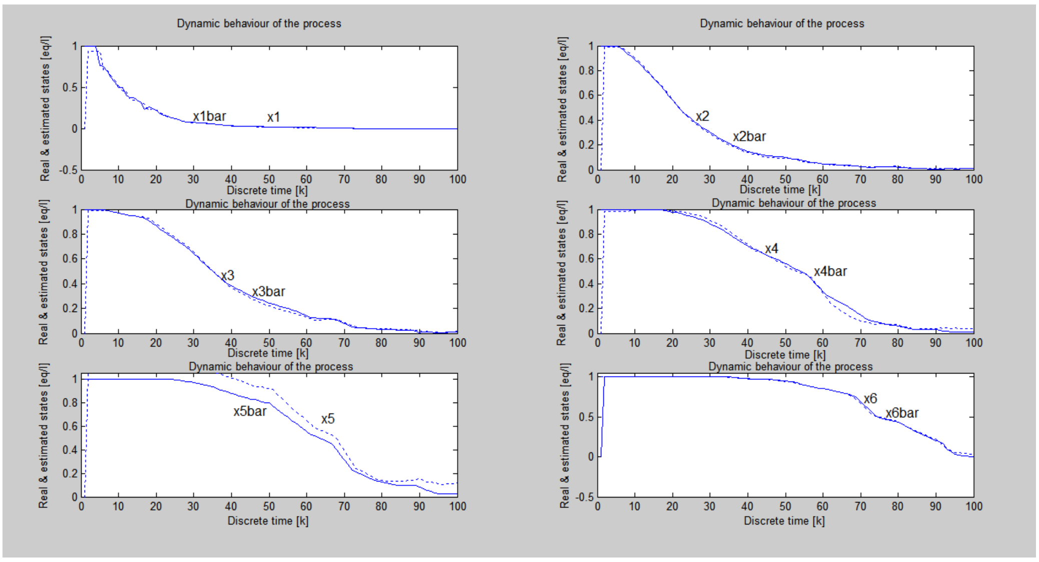


Figure 6.51. Model dynamic behaviour showing the overall real and estimated states – the worst case results –– Run 1 Table 6.15

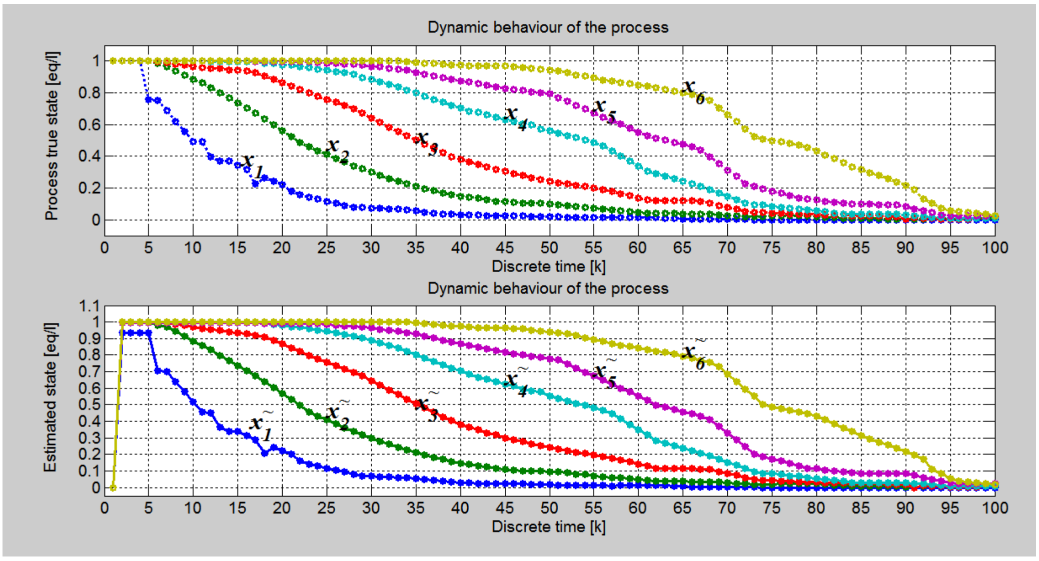


Figure 6.52. Model dynamic behaviour showing the real and estimated states of the best results – Run 5 Table 6.15

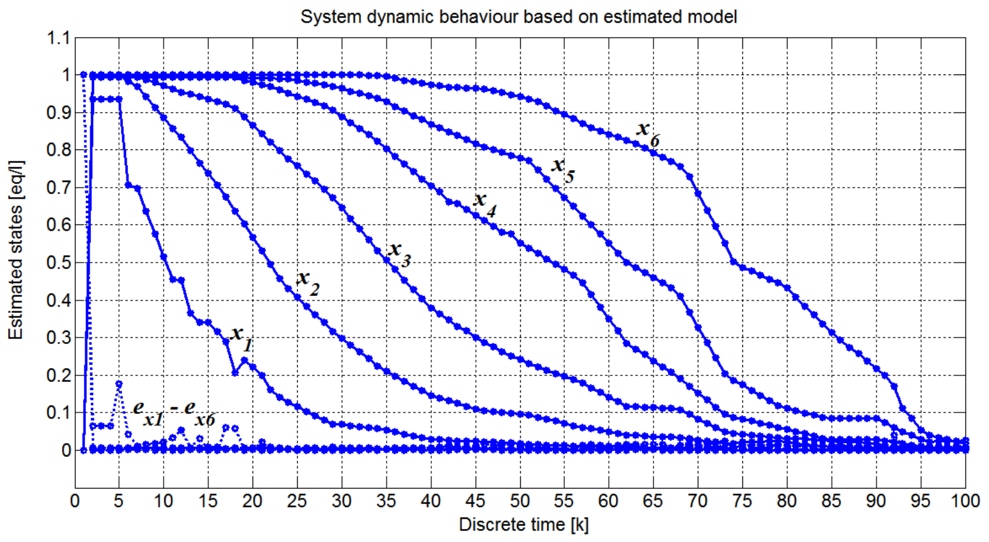


Figure 6.53. Model dynamic behaviour showing the error difference in relation to estimated states – the best results – Run 5 Table 6.15

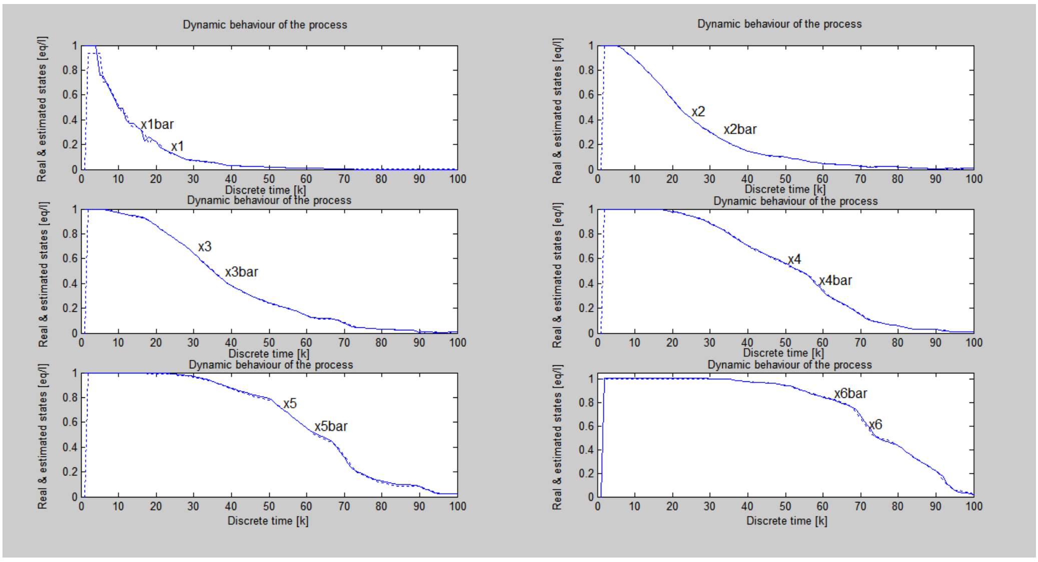


Figure 6.54. Model dynamic behaviour showing the overall real and estimated states – the best results – Run 5 Table 6.15

Table 6.16: Results of second set of experiments for the changing sampling period Δt

Estimation criterion the least squares error and the estimation run time for the second set of experiments		
Case 2: (Second set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	7.1148	274.5560
Run 2	6.4099	227.3860
Run 3	6.1996	272.1070
Run 4	6.0867	267.2340
Run 5	6.0786	234.2510

Results and discussion of the second set of experiments for the least squares method

The process performs best at higher sampling periods, Table 6.16. Though the best value to produce acceptable error is at $\Delta t = {}^{1}\!/_{32}\,\Delta t$ for practicality, in the rest of the experiments, the value is reduced even further 0.012 min, which equals to $\Delta t = {}^{1}\!/_{195}\,\Delta t$.

The third set of experiments of the least squares method

The third set of experiments is for observing the *influence of the disturbance input* on the process Table 6.17 and the sets of corresponding results are presented in Figures 6.55–6.58. The aim of this set of experiments is to determine the critical disturbance input value that should be monitored for corrective measures to be taken when necessary.

Table 6.17: The third set of experiments for monitoring the influence of the disturbance input on the process

The changing disturbance input signal amplitude						
Process parameter (units)	Value/ (Reference)	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 32.93\ l$	✓	✓		✓	✓	✓
$H = 42.809\ l$	✓	✓		✓	✓	✓
$F_L = 2000/60\ (l/\text{min})$	✓	✓		✓	✓	✓
x_f, disturbance input	$x_f = 0.435$	$x_f = 0.055$	$x_f = 0.110$	$x_f = 0.55$	$x_f = 0.75$	$x_f = 1.00$
$a = [0.8\ 0.6\ 0.5\ 0.4\ 0.3\ 0.25]$	$a = 230 \times a$	✓		✓	✓	✓
$b = [0.01\ 0.02\ 0.05\ 0.08\ 0.1\ 0.2]$	$b = 60 \times b$	✓		✓	✓	✓
M	2000	✓		✓	✓	✓
dt, sampling period (Δt)	0.012 min	✓		✓	✓	✓
Δ_p, parameter deviation	0.001	✓		✓	✓	✓
$grad$, gradient step (α)	0.82	✓		✓	✓	✓
ε, epsilon (very small number)	0.0001	✓		✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

In this set of experiments the investigation is to observe the influence of the disturbance signal on the process and to identify the worse and optimum cases of process dynamics based on the disturbance values.

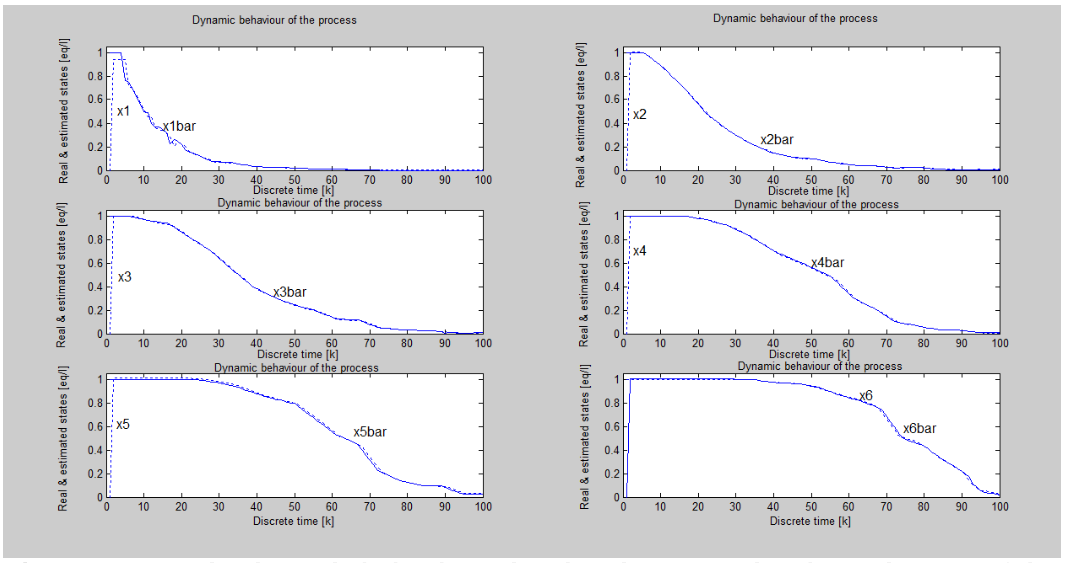


Figure 6.55. Model dynamic behaviour showing the real and estimated states of the best results – Run 1 Table 6.17

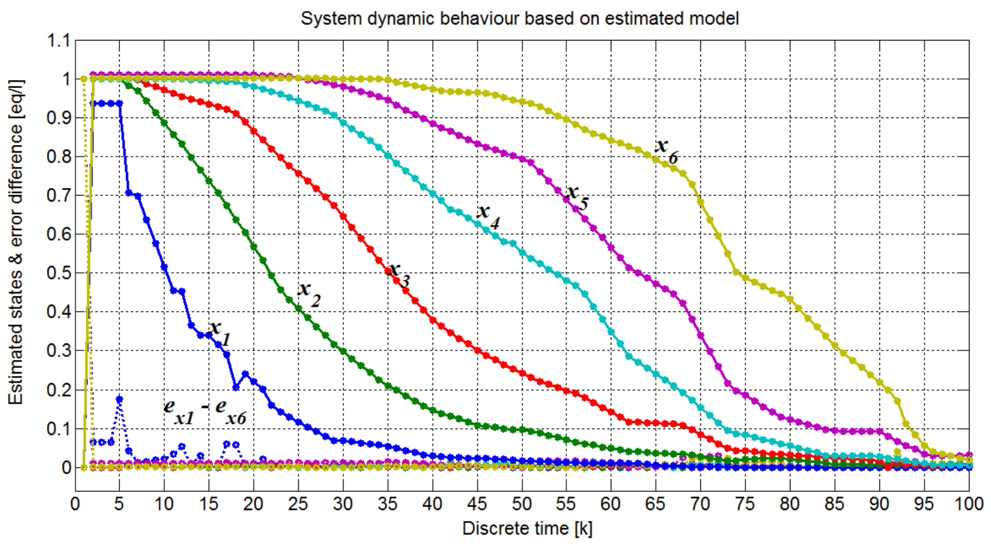


Figure 6.56. Model dynamic behaviour showing the error difference in relation to estimated states – the best results – Run 1 Table 6.17

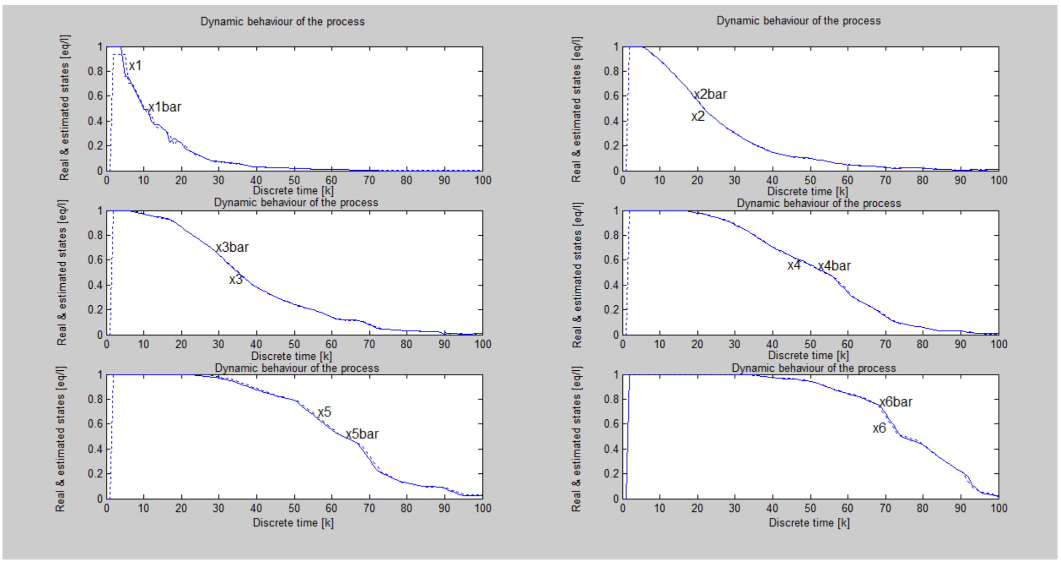


Figure 6.57. Model dynamic behaviour showing the real and estimated states of the worst performing disturbance input – Run 4 Table 6.17

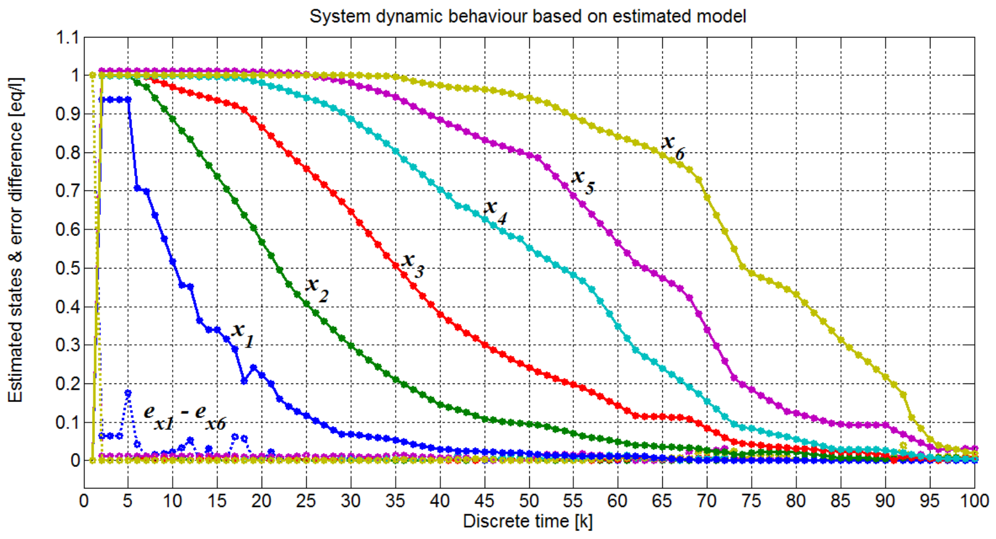


Figure 6.58. Model dynamic behaviour showing the error difference in relation to estimated states for the worst performing disturbance input – Run 4 of Table 6.17

Table 6.18: Results of third set of experiments for the changing disturbance input amplitude x_f

Estimation criterion the least squares error and the estimation run time for the third set of experiments		
Case 3: (Third set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	6.0839	241.7610
Run 2	6.0839	247.6290
Run 3	6.0840	244.6560
Run 4	6.0840	249.2570
Run 5	6.0840	241.7830

Results and discussion of the third set of experiments for the least squares method

For all the experiments of the third set, the states produced the tracking of the states as shown in Figure 6.55 and Figure 6.57. This procedure produced best results at the lowest disturbance value, $x_f = 0.055$, but the next best result is obtained at the highest value of the disturbance $x_f = 1.00$. Further, this proves that the model is responding well to the disturbances. Considering the *error least squares* criterion, there is the negligible difference in the error for all the runs; only the calculation time differed but also by infinitesimal amounts Table 6.18. The relatively "*worst case scenario*" is obtained at $x_f = 0.755$, in comparison to the rest.

The fourth set of experiments of the least squares method

The fourth set of experiments considers the *changing initial values of model parameters* based on the best model coefficients of a and b. Table 6.19 shows the changing variables and the sets of corresponding results are presented in Figures 6.59–6.62. The aim of this experiment is to identify the best initial model parameters at the best sampling period.

Table 6.19: **The fourth set of experiments of the changing model parameter initial conditions**

Changing the model parameter initial conditions					
Process parameter (units)	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 32.93\ l$	✓	✓	✓	✓	✓
$H = 42.809\ l$	✓	✓	✓	✓	✓
$F_L = 2000/60\ (l/\min)$	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓
$a = [0.8\ 0.6\ 0.5\ 0.4\ 0.3\ 0.25]$	$a = 230 \times a$	✓	✓	✓	✓
$b = [0.01\ 0.02\ 0.05\ 0.08\ 0.1\ 0.2]$	$b = 60 \times b$	✓	✓	✓	✓
p_0 , initial model parameters	$p_0 = 0.1 \times p_0$	$p_0 = 0.5 \times p_0$	$p_0 = 1.0 \times p_0$	$p_0 = 2.0 \times p_0$	$p_0 = 10 \times p_0$
M	2000	✓	✓	✓	✓
dt , sampling period (Δt)	0.012 min	✓	✓	✓	✓
Δ_p , parameter deviation	0.001	✓	✓	✓	✓
$grad$, gradient step (α)	0.82	✓	✓	✓	✓
ε , epsilon (very small number)	0.0001	✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

The table shows the changing initial conditions from 0.1 to 10 in five runs.

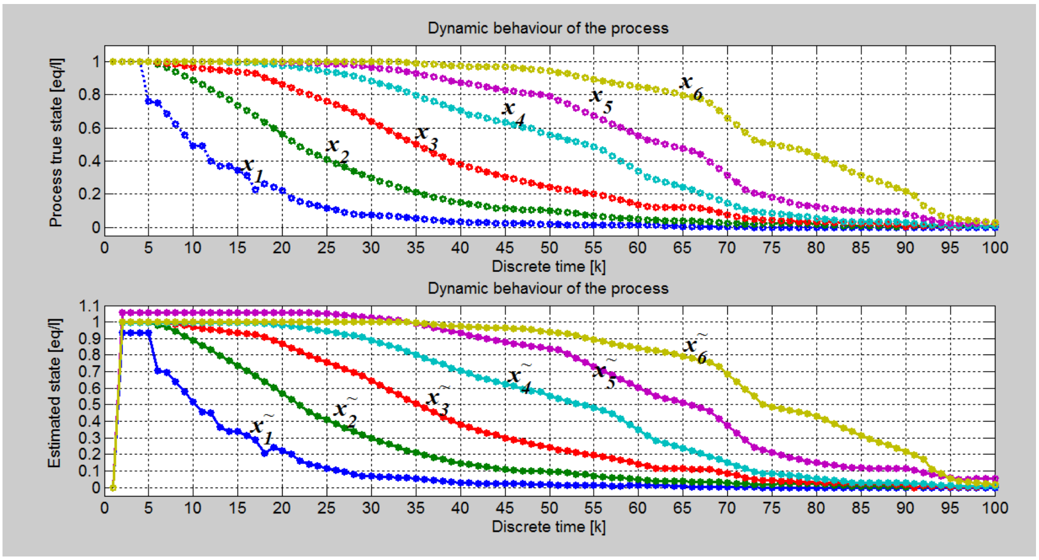


Figure 6.59. **Model dynamic behaviour showing the real and estimated states of the best results – Run 1 Table 6.19**

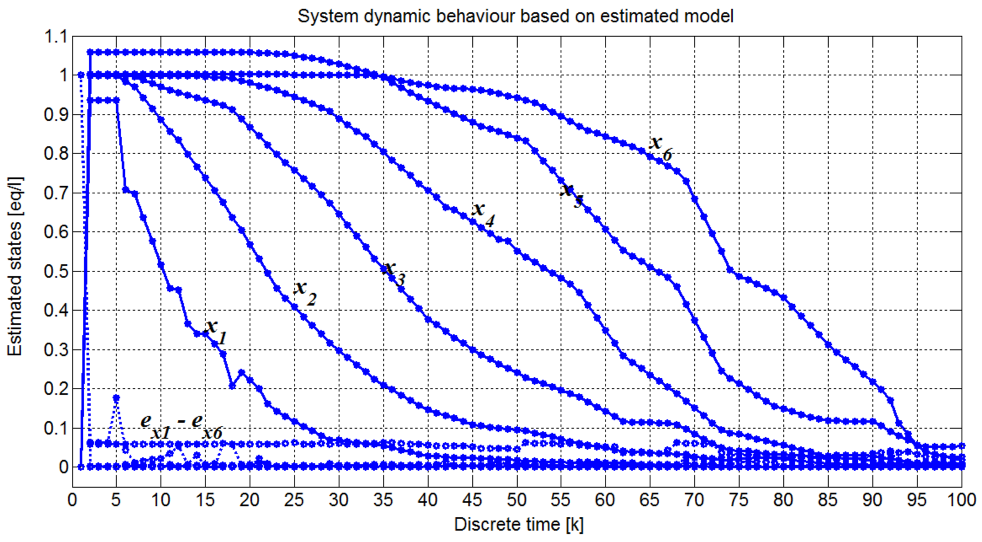


Figure 6.60. **Model dynamic behaviour showing the error difference in relation to estimated states – the best case results – Run 1 Table 6.19**

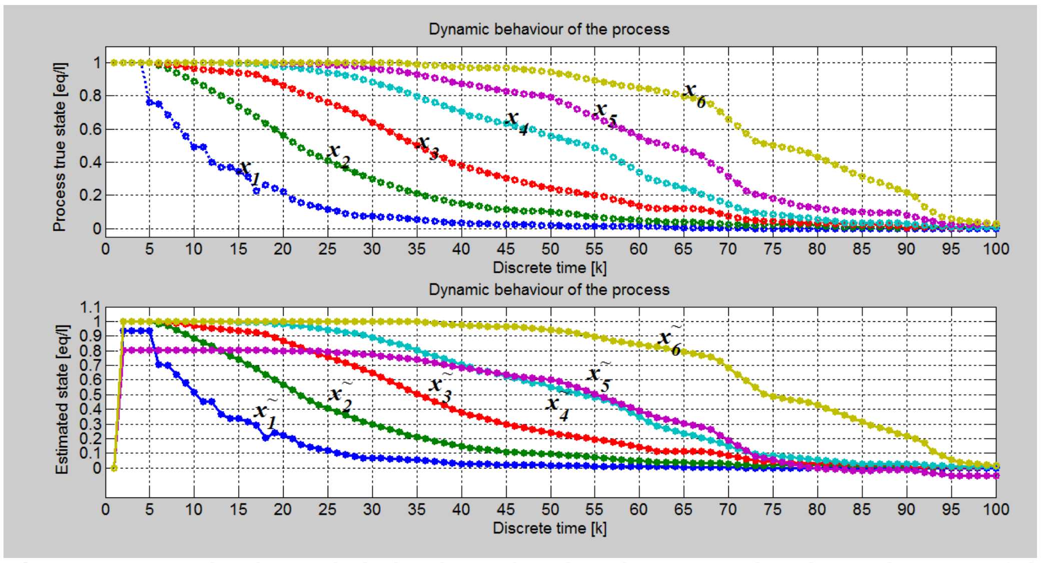


Figure 6.61. Model dynamic behaviour showing the real and estimated states of the worst case results – Run 5 Table 6.19

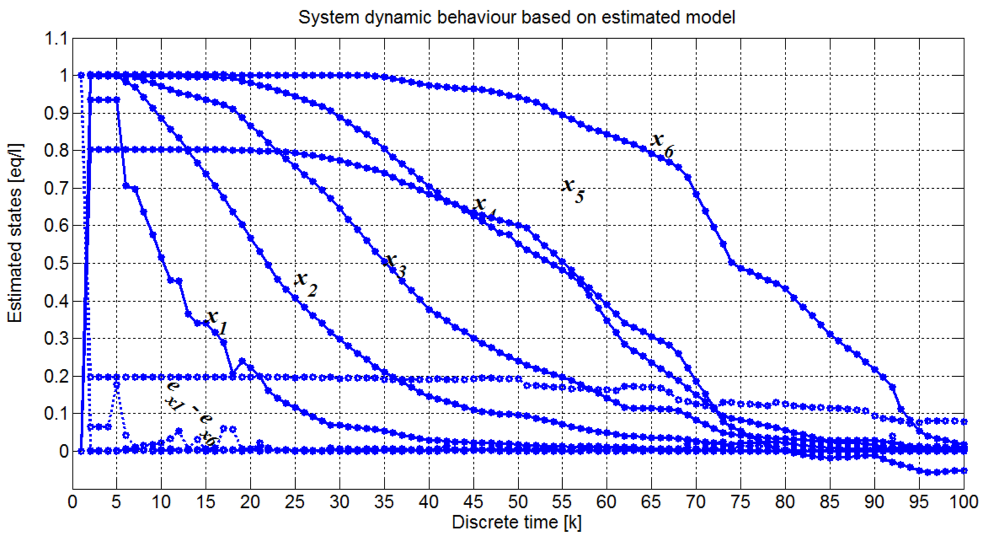


Figure 6.62. Model dynamic behaviour showing the error difference in relation to estimated states – the worst case results – Run 5 Table 6.19

Table 6.20: **Results of the fourth set of experiments of the changing initial parameter values**

Estimation criterion the least squares error and the estimation run time for the third set of experiments		
Case 4: (Fourth set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	8.8236	231.6910
Run 2	8.7964	234.1340
Run 3	8.8446	268.6780
Run 4	8.8014	270.5910
Run 5	6.3211	269.9240
Extra values tested:		
$p_0 = 100 \times p_0$	7.3261	234.1560
$p_0 = 250 \times p_0$	7.3382	234.3010

Results and discussion of the fourth set of experiments for the least squares method

The best error criterion is obtained at higher initial values of the parameters, but at higher run time, Table 6.20. Though the run time is the highest, the rest of run times are relatively high as well. This makes the best result an acceptable even if the run time is the highest.

6.6. Linear model parameter estimation using the state vector measurements (direct method)

This method of solution for parameter estimation is derived from the case where all states are measureable. In this case data that was obtained from Hendry, (1982d) is for all stages in the CCIX (continuous countercurrent ion exchange) process, which makes it possible to apply this method of solution. In this case, the state variables must be measured, the process input and the discrete states determined from the measured data are also available and then parameters can be determined from this data.

From the developed process model, using Equation (6.6), the following representation has been derived considering that the parameter vector $p(t)$ is unknown

$$\dot{x}(t) = F(x(t), F_R(t), t)\, p(t) \tag{6.17}$$

where $p(t)$ is the vector of unknown parameters that must be determined.

In its expanded form, using the discrete form, Equation (6.19) can be expressed as Equation (6.18):

$$x_n(k+1) = \begin{bmatrix} x_1(k) & x_2(k) & x_3(k) & x_4(k) & x_5(k) & x_6(k) \end{bmatrix}^T +$$

$$+\Delta t \begin{bmatrix} (x_{in}(k)-x_1(k)) & x_2(k)F_R(k) & -x_1(k)F_R(k) & F_R(k) & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & (x_1(k)-x_2(k)) & x_3(k)F_R(k) & -x_2(k)F_R(k) & F_R(k) & \dots & 0 \\ 0 & \dots & 0 & (x_{n-1}(k)-x_n(k)) & x_{n+1}(k)F_R(k) & -x_n(k)F_R(k) & F_R(k) & \dots & & 0 \\ 0 & \dots & & \dots & & 0 & (x_{N-1}(k)-x_N(k)) & -x_N(k)F_R(k) & F_R(k) & \end{bmatrix}$$

$$\times \begin{bmatrix} p_1(k) & p_2(k) & \dots & p_m(k) & \dots & p_q(k) \end{bmatrix}^T, \qquad n = \overline{1, N} \tag{6.18}$$

Equation (6.18) can also be expressed as:

$$\frac{x(k+1)-x(k)}{\Delta t} = F(k)p(k) \tag{6.19}$$

where

$F(k)$ – is the coefficients matrix for the state and bilinear terms of the process as shown in Equation (6.18).

$$F(k) =$$

$$\begin{bmatrix} (x_{in}(k)-x_1(k)) & x_2(k)F_R(k) & -x_1(k)F_R(k) & F_R(k) & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & (x_1(k)-x_2(k)) & x_3(k)F_R(k) & -x_2(k)F_R(k) & F_R(k) & & 0 \\ 0 & \dots & \ddots & \dots & \ddots & \dots & \ddots & \dots & & 0 \\ 0 & \dots & 0 & (x_{n-1}(k)-x_n(k)) & x_{n+1}(k)F_R(k) & -x_n(k)F_R(k) & F_R(k) & & & 0 \\ 0 & \dots & & \dots & & 0 & (x_{N-1}(k)-x_N(k)) & -x_N(k)F_R(k) & F_R(k) & \end{bmatrix}$$

The solution of Equation (6.21) in shortened version can be written as Equation (6.22) below.

$$F(k)^T \left[\frac{x(k+1)-x(k)}{\Delta t}\right] = F(k)^T F(k) p(k) \tag{6.20}$$

$$p(k) = \left[F(k)^T F(k)\right]^{-1} E(k) \tag{6.21}$$

where $E(k) = F(k)^T \left[\frac{x(k+1)-x(k)}{\Delta t}\right]$ (6.22)

The solution is implemented in MATLAB software based on the matrix division operator (\). This operator is said to be more accurate and time efficient in calculating a quotient of a matrix. The operator is based on Guassian elimination and it does not implement inverse approach, this helps to eliminate the matrix inverse singularity problem.

6.6.1. The algorithm and flowchart for implementing the direct method

1. The following process data values must be initialized:

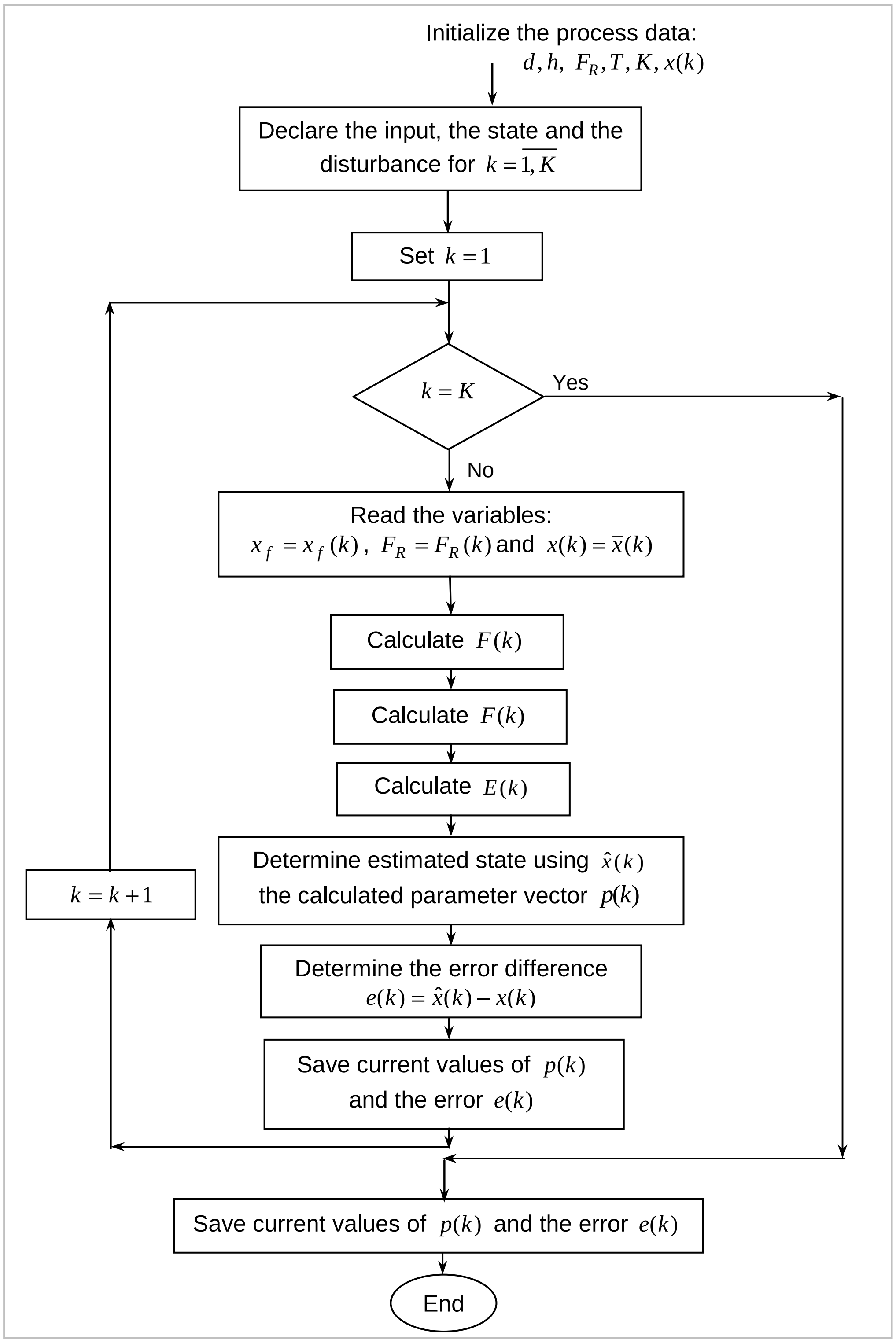
d	the resin/liquid ionic fraction ratio
h	the resin holdups per column diameter,
N	total number of stages per column,
T	process upflow period,
$u(k) = F_R$	the resin flow rate entering the top stage of the load column,
x_f	feed concentration of sodium in the liquid entering the column,
$\overline{x}(k) \in R^{N \times K}$	measured data
Δt	sampling period.

2. Set the following parameters for the process trajectory:

x_0	initial values of the state vector,
K	the number of discrete time steps in the optimization period.

3. Project the state, the disturbance and control input for full observation trajectory $k = \overline{1, K}$: $F_R \in R^{1 \times K}$, $x_f \in R^{1 \times K}$ and $x_0 = \overline{x}(k)$.
4. For the full trajectory $k = \overline{1, K}$ for each discrete moment:
   - Set the current input, $F_R = F_R(k)$,
   - Set the current state, $x(k) = \overline{x}(k)$,
   - Set the current disturbance, $x_f = x_f(k)$,
   - Calculate the $F(k)$ matrix,
   - Calculate the $E(k)$ matrix,
   - Calculate the parameter vector $p(k)$, Equation (6.23),
   - Calculate the estimated states $\hat{x}(k) \in R^{6 \times k}$ using estimated parameter values,
   - Calculate the error difference between estimated and measured states $e(k) = \hat{x}(k) - x(k)$.
5. Repeat from step 4 until the maximum number of discrete moments is reached.

The flowchart of the implementation of this method is presented in Figure 6.63.



Initialize the process data: $d, h, F_R, T, K, x(k)$
Declare the input, the state and the disturbance for $k = \overline{1, K}$
Set $k = 1$
$k = K$
Yes
No
Read the variables: $x_f = x_f(k)$, $F_R = F_R(k)$ and $x(k) = \bar{x}(k)$
Calculate $F(k)$
Calculate $F(k)$
Calculate $E(k)$
Determine estimated state using $\hat{x}(k)$ the calculated parameter vector $p(k)$
$k = k + 1$
Determine the error difference $e(k) = \hat{x}(k) - x(k)$
Save current values of $p(k)$ and the error $e(k)$
Save current values of $p(k)$ and the error $e(k)$
End

Figure 6.63. The procedure for linear model parameter estimation based on measured state vector

6.6.2. Experiments and results for the linear model parameter estimation problem using measurements of the state vector (the direct method)

In the set of experiments conducted under the direct method, the following variables are varied: the sampling period Δt, the control input F_R and the disturbance signal x_f. The evaluation criteria for each set of experiments are the least squares error $J(p) = \sum_{k=1}^{K} e(k)e(k)^T$ and the calculation run time. The *direct method* does not allow too many parameters to be varied, thus a lesser number of experiments in comparison to other methods. In all experiments' results $x_n bar = \bar{x}_n$, $x_n{}^{\sim} = x_n(k)$ for $n = \overline{1, N}$, Equation (6.6) and Equation (6.18).

The first set of experiments of the the direct method

The first set of experiments considers the *changing values of the sampling period* at constant control input, Table 6.21. The results are presented in Figures 6.64–6.71 and Table 6.22. The aim of this set of experiments is to identify the best sampling period for the direct estimation method. The evaluation criterion is based on the estimation error and the calculation run time for the rest of the experiments.

Table 6.21: **The first set of experiments of the changing sampling period at constant control input**

Changing sampling period at constant control input						
Process parameter (units)	Values	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 32.93\ l$	✓	✓	✓	✓	✓	✓
$d = 2/3$ (r/l ratio)	✓	✓	✓	✓	✓	✓
T (liquid upflow period)	$T = 17\,\mathrm{min}$	✓	✓	✓	✓	✓
$F_R = (d \times h)/T$	$F_R = 0.3049$	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓	✓
dt, sampling period (Δt)	2.35 min	$\Delta t = \frac{1}{2}\Delta t$	$\Delta t = \frac{1}{16}\Delta t$	$\Delta t = \frac{1}{64}\Delta t$	$\Delta t = \frac{1}{128}\Delta t$	$\Delta t = \frac{1}{195}\Delta t$

✓ means the value stays the same as in the previous mention in the table

The table shows the changing sampling period, it is changed at $1/2(2)^n$ for $n = 0, 3, 5, 6$ and $\Delta t = \frac{1}{195}\Delta t$.

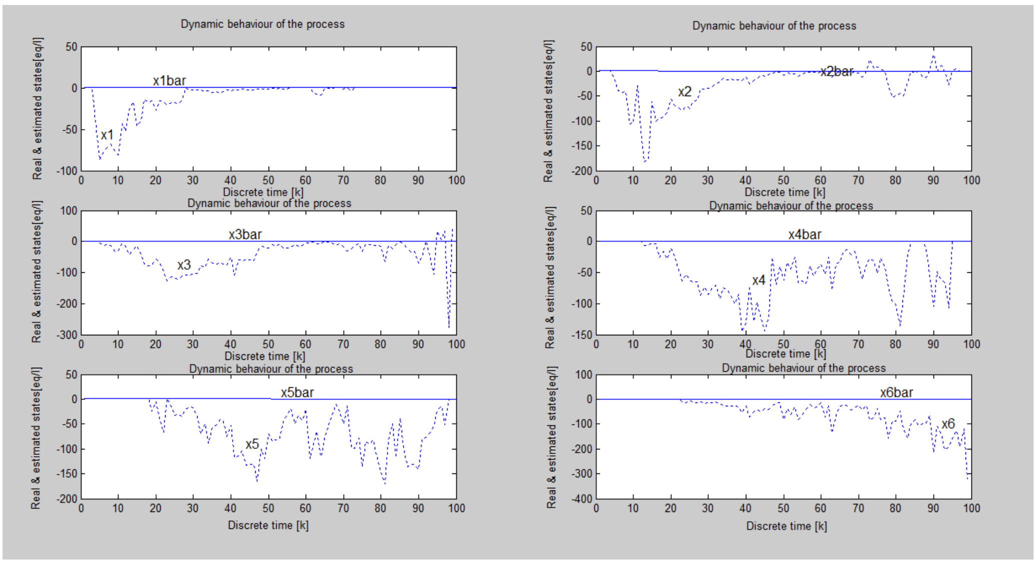


Figure 6.64. Model dynamic behaviour showing the overall real and estimated states of the worst case results at $\Delta t = \frac{1}{2}\Delta t$ Run 1 Table 6.21

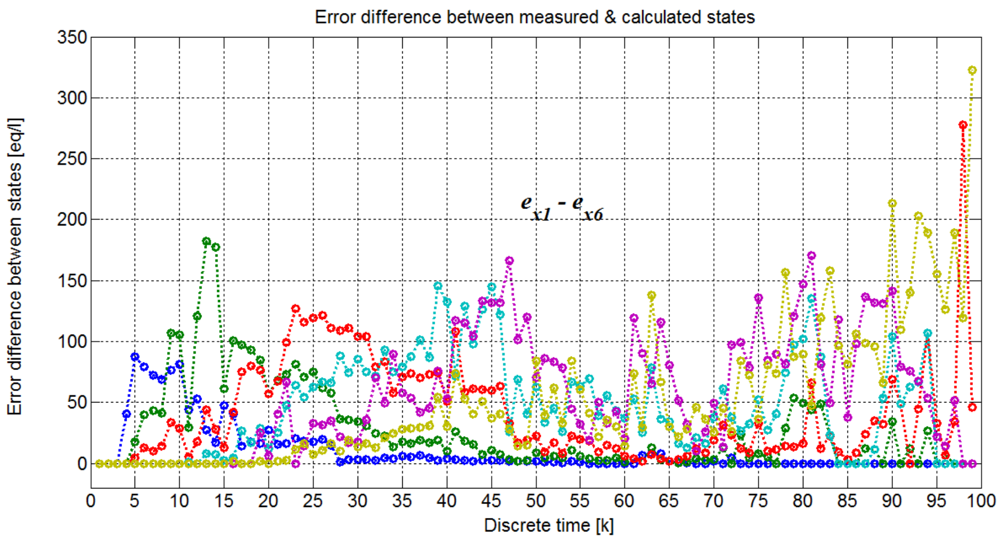


Figure 6.65. The error signal in detail for the worst case results – Run 1 of Table 6.21

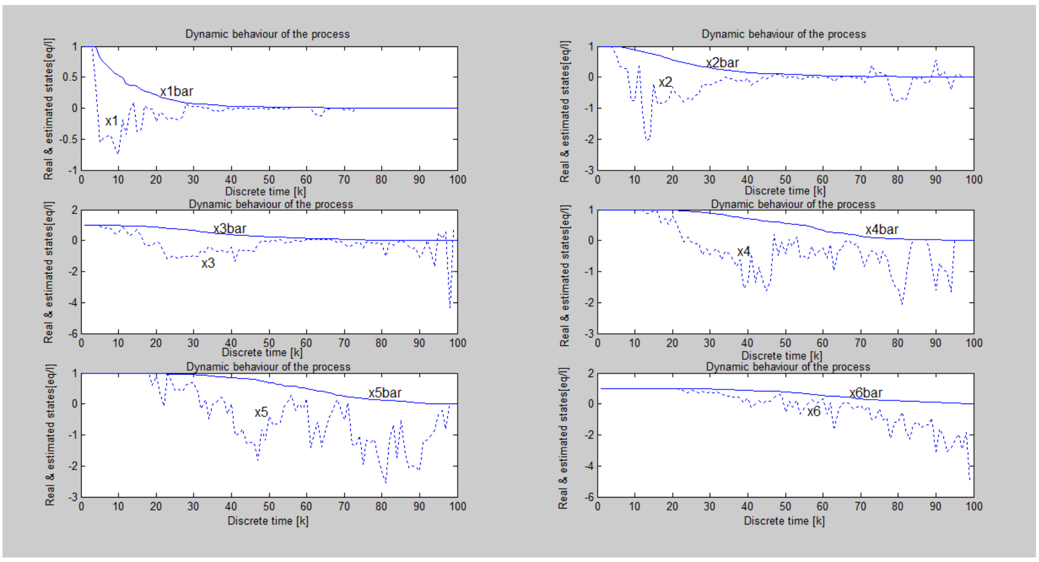


Figure 6.66. Model dynamic behaviour showing the overall real and estimated states at $\Delta t = \frac{1}{16}\Delta t$ Run 2 Table 6.21

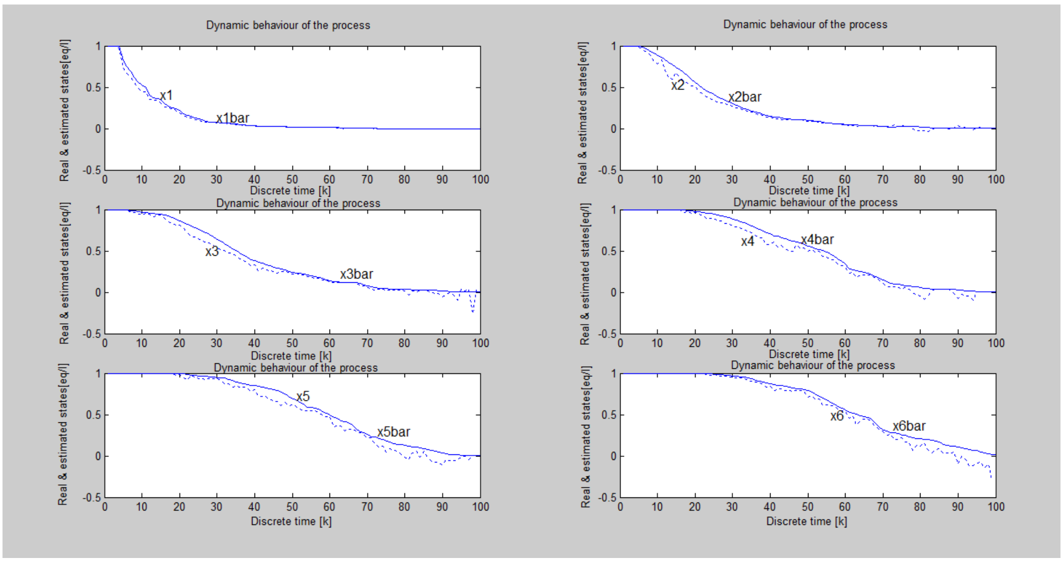


Figure 6.67. Model dynamic behaviour showing the overall real and estimated states at $\Delta t = \frac{1}{64}\Delta t$ Run 3 Table 6.21

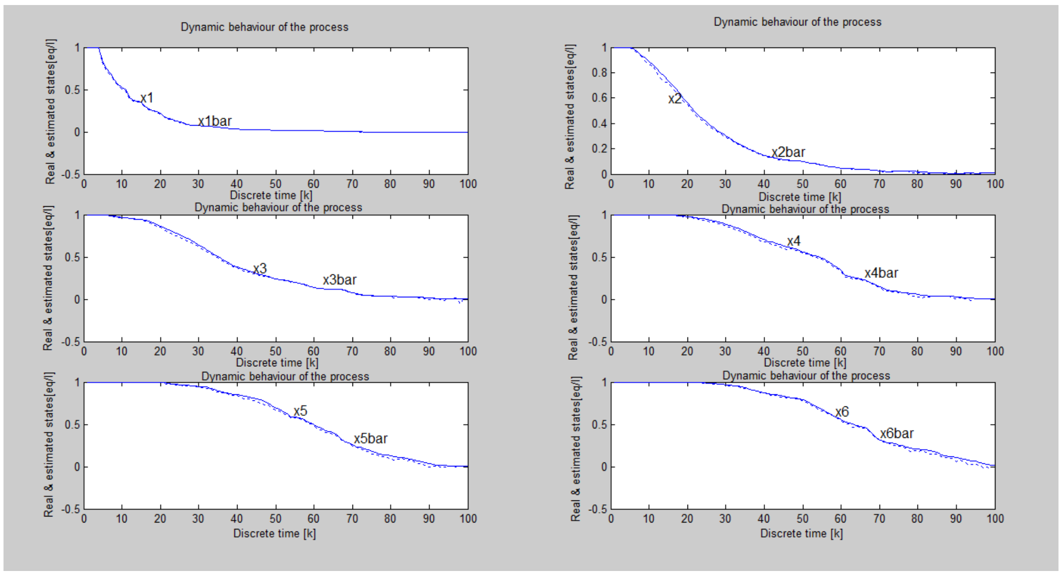


Figure 6.68. Model dynamic behaviour showing the overall real and estimated states at $\Delta t = \frac{1}{128}\Delta t$ Run 4 Table 6.21

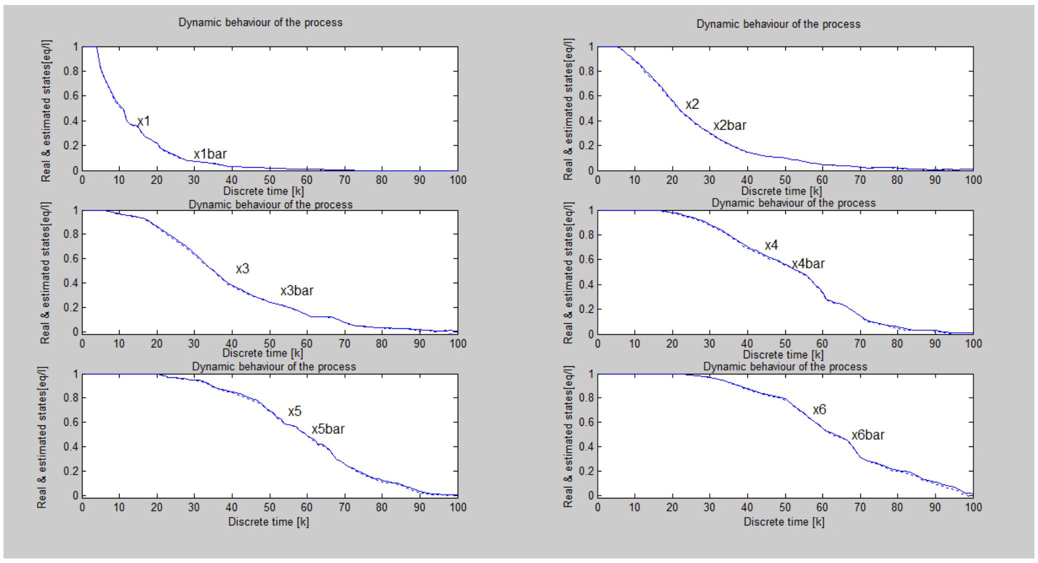


Figure 6.69. Model dynamic behaviour showing the overall real and estimated states for the best results at $\Delta t = \frac{1}{195}\Delta t$ Run 5 Table 6.21

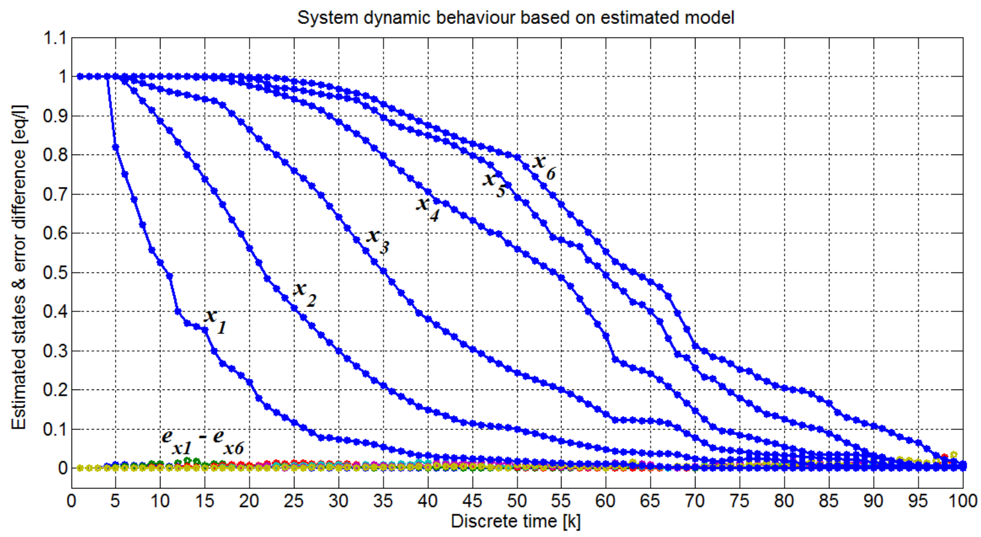


Figure 6.70. Model dynamic behaviour showing the error difference in relation to estimated states for the best case results – Run 5 Table 6.21

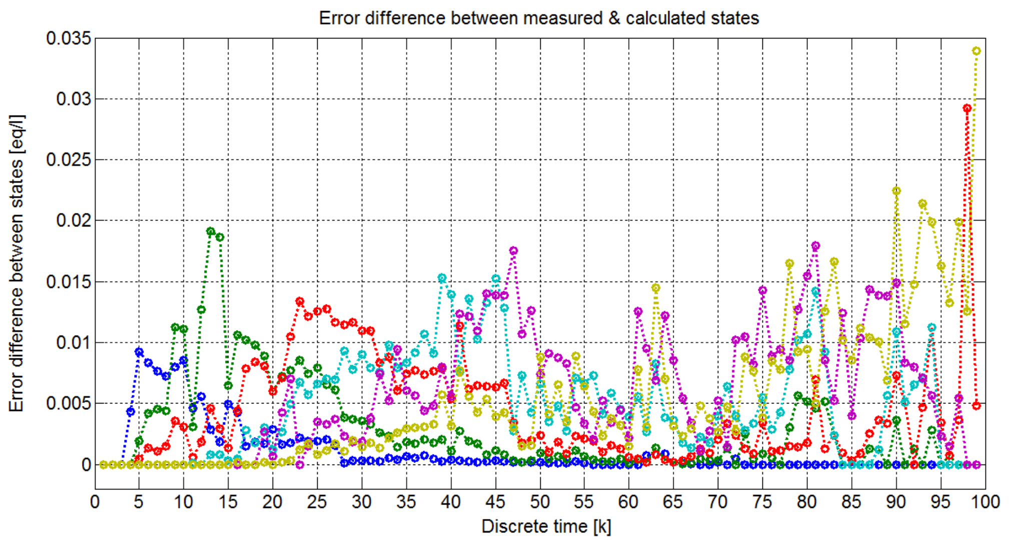


Figure 6.71. The error signal in detail for the best results – Run 5 of Table 6.21

Table 6.22: Results of the first set of experiments of changing the initial parameter values

Estimation criterion the least squares error and the estimation run time for the first set of experiments		
Case 1: (First set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	2.1301e+006	0.0150
Run 2	520.0396	0.0240
Run 3	2.0314	0.0200
Run 4	0.1270	0.0200
Run 5	0.0236	0.0200

Results and discussion of the first set of experiments of the the direct method

Estimated states start tracking the measured values in an excellent fit at $\Delta t = 1/_{64}\Delta t$, Figure 6.64. The best results are obtained at $\Delta t = 1/_{195}\Delta t$, and all the results are shown in Figures 6.64–6.71.

For the *direct method* it seems that the *process run* time is improved drastically and the *estimation square error* far outweighs all the other previous cases, including those of the *measure output based* and the *least squares* methods.

The second set of experiments of the direct method

The second set of experiments considers changing of the *control input* using the best minimum sampling period, $\Delta t = 1/_{195}\Delta t$ and the results are given in Figures 6.72–6.75. The control input is regulated by the process upflow period T, Table 6.23. The aim of this set of experiments is to observe the influence of the control input on the overall performance of the process. The criterion for evaluating the results is the least squares error and the process run time.

Table 6.23: **The second set of experiments of the changing control input at best minimum sampling period** $\Delta t = \frac{1}{32}\Delta t$

Changing control input at best sampling period						
Process parameter (units)	Values	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 32.93\ l$	✓	✓	✓	✓	✓	✓
$d = 2/3$ (liquid/res-ratio)	✓	✓	✓	✓	✓	✓
T (lupflow period in min)	$T = 17\,\mathrm{min}$	$T = 0.1\times T$	$T = 0.2\times T$	$T = 0.5\times T$	$T = T$	$T = 2.0\times T$
$F_R = (d\times h)/T$	$F_R = 1.2914$	$F_R = 12.9137$	$F_R = 6.4569$	$F_R = 2.5827$	$F_R = 1.2914$	$F_R = 0.6457$
$x_f = 0.435$	✓	✓	✓	✓	✓	✓
dt, sampling period (Δt)	0.012 min		✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in th table

The table shows the changing the control input by changing the upflow period. The "Values" heading in the table refers to reference values used to compare the changing values. The upflow period T is inversely proportional to the control input F_R.

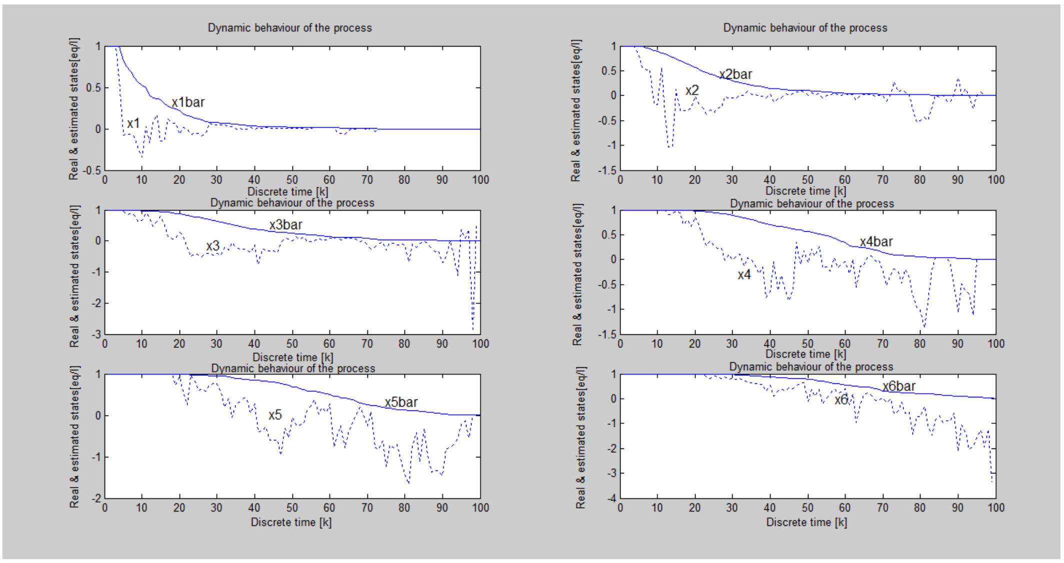


Figure 6.72. Model dynamic behaviour showing the overall real and estimated states of the worst case results – Run 1 Table 6.23

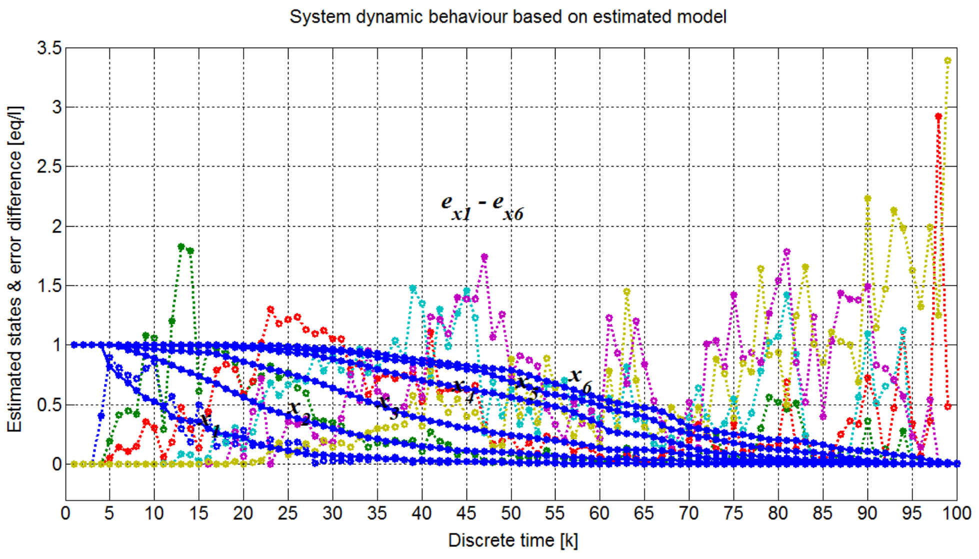


Figure 6.73. Model dynamic behaviour showing the error difference in relation to estimated states for the worst case results – Run 1 Table 6.23

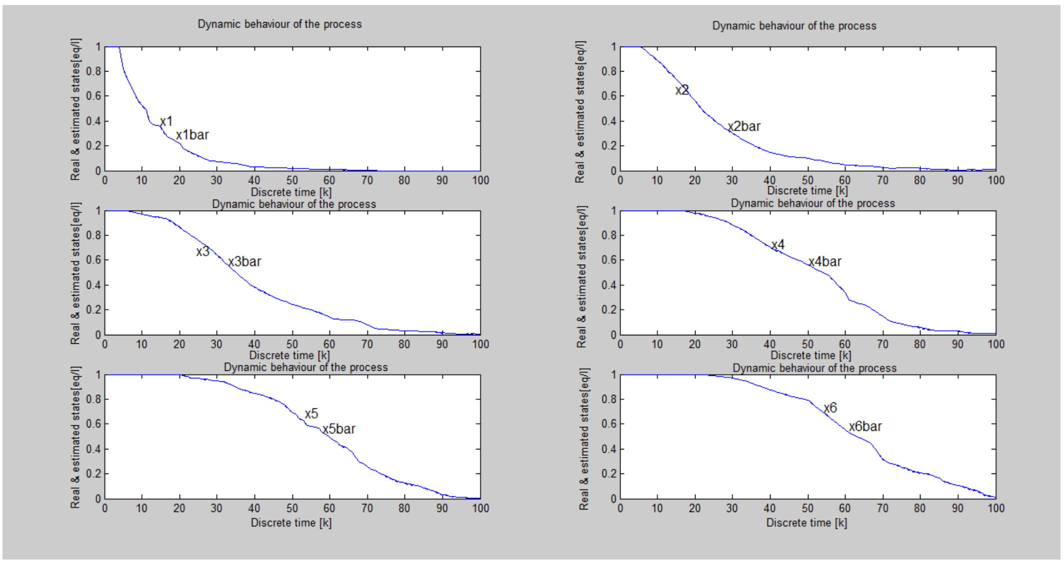


Figure 6.74. Model dynamic behaviour showing the overall real and estimated states for the best results – Run 5 Table 6.23

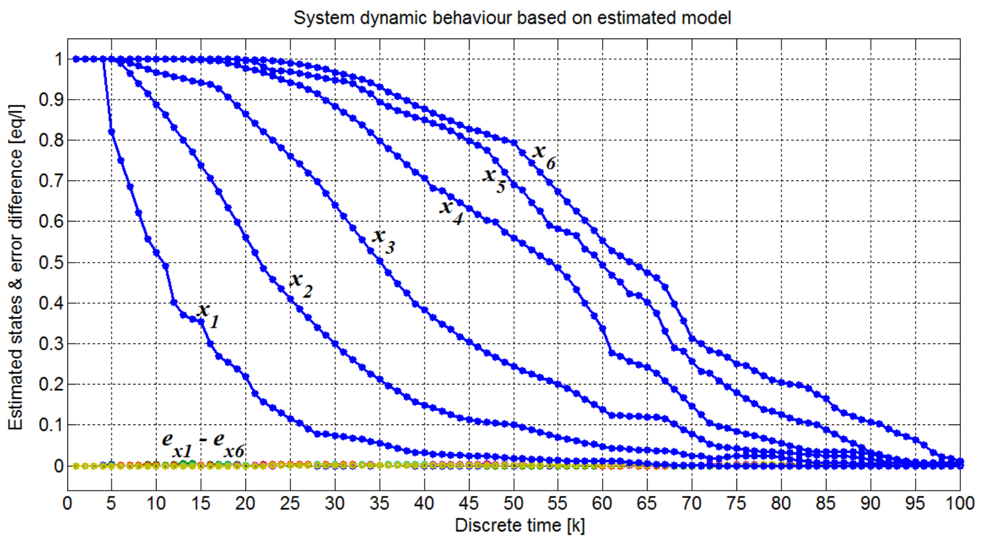


Figure 6.75. Model dynamic behaviour showing the error difference in relation to estimated states for the best results – Run 5 Table 6.23

Table 6.24: **Results of second set of experiments of the changing control input at best minimum sampling period**

Estimation criterion the least squares error and the estimation run time for the second set of experiments		
Case 2: (Second set of experiments)	$J(p)$	Run time (t_{calc})[sec]
Run 1	230.7372	0.1550
Run 2	14.4303	0.0250
Run 3	0.3711	0.0200
Run 4	0.0236	0.0180
Run 5	0.0016	0.0150

Results and discussion for the second set of experiments of the direct method

The best estimation results are obtained at smaller values of the control input at $F_R = 0.0349$, this corresponds to the highest value of the process upflow period (in this case, $T = 10T$) Table 6.24. The direct method once again produced the best results in terms of the *least squares error* and the *smallest run time* in comparison to all other cases tested so far.

The third set of experiments of the direct method

The last set of experiments considers the changing disturbance at constant control input, Table 6.25. The aim of this set of experiments is to consider the effect of the disturbance on the overall performance of the estimation of the *direct method*. The disturbance value will be varied in the order of 0.55 from 0.055 to 1.0. The results are presented in Figures 6.76–6.79.

Table 6.25: **The third set of experiments of the changing disturbance input at constant control input**

Changing disturbance input at constant control input and best minimum sampling period						
Process parameter (units)	Values/(Reference)	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 32.93\ l$	✓	✓	✓	✓	✓	✓
$d = 2/3$ (liquid/resin ratio)	✓	✓	✓	✓	✓	✓
T (upflow period (min)	$T = 17\min$	✓	✓	✓	✓	✓
$F_R = (d \times h)/T$	$F_R = 1.2914$	✓	✓	✓	✓	✓
x_f disturbance input	$x_f = 0.435$	$x_f = 0.05$	$x_f = 0.110$	$x_f = 0.550$	$x_f = 0.754$	$x_f = 1.00$
dt, sampling period (Δt)	0.012 min	✓	✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

In this set of experiments the best sampling period and constant control input are used. The control input used is based on the experimental liquid upflow period of $T = 17\min$. The investigation is to observe the influence of the disturbance signal on the process and to identify the worse and optimum cases of process dynamics based on the disturbance values.

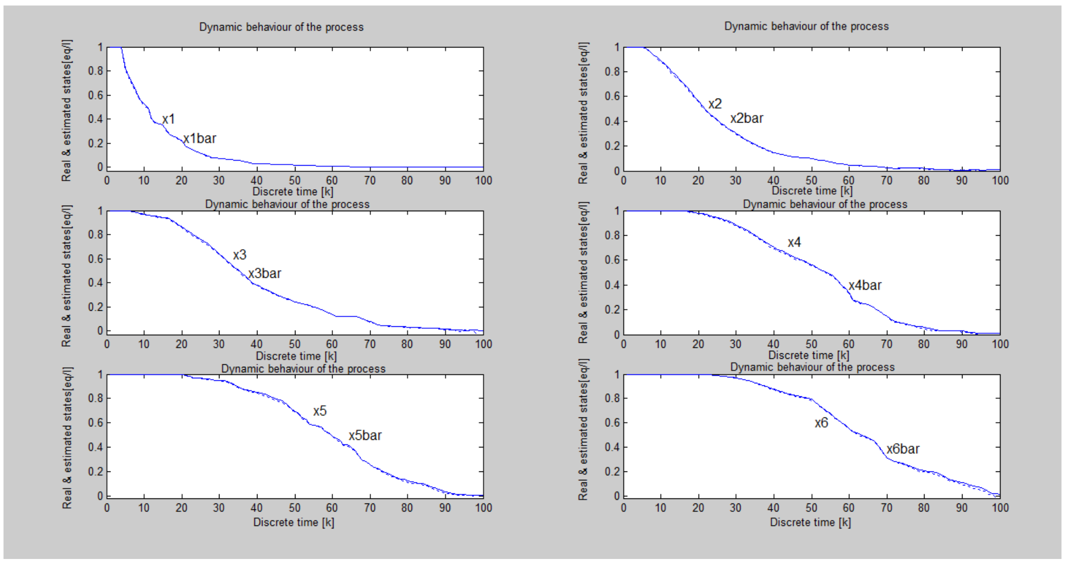


Figure 6.76. Model dynamic behaviour showing the overall real and estimated states of the worst case results – Run 1 Table 6.25

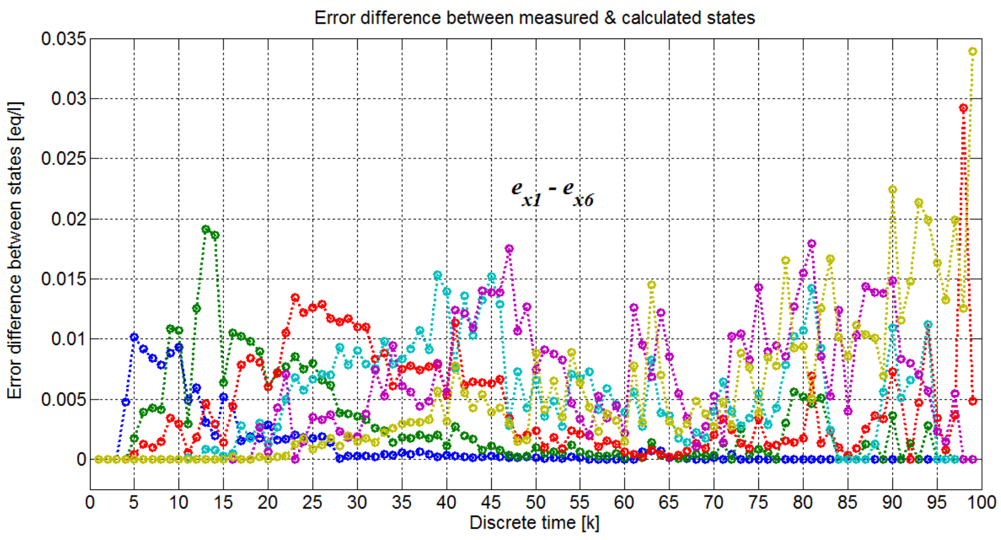


Figure 6.77. The error difference in detail for the worst case results – Run 1 of Table 6.25

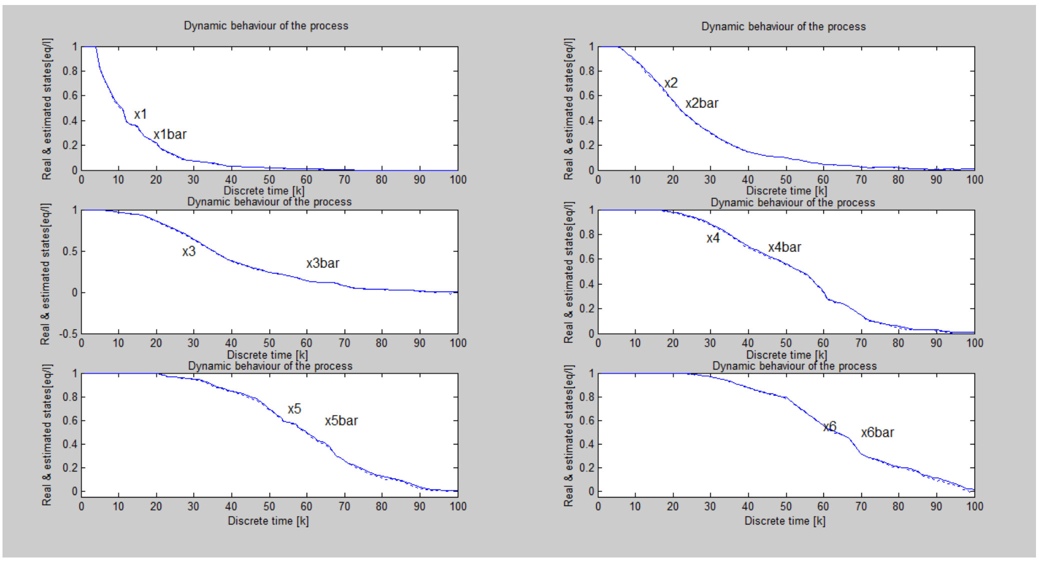


Figure 6.78. Model dynamic behaviour showing the overall real and estimated states for the best results – Run 5 Table 6.25

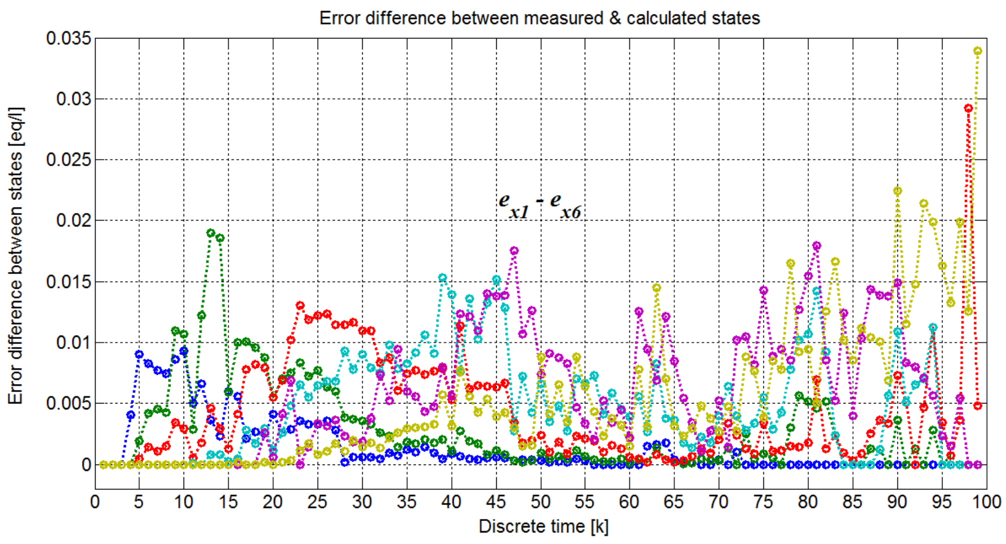


Figure 6.79. The error difference in detail for the best results – Run 5 of Table 6.25

Table 6.26: Results of third set of experiments of the changing disturbance input at constant control input

Estimation criterion the least squares error and the estimation run time for the third set of experiments		
Case 3: (Third set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	0.0236	0.0200
Run 2	0.0236	0.0150
Run 3	0.0236	0.0150
Run 4	0.0235	0.0150
Run 5	0.0235	0.0150

Results and discussion of the third set of experiments of the direct method

The best results are obtained from above the midpoint of the disturbance value $x_f = 0.754$; these are only best in terms of the *least squares error*, Table 6.26 The method seems to handle the disturbances fairly well, from the lowest to the highest. The criteria show that there is a very slight or almost no difference in the *least squares error* and the *run time*. The differences between the runs can be considered immaterial.

6.7. Linear model parameter estimation using the state vector measurements (Lagrange based method)

The Lagrange technique is one of many optimization methods used to minimize or maximize a function depending on the optimization requirement. The method uses gradient techniques of optimization if the problem is nonlinear. The basic requirement for optimization is the test for optimality conditions and then applying necessary conditions to minimize or maximize the function's variables of interest including the Lagrange multiplier (Polak, 1997:1).

Generally, there are two optimality conditions that must be met, the *necessary* and *sufficient* conditions:

1) the *necessary conditions* that must be met by the solution in all cases, and
2) the *sufficient conditions* that stipulate that if the necessary conditions are satisfied, an *extrema* (minima/maxima) is guaranteed (Polak, 1997:1; Pun, 1969:25).

The Lagrangian method if applied to solve the problem for parameter estimation formulated as follows:

Find the vector of the parameters $p(k)$, $k = \overline{1, K}$ that minimizes the error between the measured process states and the expected model states under the model structure (constraints) of the considered process.

$$J(p_x) = \sum_{i=k}^{K} \left\| (\hat{x}(k) - x(k)) \right\|_Q^2 \tag{6.23}$$

$$x(k+1) = x(k) + \Delta t \left[Ax(k) + B_1 x(k) F_R(k) + BF_R(k) + Wx_f(k) \right] \tag{6.24}$$

The Lagrangian function is formed by the performance index and the model equation using the Lagrangian variables $\lambda(k)$.

$$L_a = J(p_x) + \lambda^T(k) \left[-x(k+1) + \Delta t \left[Ax(k) + Bx(k) F_R(k) + B_1 F_R(k) + Wx_f(k) \right] \right] \tag{6.25}$$

where $\lambda(k) \in R^N$ – is the vector of the Lagrangian variables.

The values of the Lagrangian variables λ are unknown and need to be determined from the necessary conditions: $\frac{\partial L_a}{\partial p(k)} = 0$ and $\frac{\partial L_a}{\partial \lambda(k)} = 0$,

where,

$$p(k) = \left[p_1(k), p_2(k), p_3(k), p_4(k), ..., p_m(k), ... p_q(k)\right] \quad (6.26)$$

and the vector of Lagrange is

$$\lambda(k) = [\lambda_1(k), \lambda_2(k), \lambda_3(k), ..., \lambda_N(k)]^T \quad (6.27).$$

The resulting vectors for the optimality condition $\frac{\partial L_a}{\partial p(k)} = 0$ are

$$\begin{bmatrix} \frac{\partial L_{a1,1}}{\partial p_1(k)} \\ \frac{\partial L_{a2,1}}{\partial p_2(k)} \\ \frac{\partial L_{a3,1}}{\partial p_3(k)} \\ \frac{\partial L_{a4,1}}{\partial p_4(k)} \end{bmatrix} = \begin{bmatrix} \Delta t\lambda_1(k)[x_0(k) - x_1(k)] \\ \Delta t\lambda_1(k)[x_2(k)F_R(k)] \\ -\Delta t\lambda_1(k)[x_1(k)F_R(k)] \\ \Delta t\lambda_1(k)F_R(k) \end{bmatrix} = 0 \quad (6.28)$$

$$\begin{bmatrix} \frac{\partial L_{a5,2}}{\partial p_5(k)} \\ \frac{\partial L_{a6,2}}{\partial p_6(k)} \\ \frac{\partial L_{a7,2}}{\partial p_7(k)} \\ \frac{\partial L_{a8,2}}{\partial p_8(k)} \end{bmatrix} = \begin{bmatrix} \Delta t\lambda_2(k)[x_1(k) - x_2(k)] \\ \Delta t\lambda_2(k)[x_3(k)F_R(k)] \\ -\Delta t\lambda_2(k)[x_2(k)F_R(k)] \\ \Delta t\lambda_2(k)F_R(k) \end{bmatrix} = 0 \quad (6.29)$$

$$\begin{bmatrix} \frac{\partial L_{a9,3}}{\partial p_9(k)} \\ \frac{\partial L_{a10,3}}{\partial p_{10}(k)} \\ \frac{\partial L_{a11,3}}{\partial p_{11}(k)} \\ \frac{\partial L_{a12,3}}{\partial p_{12}(k)} \end{bmatrix} = \begin{bmatrix} \Delta t\lambda_3(k)[x_2(k) - x_3(k)] \\ \Delta t\lambda_3(k)[x_4(k)F_R(k)] \\ -\Delta t\lambda_{3(}k)[x_3(k)F_R(k)] \\ \Delta t\lambda_3(k)F_R(k) \end{bmatrix} = 0 \quad (6.30)$$

$$\begin{bmatrix} \dfrac{\partial L_{a13,4}}{\partial p_{13}(k)} \\ \dfrac{\partial L_{a14,4}}{\partial p_{14}(k)} \\ \dfrac{\partial L_{a15,4}}{\partial p_{15}(k)} \\ \dfrac{\partial L_{a16,5}}{\partial p_{16}(k)} \end{bmatrix} = \begin{bmatrix} \Delta t\lambda_4(k)\left[x_3(k) - x_4(k)\right] \\ \Delta t\lambda_4(k)\left[x_5(k)F_R(k)\right] \\ -\Delta t\lambda_4(k)\left[x_4(k)F_R(k)\right] \\ \Delta t\lambda_4(k)F_R(k) \end{bmatrix} = 0 \tag{6.31}$$

$$\begin{bmatrix} \dfrac{\partial L_{a17,5}}{\partial p_{17}(k)} \\ \dfrac{\partial L_{a18,5}}{\partial p_{18}(k)} \\ \dfrac{\partial L_{a19,5}}{\partial p_{19}(k)} \\ \dfrac{\partial L_{a20,5}}{\partial p_{20}(k)} \end{bmatrix} = \begin{bmatrix} \Delta t\lambda_5(k)\left[x_4(k) - x_5(k)\right] \\ \Delta t\lambda_5(k)\left[x_6(k)F_R(k)\right] \\ -\Delta t\lambda_5(k)\left[x_5(k)F_R(k)\right] \\ \Delta t\lambda_5(k)F_R(k) \end{bmatrix} = 0 \tag{6.32}$$

$$\begin{bmatrix} \dfrac{\partial L_{a21,6}}{\partial p_{21}(k)} \\ \dfrac{\partial L_{a22,6}}{\partial p_{22}(k)} \\ \dfrac{\partial L_{a23,6}}{\partial p_{23}(k)} \end{bmatrix} = \begin{bmatrix} \Delta t\lambda_6(k)\left[x_5(k) - x_6(k)\right] \\ -\Delta t\lambda_6(k)\left[x_6(k)F_R(k)\right] \\ \Delta t\lambda_6(k)F_R(k) \end{bmatrix} = 0 \tag{6.33}$$

The notations $\dfrac{\partial L_{ap,n}}{\partial p(k)}$ are derivatives of the Lagrange function, Equation (6.25) with respect to each parameter, $p = \overline{p_1, p_q}$ for all stages, $n = \overline{1, N}$. The second part of the necessary conditions expression is:

$$\frac{\partial L_a}{\partial \lambda(k)} = \left[-x(k+1) + \Delta t\left[Ax(k) + B_1 x(k)F_R(k) + BF_R(k) + Wx_f(k)\right]\right] = 0 \tag{6.34}$$

where $\lambda \in R^N$, $N = 6$

The derivative of the Lagrange function L_a with respect to the multipliers represents the gradient of the model according to the multipliers λ, and the gradient function is not a function of the Lagrange variables, Equation (6.34). The necessary conditions, Equations (6.28)–(6.33) and Equation (6.34) are solved using the gradient methods since these equations cannot be solved analytically. The *unknown parameters* and *Lagrangian multipliers* must be determined. It is necessary to guess the initial values of the $p(k)$ and $\lambda(k)$, i.e., p^0 and λ^0 before proceeding with the procedure. The vectors $x(k)$ and $F_R(k)$ are measured or obtained from the existing data.

The improved values of the vectors $p(k)$ and $\lambda(k)$ are then calculated using the gradient step procedure where for each and every stage

$$p^{new}(k) = p^{old}(k) - \alpha_p \frac{\partial L_a}{\partial p(k)} \tag{6.35}$$

$$\lambda^{new}(k) = \lambda^{old}(k) + \alpha_\lambda \frac{\partial L_a}{\partial \lambda(k)} \tag{6.36}$$

where α_p – is the step of the gradient for parameters and

α_λ – is the step of the gradient for the Lagrange multipliers.

The iterations for variables improvement are repeated until gradient search reaches some small number, i.e.,

$$\frac{\partial L_a}{\partial p(k)} < \varepsilon_p \tag{6.37}$$

$$\frac{\partial L_a}{\partial \lambda(k)} < \varepsilon_\lambda \qquad (6.38),$$

where ε_p and ε_λ are very small numbers for stopping the procedure for unknown parameters and unknown Lagrangian multipliers respectively.

or a maximum number of operations M is reached. Both the gradient conditions must be met to stop the gradient iterations, and this translates to optimal values of the parameters and multipliers being reached.

6.7.1. Procedure for linear model Lagrange based parameter estimation

In order to determine the unknown parameters using Lagrange's method, the following procedure is used (Figure 6.80):

- initialization of the unknown parameters and the Lagrange's multipliers.
- The procedure starts at the given initial Lagrange multipliers, λ_0. Then the creation of a Lagrange function L_a.
- The function is composed of the *function to be optimized* and *constraints if included*, Equation (6.25).
- Find the derivatives of the Lagrange function with respect to each variable involved (partial derivatives), including the Lagrangian multipliers λ, Equations (6.28)–(6.33) and Equation (6.34).

- Each derivative are set to zero Equation (6.28)–(6.34), for the procedure to produce optimal variables.
- An iteration condition is then declared such that calculations are repeated until optimal values are obtained. At the beginning of the iteration, the unknown variables/parameters are initial guesses.
- The solution procedure is gradients based; after the first iteration, use the previous values to improve the unknown variable/parameter in the next calculation, Equations (6.35) and (6.36). The iterations are repeated until some stopping procedure is reached, Equations (6.37) and (6.38).

6.7.2. Algorithm and flowchart for the linear model parameter estimation using the state vector measurements (theLagrange method)

1. Initialize the process data values as in Section 6.4.2:
2. Initial parameters for the gradient procedure as in Section 6.4.2:
3. Formulate the initial parameters vector, p_0 from process parameters and initialize state vector x_0 and the disturbance x_f.
4. Initialize the gradient procedure parameters, $\lambda(k)$, α_p, α_λ, ε_p and ε_λ.
5. Initialize parameters for the full process trajectory, $p_0 \in R^{23\times K}$, $x_0 \in R^{6\times K}$, $x_f(k) \in^{1\times K}$, $\lambda_0 \in R^{6\times K}$.
6. While $j \le M$:

   6.1. Using all the necessary model parameters Δt, A, B, and B_1, and the model variables, the state $x(k)$, the control output $F_R(k)$, and the disturbance $x_f(k)$, perform the following calculations:

   - Calculate the gradients for the unknown parameters, Equations (6.30)–(6.35),
   - Formulate the gradient vector from calculated gradients

   $$\frac{\partial L_a}{\partial p(k)} = e_p(k) \tag{6.39}$$

   6.2. Check for optimality (stopping procedure)

   » If the gradient is less than the preselected very small number $e_p(k) < \varepsilon_p$, then,

   $$p^{(i+1)}(k) = p^{(i)}(k) \tag{6.40}$$

   » or else, calculate the improved parameters,

$$p^{(i+1)}(k) = p^{(i)}(k) - \alpha_p e_p(k) \tag{6.41}$$

6.3. Calculate the gradient of the Lagrange multipliers, Equation (6.44)

$$\frac{\partial L_a}{\partial \lambda(k)} = e_\lambda(k) \tag{6.42}$$

6.4. Check for optimality (stopping procedure):

» If the gradient is less than the preselected very small number, $e_\lambda(k) < \varepsilon_\lambda$, then

$$\lambda^{(i+1)}(k) = \lambda^{(i)}(k) \tag{6.43}$$

» or else, calculate the improved Lagrange multipliers

$$\lambda^{(i+1)}(k) = \lambda^{(i)}(k) + \alpha_\lambda e_\lambda(k) \tag{6.44}$$

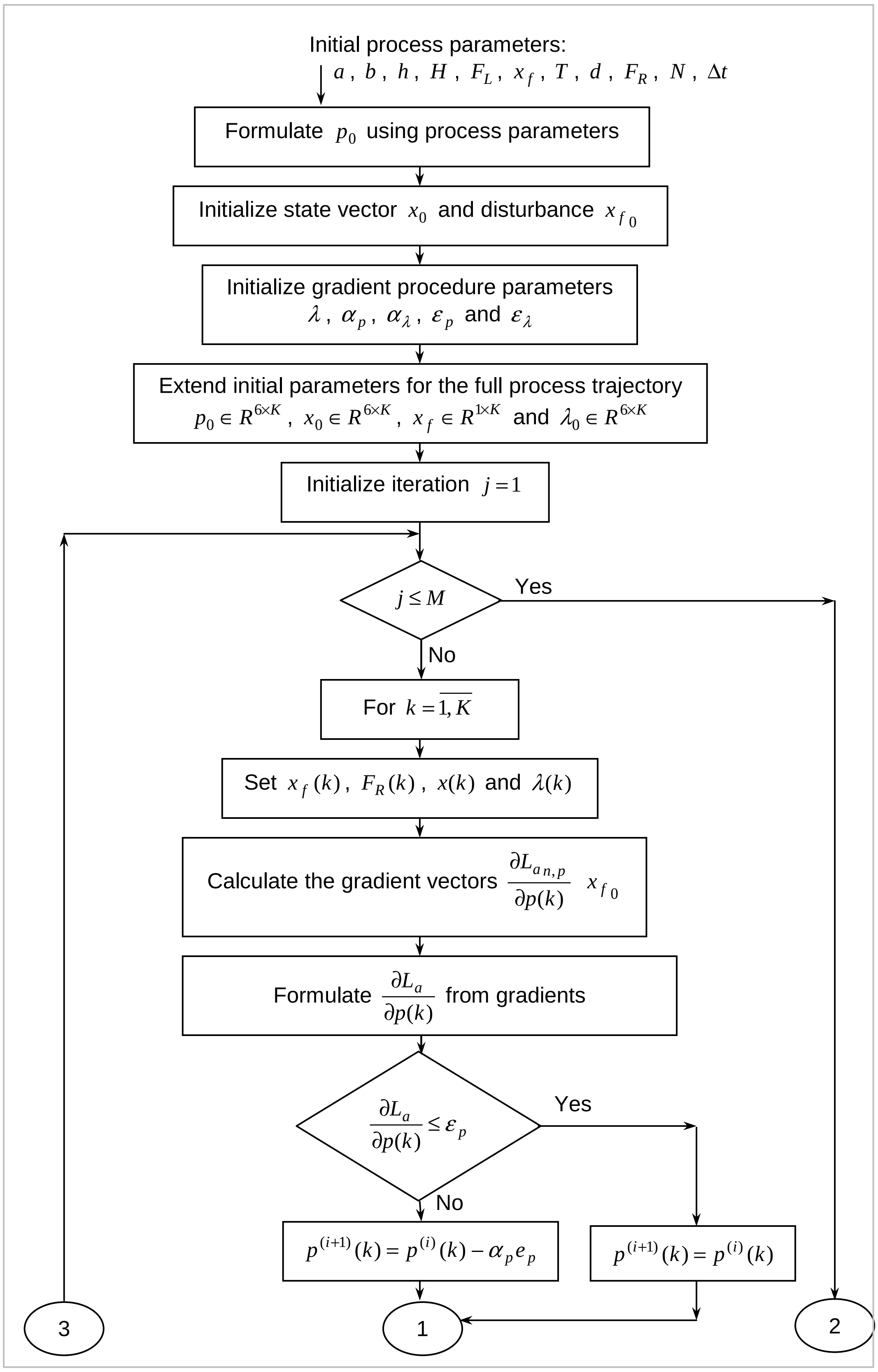
6.5. Increment iteration index $j = j+1$ and repeat from step 5.

7. For $k = \overline{1,K}$ calculate the estimated states using obtained parameters.
8. Calculate the error difference between measured and calculated states.
9. Plot the both states trajectories and the error condition.

6.7.3. Results from the experiments with the linear model Lagrange based parameter estimation procedure using state vector measurements

Experiments were conducted to validate the performance of the Lagrange method for parameter estimation. The following process parameters, *sampling period* Δt, *Lagrangian* and *parameters gradient steps*, α_λ and α_p, and the *initial model parameters*' vector p are changed in values to observe their influence on the overall process dynamics. Each parameter's influence is considered in a different set of experiments and the best results are applied in the next set. The criterion for evaluation of results is once again is based on the value of the *least squares error* $J(p) = \sum_{k=1}^{K} \|e(k)\|^2$ and the algorithm's *run time*.

In all experiments' results $x_n bar = x_n$, and $x_n^{\sim} = \hat{x}_n(k)$ for $n = \overline{1,N}$, Equation (6.23)


Initial process parameters:
a, b, h, H, F_L, x_f, T, d, F_R, N, Δt
Formulate p_0 using process parameters
Initialize state vector x_0 and disturbance x_{f0}
Initialize gradient procedure parameters λ, α_p, α_λ, ε_p and ε_λ
Extend initial parameters for the full process trajectory $p_0 \in R^{6\times K}$, $x_0 \in R^{6\times K}$, $x_f \in R^{1\times K}$ and $\lambda_0 \in R^{6\times K}$
Initialize iteration $j=1$
$j \le M$
Yes
No
For $k=\overline{1,K}$
Set $x_f(k)$, $F_R(k)$, $x(k)$ and $\lambda(k)$
Calculate the gradient vectors $\frac{\partial L_{a\,n,p}}{\partial p(k)}$ x_{f0}
Formulate $\frac{\partial L_a}{\partial p(k)}$ from gradients
$\frac{\partial L_a}{\partial p(k)} \le \varepsilon_p$
Yes
No
$p^{(i+1)}(k) = p^{(i)}(k) - \alpha_p e_p$
$p^{(i+1)}(k) = p^{(i)}(k)$
3
1
2

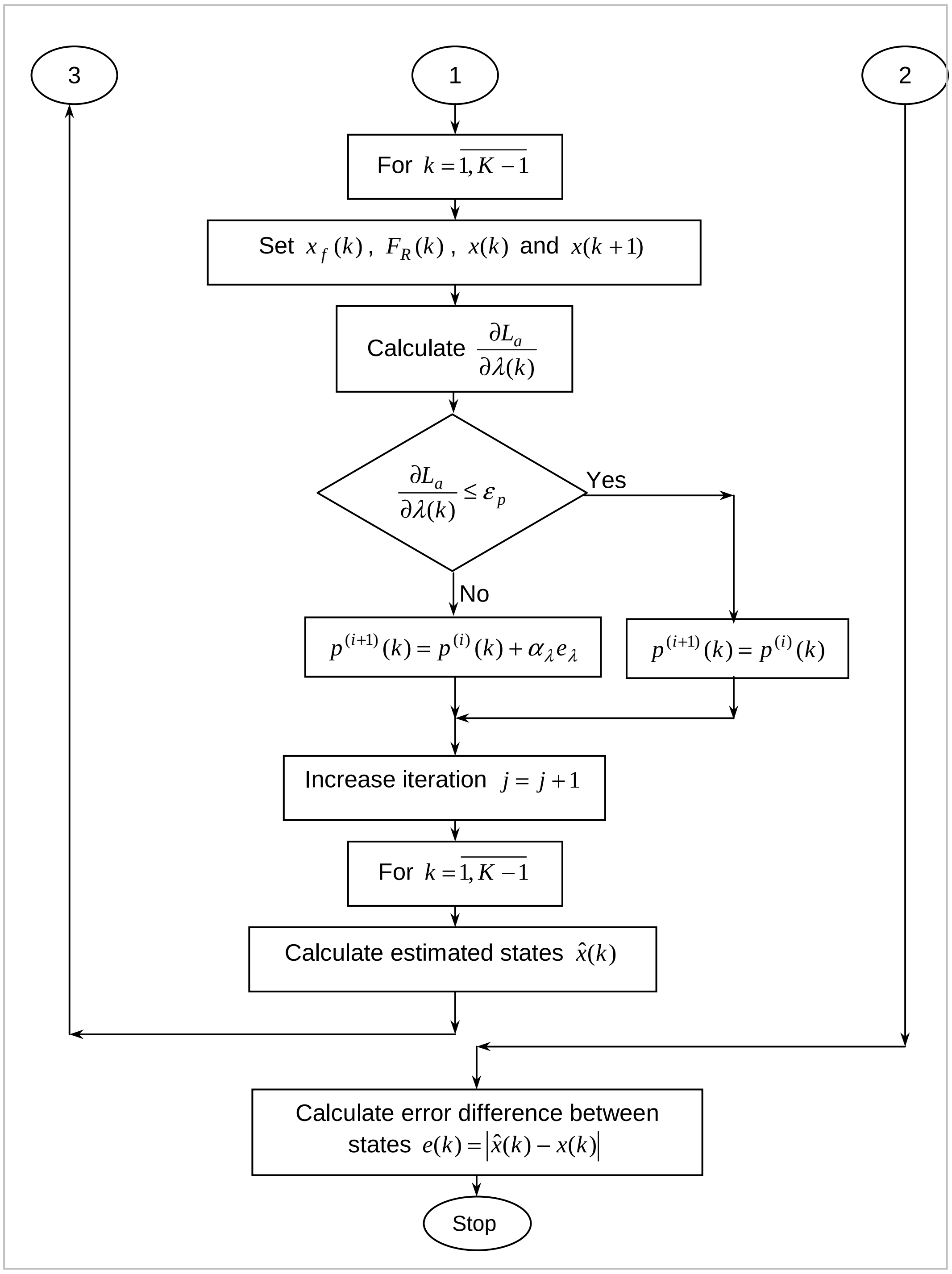

3
1
2
For $k = \overline{1, K-1}$
Set $x_f(k)$, $F_R(k)$, $x(k)$ and $x(k+1)$
Calculate $\frac{\partial L_a}{\partial \lambda(k)}$
$\frac{\partial L_a}{\partial \lambda(k)} \le \varepsilon_p$
Yes
No
$p^{(i+1)}(k) = p^{(i)}(k) + \alpha_\lambda e_\lambda$
$p^{(i+1)}(k) = p^{(i)}(k)$
Increase iteration $j = j+1$
For $k = \overline{1, K-1}$
Calculate estimated states $\hat{x}(k)$
Calculate error difference between states $e(k) = |\hat{x}(k) - x(k)|$
Stop

Figure 6.80. Flowchart for the linear Lagrange–based model parameter estimation using the state vector measurements

The first set of experiments of the Lagrange based method

The first set of experiments considers varying the values of the *sampling period* at the constant *control input*, *model parameters* and *gradient steps*, Table 6.27. The results are presented in Figures 6.81–6.84. For this set of experiments to obtain improved estimated parameter values, holdup values were reduced.

Table 6.27: **The first set of experiments of the changing sampling period at fixed gradient steps**

Changing sampling period at fixed gradient steps						
Process parameter (units)	Values	Run 1	Run 2	Run 3	Run 4	Run 5
$h \;= 0.3293\; l$	✓	✓	✓	✓	✓	✓
$H \;= 0.42809\; l$	✓	✓	✓	✓	✓	✓
$F_L = 2000/60\;\; (l/\min)$	✓	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓	✓
$a = [0.8\;\; 0.7\;\; 0.5\;\; 0.4\;\; 0.3\;\; 0.02]$	$a = 230 \times a$	✓	✓	✓	✓	✓
$b = [0.025\;\; 0.1\;\; 0.2\;\; 0.3\;\; 0.4\;\; 0.6]$	$b = 60 \times b$	✓	✓	✓	✓	✓
M	2000	✓	✓	✓	✓	✓
dt , sampling period (Δt)	2.35 min	$\Delta t = \frac{1}{4}\Delta t$	$\Delta t = \frac{1}{8}\Delta t$	$\Delta t = \frac{1}{16}\Delta t$	$\Delta t = \frac{1}{64}\Delta t$	$\Delta t = \frac{1}{128}\Delta t$
Parameters gradient step (α_p)	0.3	✓	✓	✓	✓	✓
Lagrange multiplier gradient step (α_λ)	0.1	✓	✓	✓	✓	✓
ε_p , epsilon (very small number)	0.0001	✓	✓	✓	✓	✓
ε_λ , epsilon $\lambda(k)$ (very small number)	0.0001					

✓ means the value stays the same as in the previous mention in th table

The table shows the changing sampling period, with the reference value of $\Delta t = 2.35\,\text{min}$. This value is reduced in multiples of a quarter of the previous value for every run.

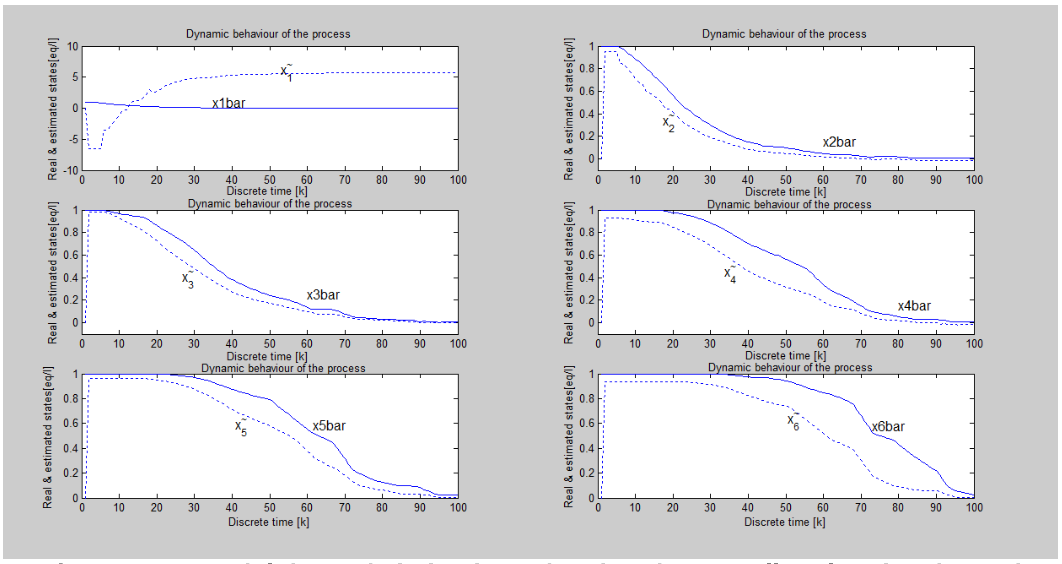


Figure 6.81. Model dynamic behaviour showing the overall real and estimated states for the worst case results – Run 1 Table 6.27

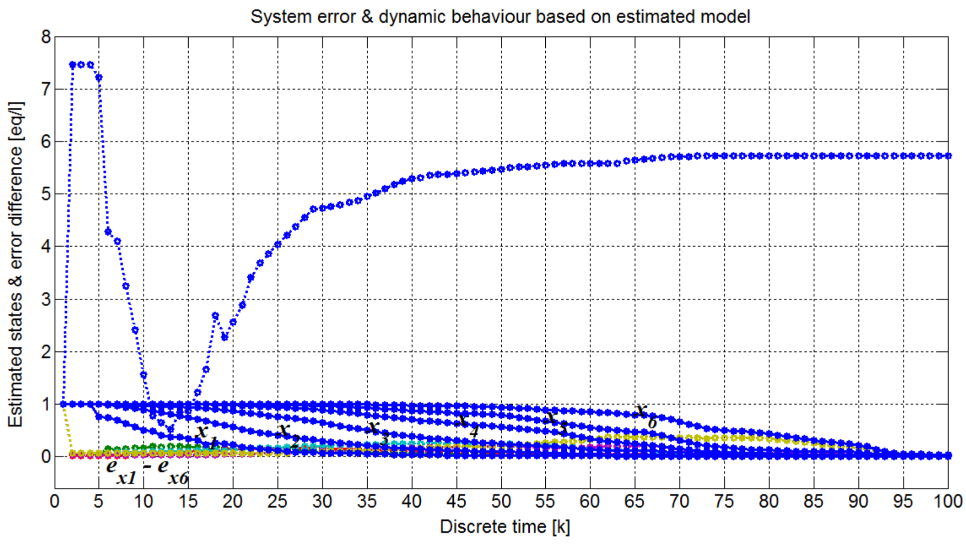


Figure 6.82. Model dynamic behaviour showing the error difference in relation to estimated states for the worst case results – Run 1 Table 6.27

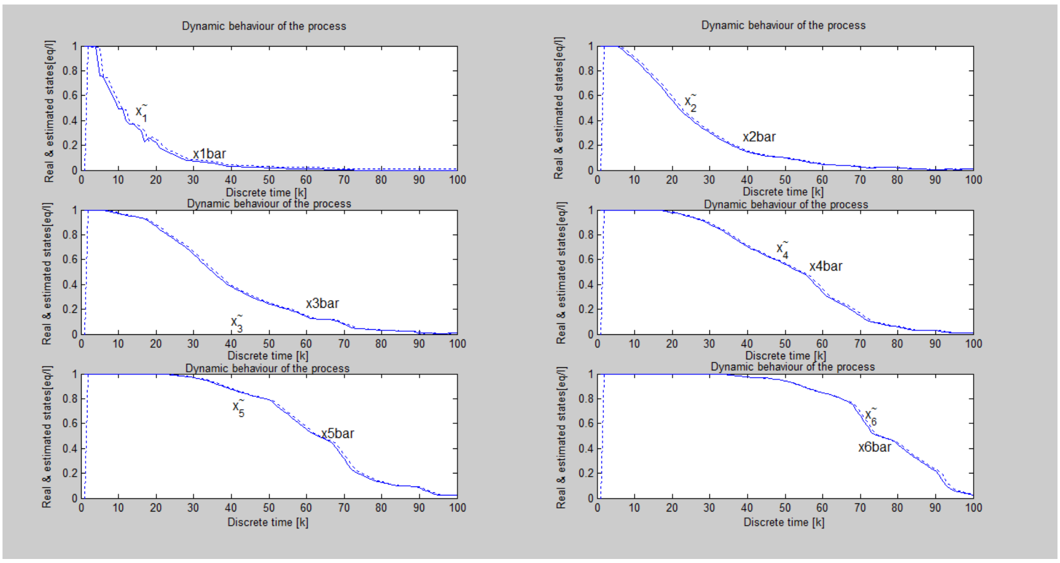


Figure 6.83. Model dynamic behaviour showing the overall real and estimated states for the best results – Run 5 Table 6.27

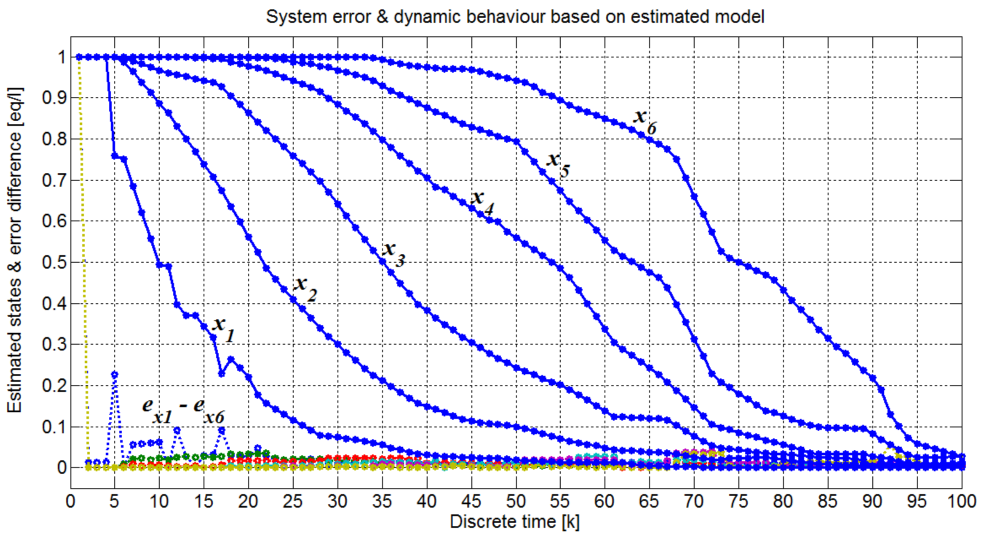


Figure 6.84. Model dynamic behaviour showing the error difference in relation to estimated states for the best case results – Run 5 Table 6.27

Table 6.28: Results of the first set of experiments of changing the disturbance input

Estimation criterion the least squares error and the estimation run time for the first set of experiments		
Case 1: (First set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	2.6357e+003	29.5310
Run 2	179.0961	29.4530
Run 3	17.9705	29.7490
Run 4	6.2011	30.0280
Run 5	6.1682	30.4820

Results and discussion of the first set of experiments for the Lagrange based method

From the first set of experiments that of changing *sampling period* Δt with fixed *gradient steps*, α_p and α_λ, Table 6.27, the estimated states start tracking the measured values convincingly at $\Delta t = \frac{1}{64}\Delta t$, Figure 6.81–6.84. At the lowest sampling period given, $\Delta t = \frac{1}{128}\Delta t$ the algorithm produces the best results, Table 6.28.

The second set of experiments of the Lagrange based method

The second set of experiments considers the varying *parameters gradient step* at constant *Lagrange multipliers gradient step* $\alpha_\lambda = 0.3$, using the best sampling period, Table 6.29. For practical reasons and better presentation, the sampling period has been set at $\Delta t = \frac{1}{195}\Delta t$. The results are presented by Figures 6.85–6.88. The evaluation criterion of the method is presented in Table 6.30.

Table 6.29: **The second set of experiments of the changing parameters gradient step values and constant Lagrange gradient step at best sampling period**

Changing parameters gradient step values and constant Lagrange gradient step						
Process parameter (units)	Values (Reference)	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 0.3293\ l$	✓	✓	✓	✓	✓	✓
$H = 0.42809\ l$	✓	✓	✓	✓	✓	✓
$F_L = 2000/60\ (l/\min)$	✓	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓	✓
$a = [0.8\ 0.7\ 0.5\ 0.4\ 0.3\ 0.02]$	$a = 230 \times a$	✓	✓	✓	✓	✓
$b = [0.025\ 0.1\ 0.2\ 0.3\ 0.4\ 0.6]$	$b = 60 \times b$	✓	✓	✓	✓	✓
M	10 000	✓	✓	✓	✓	✓
dt, sampling period (Δt)	0.012 min	✓	✓	✓	✓	✓
Parameters gradient step (α_p)	0.1	0.01	0.05	0.1	0.5	1.0
Lagrange multiplier gradient step (α_λ)	0.3	✓	✓	✓	✓	✓
ε_p, epsilon (very small number)	0.001	✓	✓	✓	✓	✓
ε_λ, epsilon (very small number)	0.001	✓	✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

The table show the changing values of the parameters gradient step from 0.1 to 1.0 nor respective order.

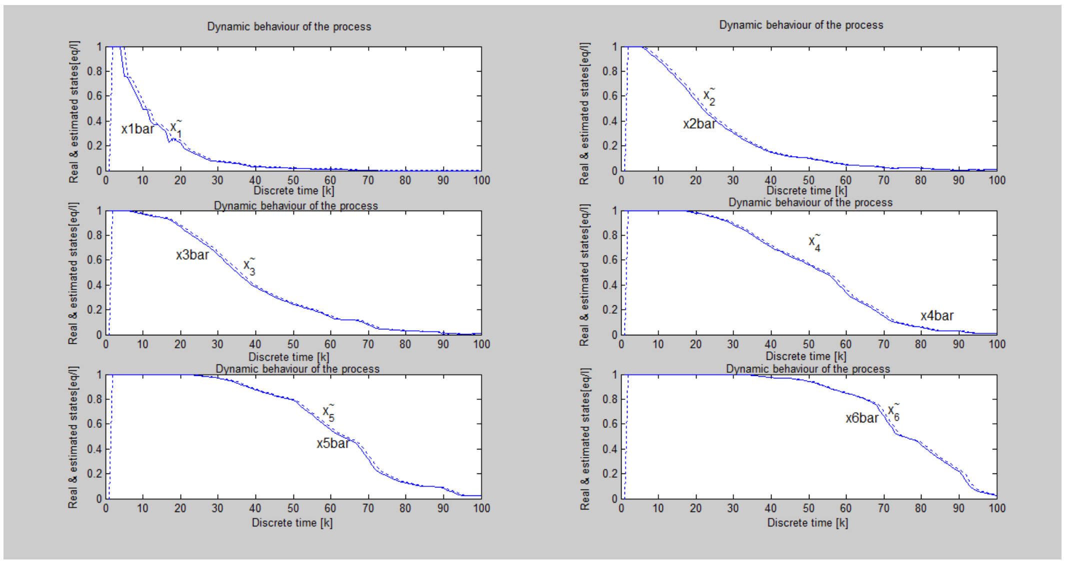


Figure 6.85. Model dynamic behaviour showing the overall real and estimated states for the best results – Run 4 Table 6.29

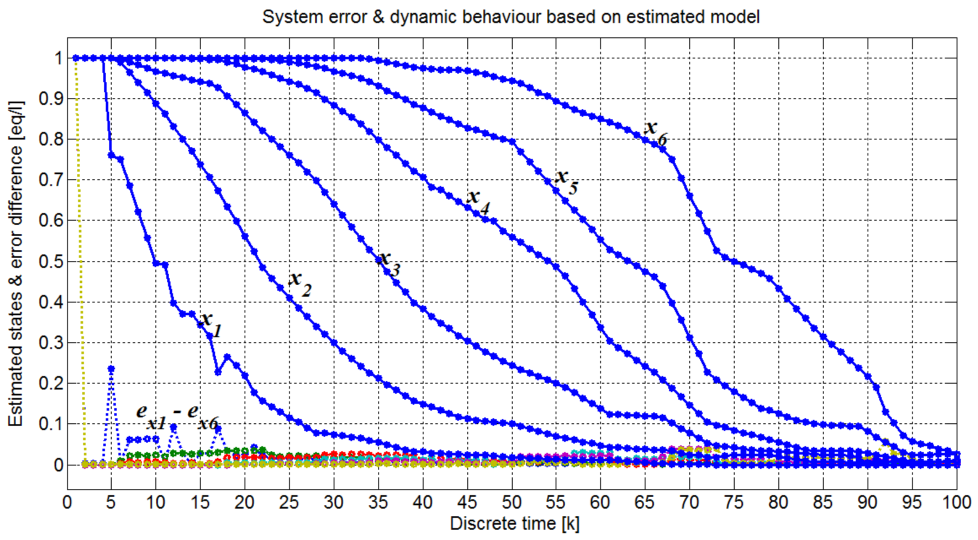


Figure 6.86. Model dynamic behaviour showing the error difference in relation to estimated states – the best results – Run 4 Table 6.29

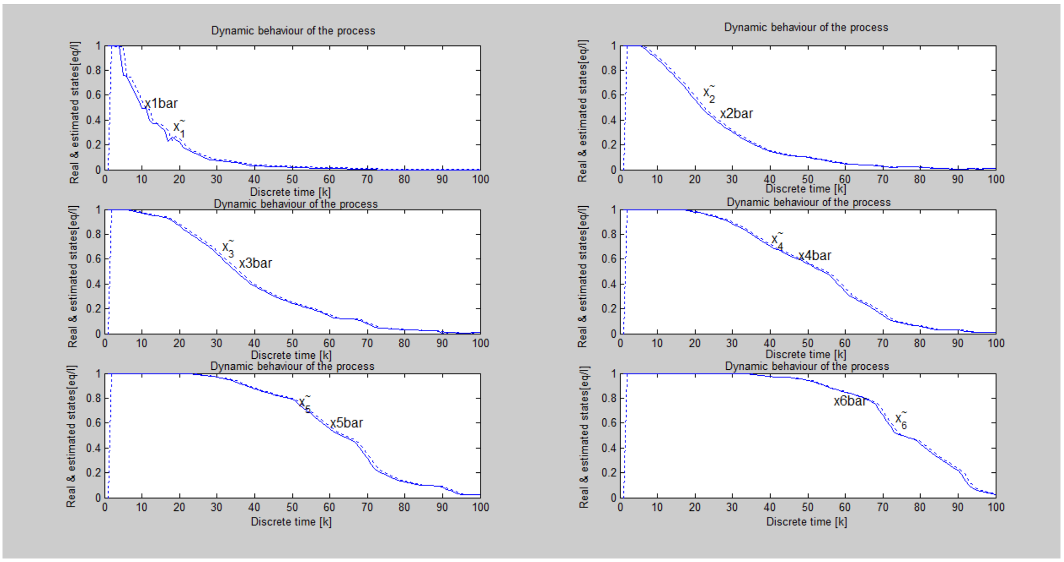


Figure 6.87. Model dynamic behaviour showing the overall real and estimated states for the worst case results – Run 1 Table 6.29

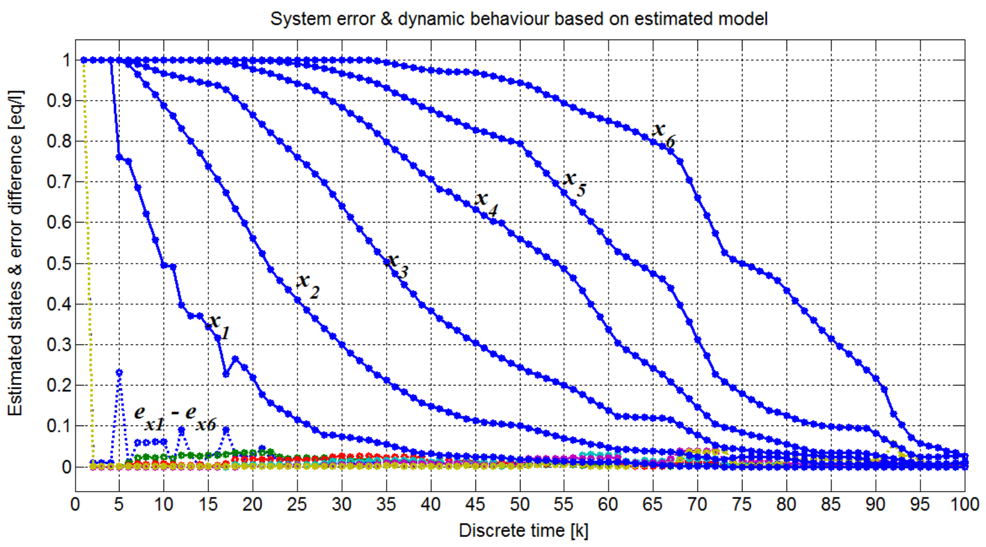


Figure 6.88. Model dynamic behaviour showing the error difference in relation to estimated states – the worst case results – Run 1 Table 6.29

Table 6.30: Results of the second set of experiments for the changing parameter gradient step and constant Lagrange gradient step at best sampling period

Estimation criterion the least squares error and the estimation run time for the second set of experiments		
Case 2: (Second set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	6.1806	31.9750
Run 2	6.1765	31.2930
Run 3	6.1765	30.9660
Run 4	6.1774	30.2480
Run 5	6.1806	31.7870

Results and discussion of second set of experiments for the Lagrange based method

The best results are obtained at the parameters gradient step of $\alpha_p = 0.5$. At the parameters gradient step of $\alpha_p = 1.0$, the algorithm starts behaving like those of lower values, Table 6.30. This set of experiments also demonstrates that for the given values, the estimation process was not influenced much by the given *parameter gradient step* values.

The third set of experiments of the Lagrange based method

The third set of experiments (Table 6.31) considers the changing *Lagrange multipliers gradient iteration step* at (best) constant *parameters gradient step*, $\alpha_p = 0.5$ from the previous experiment. With not so big difference in the criterion, the first and the last runs are simply presented as a reference, Figures 6.89–6.92.

Table 6.31: The third set of experiments for changing the Lagrange gradient step at the constant parameter gradient step and sampling period

Changing Lagrange gradient iteration step at constant parameter gradient iteration step						
Process parameter (units)	Values/(Reference)	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 0.3293\ l$	✓	✓	✓	✓	✓	✓
$H = 0.42809\ l$	✓	✓	✓	✓	✓	✓
$F_L = 2000/60\ (l/\min)$	✓	✓	✓	✓	✓	✓
$x_f = 0.435\ eq/l$	✓	✓	✓	✓	✓	✓
$a = [0.8\ 0.6\ 0.5\ 0.4\ 0.3\ 0.25]$	$a = 230 \times a$	✓	✓	✓	✓	✓
$b = [0.01\ 0.02\ 0.05\ 0.08\ 0.1\ 0.2]$	$b = 60 \times b$	✓	✓	✓	✓	✓
M	2 000	✓	✓	✓	✓	✓
dt, sampling period (Δt)	$0.012 \min$	✓	✓	✓	✓	✓
Parameters gradient step (α_p)	0.5	✓	✓	✓	✓	✓
Lagrange multiplier gradient step (α_λ)	0.01	0.001	0.01	0.1	0.5	1.0
ε_p, epsilon- p (very small number)	0.0001	✓	✓	✓	✓	✓
ε_λ, epsilon- $\lambda(k)$ (very small number)	0.0001	✓	✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

The table shows the changing values of the Lagrange gradient step while keeping all other values constant.

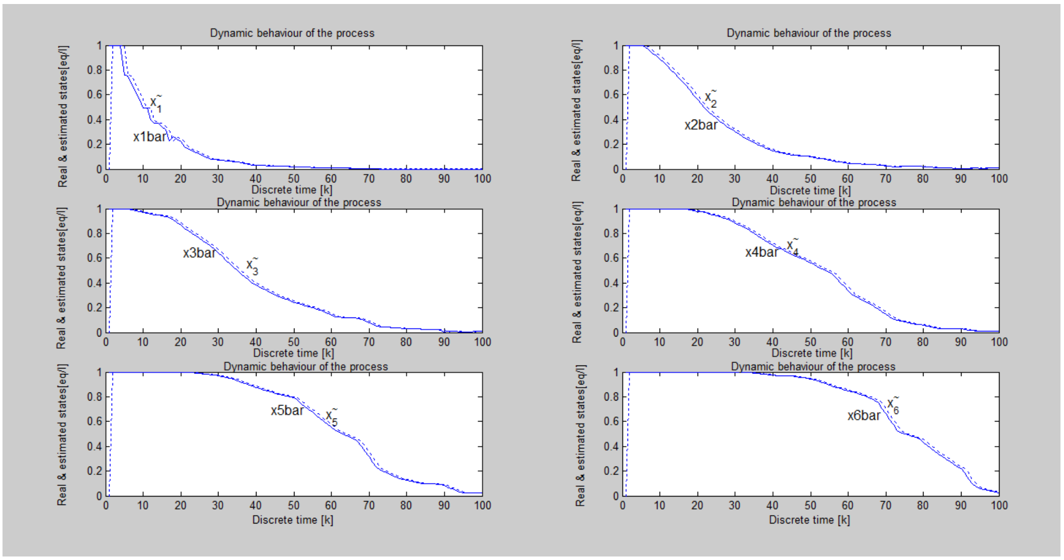


Figure 6.89. Model dynamic behaviour showing the overall real and estimated states for Run 1 of Table 6.31

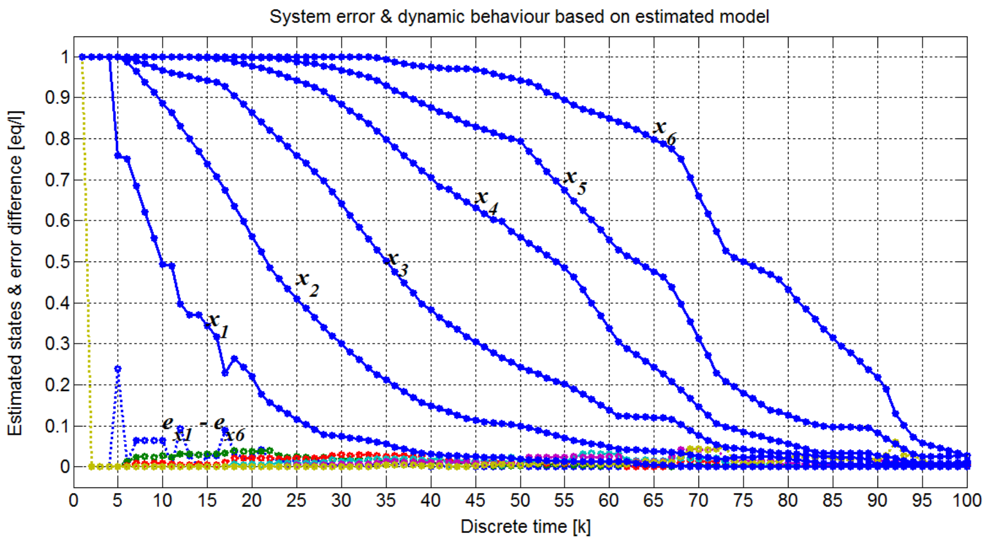


Figure 6.90. Model dynamic behaviour showing the error difference in relation to estimated states for Run 1 of Table 6.31

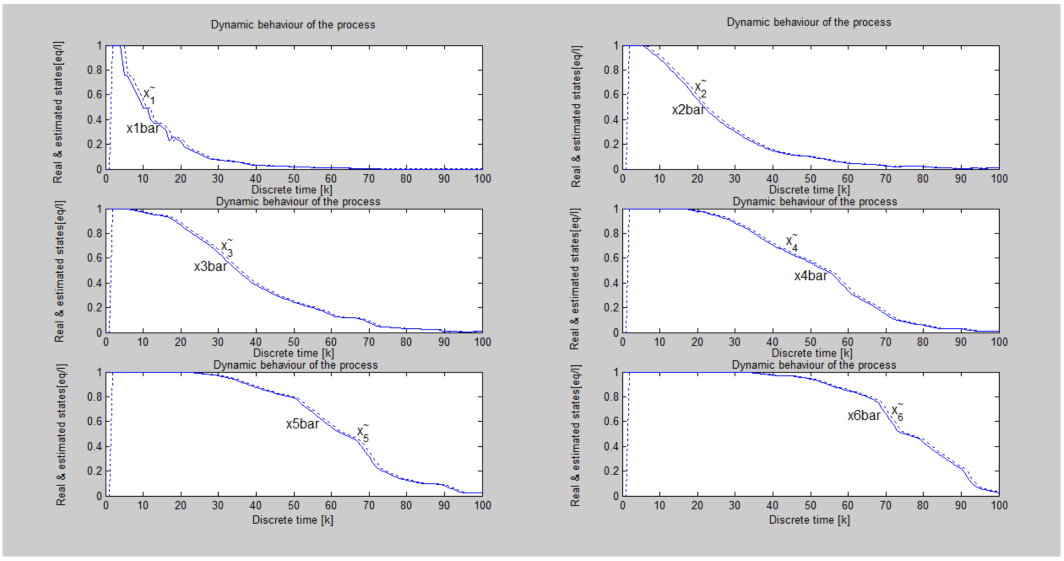


Figure 6.91. Model dynamic behaviour showing the overall real and estimated states for Run 5 of Table 6.31

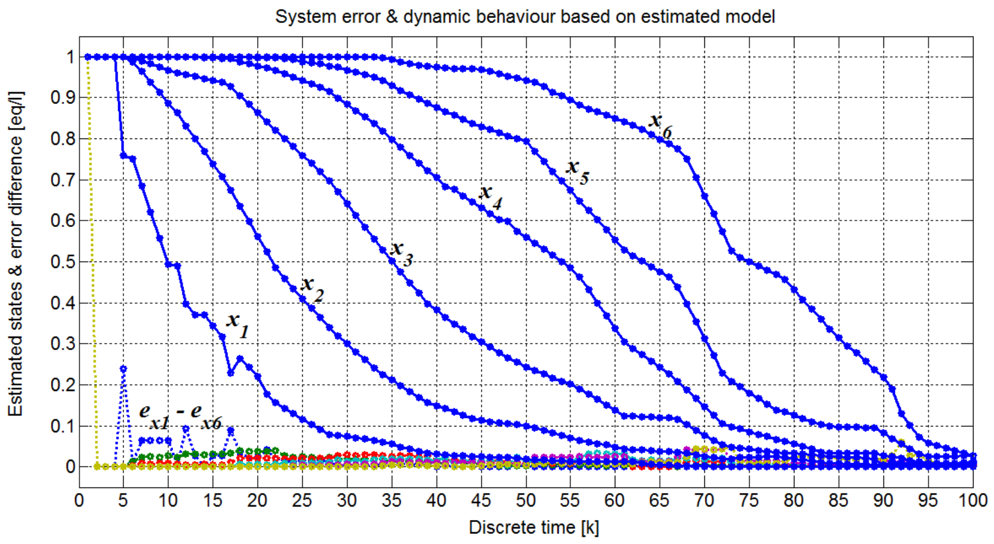


Figure 6.92. Model dynamic behaviour showing the error difference in relation to estimated states for Run 5 of Table 6.31

Table 6.32: Results of the third set of experiments for the changing Lagrange gradient step

Estimation criterion the least squares error and the estimation run time for the third set of experiments		
Case 3: (Third set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	6.2014	29.2970
Run 2	6.2014	30.8260
Run 3	6.2014	29.5460
Run 4	6.2014	29.8590
Run 5	6.2014	29.2650

Results and discussion of third set of experiments for the Lagrange based method

The results for Table 6.30 experiments are presented in Figures 6.89–6.92. The results show that at this best *parameters gradient step*, $\alpha_p = 0.5$, the *Lagrange multipliers gradient step* values have no much influence over the *estimated parameters* nor *the error difference* between calculated and measured states. All the runs produced very good least squares error and run times. The results from all the experiment runs are so close from one another such that there is no distinctive separation between all results.

The fourth set of experiments for the Lagrange based method

The fourth set of experiments considers both *the parameter* and *the Lagrange multipliers gradient steps* changing at the same rate, Table 6.33. The experiments results are demonstrated in Figures 6.93–6.96. Higher values of the *Lagrange multipliers gradient step* produce a higher error difference between calculated and estimated states.

Table 6.33: **The fourth set of experiments of both the gradient steps changing at the same rate**

Increasing and decreasing iteration gradient step values at the same rate						
Process parameter (units)	Values/ (Reference)	Run 1	Run 2	Run 3	Run 4	Run 5
$h = 0.3293\ l$	✓	✓	✓	✓	✓	✓
$H = 0.42809\ l$	✓	✓	✓	✓	✓	✓
$F_L = 2000/60\ (l/\min)$	✓	✓	✓	✓	✓	✓
$x_f = 0.435$	✓	✓	✓	✓	✓	✓
$a = [0.8\ 0.6\ 0.5\ 0.4\ 0.3\ 0.25]$	$a = 230 \times a$	✓	✓	✓	✓	✓
$b = [0.01\ 0.02\ 0.05\ 0.08\ 0.1\ 0.2]$	$b = 60 \times b$	✓	✓	✓	✓	✓
M	2000	✓	✓	✓	✓	✓
dt, sampling period (Δt)	0.012 min	✓	✓	✓	✓	✓
Parameters gradient step (α_p)	0.01	0.01	0.05	0.1	0.5	1.0
Lagrange multiplier gradient step (α_λ)	0.01	0.01	0.05	0.1	0.5	1.0
ε_p, epsilon- p (very small number)	0.001	✓	✓	✓	✓	✓
ε_λ, epsilon- $\lambda(k)$ (very small number)	0.001	✓	✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

The table shows both the *parameter* and *Lagrange multiplier iteration gradient steps* changing at the same rate.

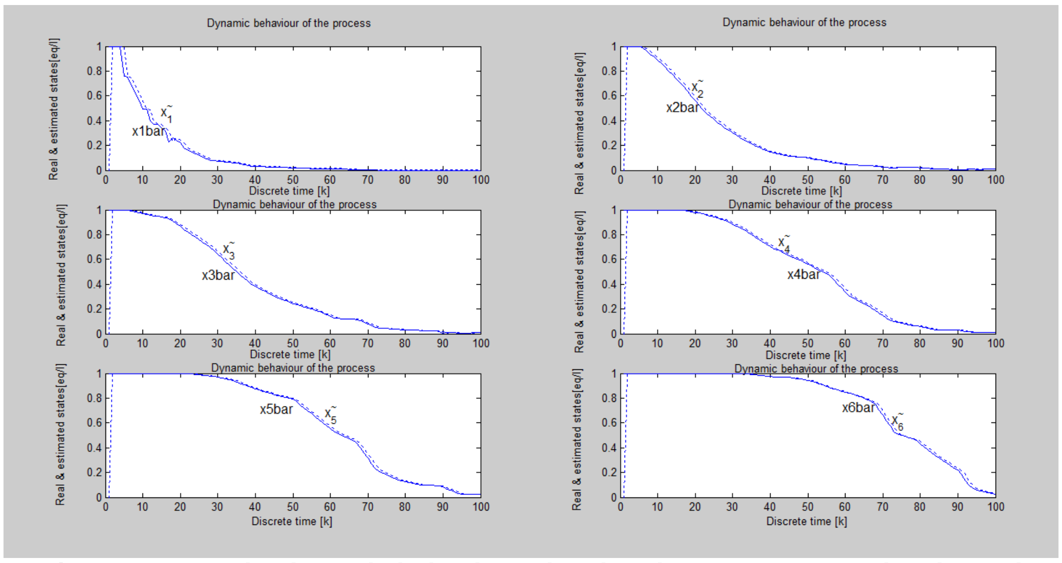


Figure 6.93. Model dynamic behaviour showing the overall real and estimated states for Run 1 of Table 6.33

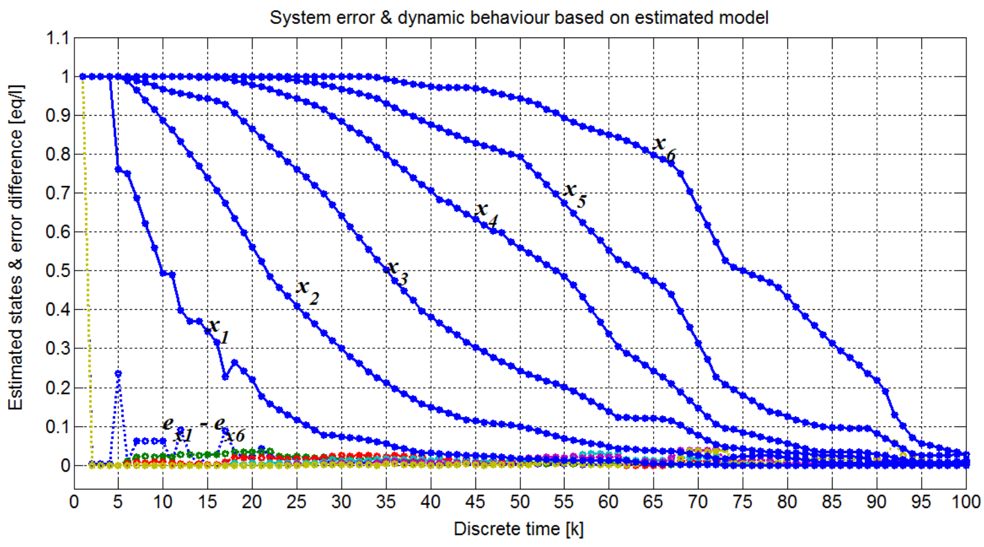


Figure 6.94. Model dynamic behaviour showing the error difference in relation to estimated states for Run 1 of Table 6.33

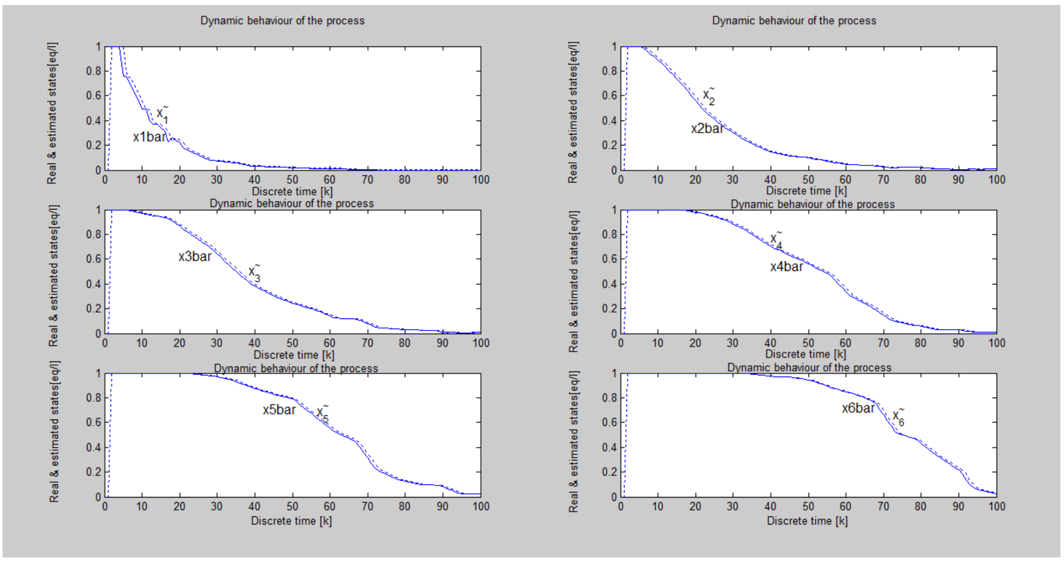


Figure 6.95. Model dynamic behaviour showing the overall real and estimated states for Run 5 of Table 6.33

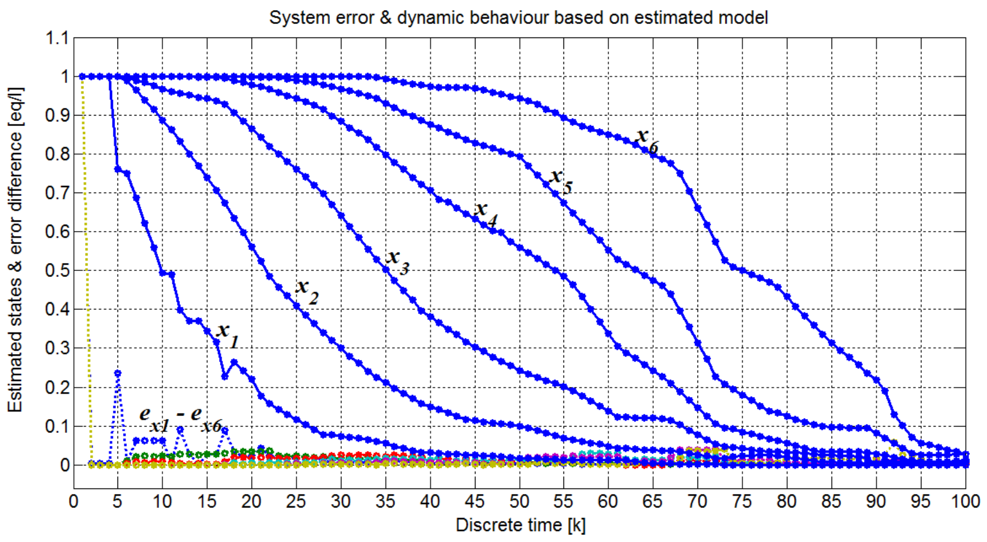


Figure 6.96. Model dynamic behaviour showing the error difference in relation to estimated states for Run 5 of Table 6.33

Table 6.34: Results of the fourth set of experiments of changing both the parameter and Lagrange gradient steps values

Estimation criterion the least squares error and the estimation run time for the fourth set of experiments		
Case 4: (Fourth set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	6.1765	30.8160
Run 2	6.1765	32.2260
Run 3	6.1765	31.0400
Run 4	6.1774	31.0460
Run 5	6.1806	31.1260

Results and discussion of the fourth set of experiments for the Lagrange based method

At very high gradient steps values the overall *gradient steps influence* on the error criterion, the calculation time and parameter values remain practically the same throughout all the runs. At very low gradient steps values, the tracking is the best, Table 6.34.

The fifth set of experiments of the Lagrange based method

The fifth set of experiments considers the changing initial parameter values at best sampling period and the *parameter gradient step* $\alpha_p = 0.5$ and the *Lagrange multipliers gradient step* of $\alpha_\lambda = 0.5$, Table 6.35. The results are presented in Figures 6.97–6.100. Higher initial parameter values improve the problem of most parameters settling around zero or close to zero.

Table 6.35: **The fifth set of experiments of the changing initial parameter values at best sampling period**

Changing initial values of model parameters						
Process parameter (units)	Values	Run 1	Run 2	Run 3	Run 4	Run 5
$h\ =0.3293\ l$	✓	✓	✓	✓	✓	✓
$H\ =0.42809\ l$	✓	✓	✓	✓	✓	✓
$F_L =2000/60\ l/\min$	✓	✓	✓	✓	✓	✓
$x_f =0.755$	✓	✓	✓	✓	✓	✓
$a=[0.8\ 0.7\ 0.5\ 0.4\ 0.3\ 0.02]$	$a=230\times a$	✓	✓	✓	✓	✓
$b=[0.025\ 0.1\ 0.2\ 0.3\ 0.4\ 0.6]$	$b=60\times b$	✓	✓	✓	✓	✓
Parameters (p)	$p=p$	$p=0.5p$	$p=1.5p$	$p=2.5p$	$p=10p$	$p=25p$
M	2000	✓	✓	✓	✓	✓
dt , sampling period (Δt)	2.35 min	✓	✓	✓	✓	✓
Parameters gradient step (α_p)	0.3	✓	✓	✓	✓	✓
Lagrange multiplier gradient step (α_λ)	0.01	✓	✓	✓	✓	✓
ε_p , epsilon- p (very small number)	0.001	✓	✓	✓	✓	✓
ε_λ , epsilon- $\lambda(k)$ (very small number)	0.001	✓	✓	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

The table shows the changing initial parameter values in multiples of 0.5, and not in any particular order from 0.5 to 25.

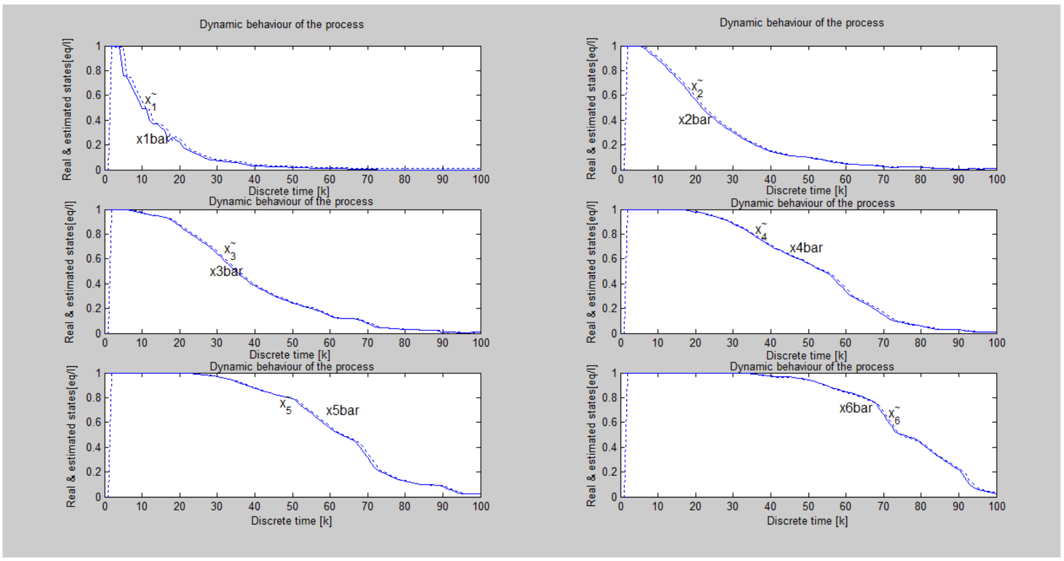


Figure 6.97. Model dynamic behaviour showing the overall real and estimated states for the best results – Run 3 Table 6.35

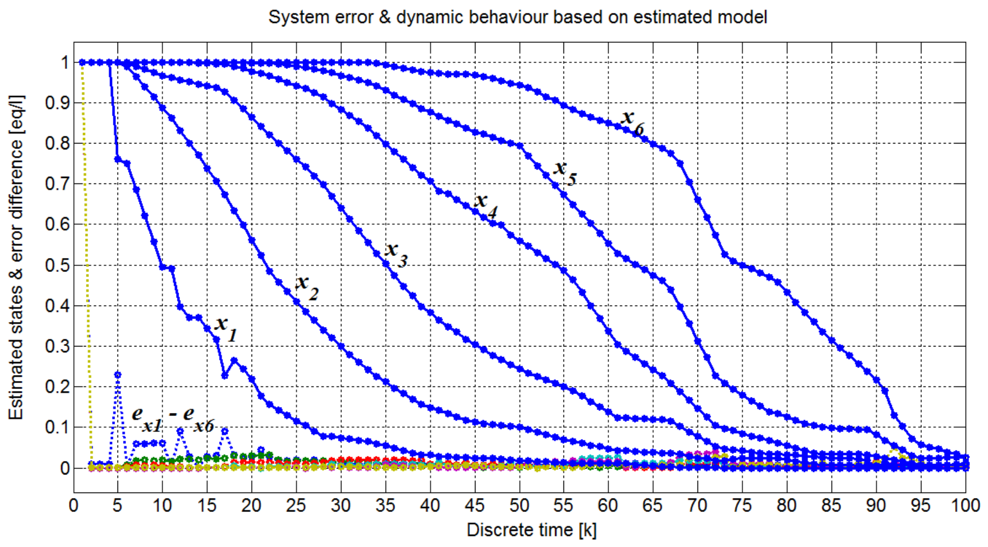


Figure 6.98. Model dynamic behaviour showing the error difference in relation to estimated states for the best results – Run 3 Table 6.35

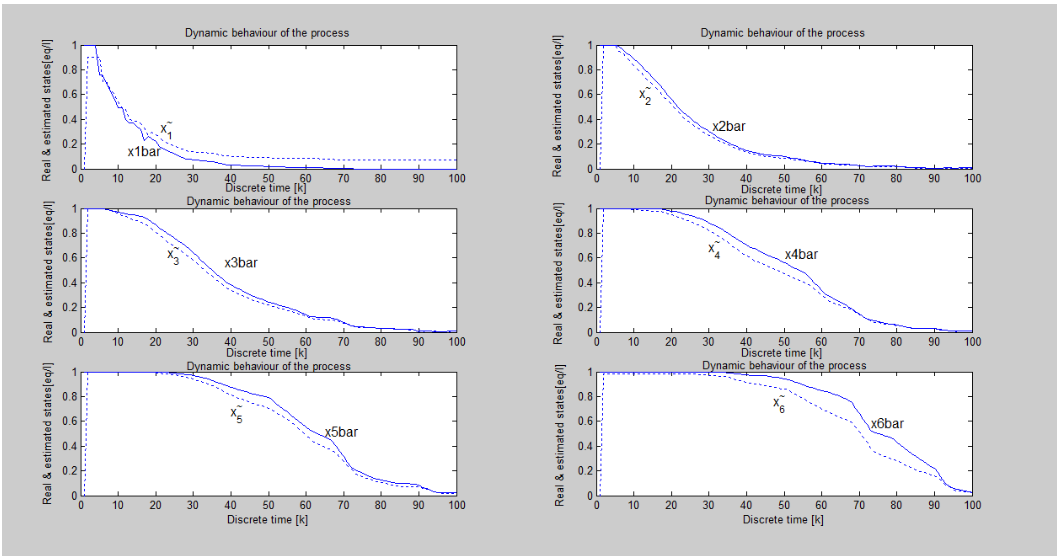


Figure 6.99. Model dynamic behaviour showing the overall real and estimated states for the worst case results – Run 5 Table 6.35

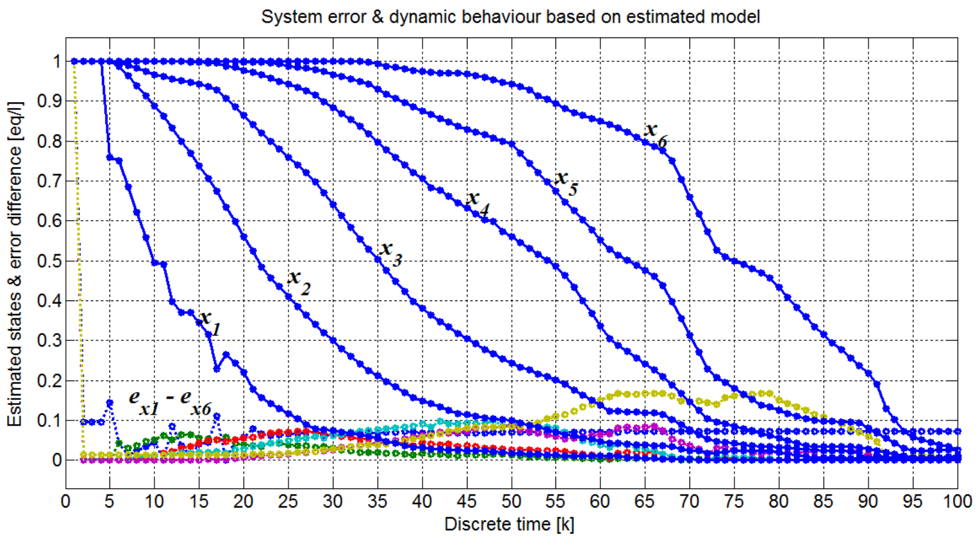


Figure 6.100. Model dynamic behaviour showing the error difference in relation to estimated states for the worst case results – Run 5 Table 6.35

Table 6.36: Results of the fifth set of experiments of changing the initial parameter values

Estimation criterion values and estimation run time for the fifth set of experiments		
Case 5: (Fifth set of experiments)	$J(p)$	Run time (t_{calc}) [sec]
Run 1	6.1895	29.5460
Run 2	6.1656	29.3280
Run 3	6.1496	29.6080
Run 4	6.2857	32.1710
Run 5	7.9117	31.6590

Results and discussion of the fifth set of experiments of the Lagrange based method

The best results for the changing initial parameter values in multiples of 0.5 is obtained at $p_0 = 2.5 p_0$, Table 6.36. At high multiples of initial parameter values, the *least square error* starts to grow drastically high.

6.8. Summary of the best results from all methods

Tables 6.37–6.40 show the collection of all the best results of each case for all the parameter estimation methods presented. Each table presents the different parameters varied for each case in an experiment set.

Table 6.37: Results from the process output based parameter estimation method

Method	Algorithm technique & the system parameter varied	Criterion	
Method 1	Process output (output from measured data)	$J(p)$	Run time (t_{calc}) [sec]
Case 1	Sampling period	20.2932	225.6700
Case 2	Model coefficients	22.9868	233.2180
Case 3	Model coefficients at reduced holdups	338.1485	229.3940
Case 4	Iteration gradient step	13.5571	257.0940
Case 5	Disturbance input	113.5492	253.3760
Case 6	Initial model parameters (p_0)	14.5746	257.1210

Table 6.38: Results from the least squares method of parameter estimation

Method	Algorithm technique & the system parameter varied	Criterion	
Method 2	Least squares method (state vector measurements)	$J(p)$	Run time (t_{calc}) [sec]
Case 1	Iteration gradient	6.0758	283.2140
Case 2	Sampling period	6.0786	234.2510
Case 3	Disturbance input	6.0839	241.7610
Case 4	Initial model parameters	6.3211	269.9240

Table 6.39: Results from the direct method (state vector measurement based parameter estimation)

Method	Algorithm technique & the system parameter varied	Criterion	
Method 3	Direct method (state vector measurements)	$J(p)$	Run time (t_{calc}) [sec]
Case 1	Sampling period	0.0236	0.0200
Case 2	Control input	0.0016	0.0150
Case 3	Disturbance input	0.0235	0.0150

Table 6.40: Results from the Lagrange based parameter estimation method

Method	Algorithm technique & the system parameter varied	Criterion	
Method 4	Lagrange method (state vector measurements)	$J(p)$	Run time (t_{calc}) [sec]
Case 1	Disturbance input	6.1682	30.4820
Case 2	Parameter gradient step	6.1774	30.2480
Case 3	Lagrange multipliers gradient step	6.2014	29.2650
Case 4	Parameters and multipliers gradient steps	6.1774	31.0460
Case 5	Initial model parameters	6.1496	29.6080

6.9. Discussion on all four methods

The overall best performance for each method is presented in Table 6.41. The best results are obtained from the direct method in all aspects from all cases. The direct method seems to outperform all the other methods by a very big margin if one considers the criteria used, the *least square error* and algorithm *processing time*. This could be associated with the fact that the direct method does not use the gradient iteration. The output based method has been the most poorly performing method. Its least square error is in the highest region and produced longest processing time.

Table 6.41: Results of the best criterion values from each method

Estimation criterion the least squares error and the estimation run time for the four estimation methods				
Method	Algorithm technique	Case	$J(p)$	Run time (t_{calc}) [sec]
1	Process output (output from measured data)	1	20.2932	225.6700
2	Least squares method (state vector measurements)	2	6.0786	234.9240
3	Direct method (state vector measurements)	2	0.0016	0.0150
4	Lagrange method (state vector measurements)	5	6.1496	29.6080

The state vector based least squares method performed in the same region as the Lagrangian based method in terms of the least squares error but has a very poor processing time. This make the Lagrange based parameter estimation method to be the second best performing followed by the state vector based least squares parameter estimation method.

6.10. Conclusion

Four different methods for the parameter estimation problem of the continuous countercurrent ion exchange (CCIX) process bilinear model have been presented.

The first method uses only the *measurable output vector* of the process. Its solution is based on the *gradient method of optimizing the error difference* between the model output and the process measurable output. The method considers that states are not measurable.

The second method is based on the possibility that the *full state of the process is measureable*. In this method, the model parameters are determined using the *least squares gradient minimization* procedure.

The third method is the direct method where the *state vector is known* and the model parameters are directly calculated using an *inverse matrix*.

The fourth method of solution is based on the *Lagrangian optimization procedure* to solve for the unknown parameters and the newly introduced Lagrangian vector.

The *direct state vector* based method produced the best results by far. In all different cases of the sets of experiments performed, the direct method performed extremely well. This method will be best in terms of real-time implementation of the estimation process. It does not involve a lot of calculation as long as its matrices are set correctly; it will produce best estimates in shortest time possible.

The next chapter considers the parameter estimation problem for the CCIX nonlinear model. The problem is solved using nonlinear formulation of the *separation factor* between the process’s sodium content (in liquid) and hydrogen content (in resin) of the exchanging streams. The chapter proposes a solution based on either *liquid phase* or *resin phase*.

CHAPTER SEVEN

METHOD FOR PARAMETER ESTIMATION (NONLINEAR MODEL)

CHAPTER SEVEN

7. METHOD FOR PARAMETER ESTIMATION (NONLINEAR MODEL)

7.1. Introduction

This chapter considers the second case of parameter estimation problem, the nonlinearity according to the parameters of the model. This is the second development of methods for parameter estimation as proposed in Chapter 5. Due to problems experienced with estimating parameters using the linear vector models that relate the *resin and liquid inter-phase connection*, a model that is based on only one phase, either liquid or resin is proposed. In this model, the relationship between the two ion exchange process phases (liquid and solid) is expressed through both phases but using the relevant *ionic cation* of interest, i.e., either sodium or hydrogen, depending on the available data. The advantage of this model is that there is not so much ionic inter-phase conversion required for changing from one phase to the other using assumed linear relationship.

The proposed process model directly relates the behaviour of one cation to the other using the *fractional interchange* between the resin and liquid phases. The drawback of this model is its nonlinear characteristics according to the model parameters. An algorithm for parameter estimation is developed for implementation using a MATLAB software program. The simulation results are presented at the end of the chapter.

In the first section of the chapter, the process model for nonlinear parameter estimation is developed. Design of nonlinear parameter estimation method using direct solution is proposed and simulation results presented. In the second section, the design of nonlinear parameter estimation method using Lagrange optimization technique is proposed based on matrices presentation.

7.2. Model reformulation for nonlinear parameter estimation

The relationship between *sodium in liquid* and *sodium in resin* presented using the *separation* factor is described by (Hendry, 1982a; Hendry, 1982d) as follows:

$$\underset{resin}{y_{Na}}(t) = \frac{y_H(t)x_{Na}(t)}{\alpha_H^{Na}x_H(t)} = \frac{x_{Na}(t)(1-y_{Na}(t))}{\alpha_H^{Na}(t)(1-x_{Na}(t))} \tag{7.1}$$

$$y_{Na}(t) = \frac{x_{Na}(t)}{\alpha_H^{Na}(t) - x_{Na}(t)\alpha_H^{Na}(t) + x_{Na}(t)} \tag{7.2}$$

where

$y_{\underset{resin}{Na}}(t)$ – is the equivalent ionic fraction of sodium ions (Na) in the resin phase, as seen from the resin phase perspective,

$y_{Na}(t)$ – is the equivalent ionic fraction of sodium ions (Na) in the resin phase, as seen from the liquid phase perspective,

$y_H(t)$ – is the equivalent ionic fraction of hydrogen ions (H) in the resin phase,

$x_{Na}(t)$ – is the equivalent ionic fraction of sodium ions (Na) in the liquid phase,

$x_H(t)$ – is the equivalent ionic fraction of hydrogen ions (H) in the liquid phase,

$\alpha_H^{Na}(t)$ – is the separation factor between the two exchanging cations.

Recalling the general model, based on the case where it is derived for the sodium ions:

$$\begin{aligned} &H\dot{x}_{Na,n}(t) + h\dot{y}_{Na,n}(t) = \\ &\quad = F_L[x_{Na,(n-1)}(t) - x_{Na,n}(t)] + F_R(t)[y_{Na,(n-1)}(t) - y_{Na,n}(t)], \quad n = \overline{1, N} \end{aligned} \tag{7.3}$$

where

h and H – are resin and liquid holdups in the exchanging solutions, respectively,

F_L – is the process liquid flow rate,

$F_R(t)$ – is the process resin flow rate,

$x_{Na,n}(t)$ – is the state variable representing the equivalent ionic fraction of sodium ions (Na) in the liquid phase, in the n^{th} stage,

$x_{Na,(n-1)}(t)$ – is the state variable representing the equivalent ionic fraction of sodium ions (Na) in the liquid phase, in the $(n-1)^{th}$ stage,

$y_{Na,n}(t)$ – is the state variable representing the equivalent ionic fraction of sodium ions (Na) in the resin phase, in the n^{th} stage,

$y_{Na,(n-1)}(t)$ – is the state variable representing the equivalent ionic fraction of sodium ions (Na) in the resin phase, in the $(n-1)^{th}$ stage.

Equation (7.2) is substituted to the main model equation, Equation (7.3) and the resulting model is nonlinear according to its parameters. Considering the model written for each stage of the column, the *separation facto*r will be different for each stage and therefore, every stage is considered independently in the following manner;

$$H\dot{x}_{Na,n}(t)+h\dot{y}_{Na,n}(t)=F_L\left[x_{Na,(n-1)}(t)-x_{Na,n}(t)\right]\;+F_R\left[y_{Na,(n+1)}(t)-y_{Na,n}(t)\right]\;, \quad n=\overline{1,N} \tag{7.4}$$

$$H\dot{x}_{Na,n}(t)+h\frac{d\left[\dfrac{x_{Na,n}(t)}{[\alpha^{Na}_{H,n}+x_{Na,n}(t)(1-\alpha^{Na}_{H,n})]}\right]}{dt}=$$
$$=F_L x_{Na,(n-1)}(t)-F_L x_{Na,n}(t)+F_R\frac{x_{Na,(n+1)}(t)}{[\alpha^{Na}_{H,(n+1)}+x_{Na,(n+1)}(t)(1-\alpha^{Na}_{H,(n+1)})]}-$$
$$-F_R\frac{x_{Na,n}(t)}{[\alpha^{Na}_{H,n}+x_{Na,n}(t)(1-\alpha^{Na}_{H,n})]}$$
$$n=\overline{1,N} \tag{7.5}$$

Using the first principle of differentiation, if $f(t)=f_1(t)/f_2(t)$ then the differentiation of $f(t)$ is given by $\dot{f}(t)=\dfrac{f_2(t)\,\dot{f}_1(t)-f_1(t)\,\dot{f}_2(t)}{[f_2(t)]^2}$.

Then if Equation (7.5) is consider $f_2(t)=x_{Na,n}(t)$ and $f_1(t)=[\alpha^{Na}_{H,n}+x_{Na,n}(t)(1-\alpha^{Na}_{H,n})]$. The model is then expanded to

$$H\dot{x}_{Na,n}(t)+$$
$$+h\left[\frac{[\alpha^{Na}_{H,n}+x_{Na,n}(t)(1-\alpha^{Na}_{H,n})]\dfrac{dx_{Na,n}(t)}{dt}-x_{Na,n}(t)\dfrac{d}{dt}[\alpha^{Na}_{H,n}+x_{Na,n}(t)(1-\alpha^{Na}_{H,n})]}{[\alpha^{Na}_{H,n}+x_{Na,n}(t)(1-\alpha^{Na}_{H,n})]^2}\right]=$$
$$=F_L x_{Na,(n-1)}(t)-F_L x_{Na,n}(t)+F_R\frac{x_{Na,(n+1)}(t)}{[\alpha^{Na}_{H,(n+1)}+x_{Na,(n+1)}(t)(1-\alpha^{Na}_{H,(n+1)})]}-$$
$$-F_R\frac{x_{Na,n}(t)}{[\alpha^{Na}_{H,n}+x_{Na,n}(t)(1-\alpha^{Na}_{H,n})]} \tag{7.6}$$

where $\alpha^{Na}_{H,n}=\alpha^{Na}_{H,n}(t)$

The above equation, Equation (7.6) can be simplified to,

$$H\dot{x}_{Na,n}(t)+h\left[\frac{\alpha^{Na}_{H,n}}{\left[\alpha^{Na}_{H,n}+x_{Na,n}(t)(1-\alpha^{Na}_{H,n}\right]^2}\right]\dot{x}_{Na,n}(t)=F_L x_{Na,(n-1)}(t)-$$
$$-F_L x_{Na,n}(t)+\frac{1}{[\alpha^{Na}_{H,(n+1)}+x_{Na,(n+1)}(t)(1-\alpha^{Na}_{H,(n+1)})]}F_R x_{Na,(n+1)}(t)- \tag{7.7}$$
$$-\frac{1}{[\alpha^{Na}_{H,n}+x_{Na,n}(t)(1-\alpha^{Na}_{H,n})]}F_R x_{Na,n}(t)$$

$$\left[\frac{H\left[\alpha_{H,n}^{Na}+x_{Na,n}(t)(1-\alpha_{H,n}^{na})\right]^2+h\alpha_{H,n}^{Na}}{\left[\alpha_{H,n}^{Na}+x_{Na,n}(t)(1-\alpha_{H,n}^{Na})\right]^2}\right]\dot{x}_{Na,n}(t)=$$
$$=\left[\begin{array}{l}F_L x_{Na,(n-1)}(t)-F_L x_{Na,n}(t)+\dfrac{1}{\left[\alpha_{H,(n+1)}^{Na}+x_{Na,(n+1)}(t)(1-\alpha_{H,(n+1)}^{Na})\right]}F_R x_{Na,(n+1)}(t)-\\ \qquad -\dfrac{1}{[\alpha_{H,n}^{Na}+x_{Na,n}(t)(1-\alpha_{H,n}^{Na})]}F_R x_{Na,n}(t)\end{array}\right]$$

(7.8)

Finally, rearranging the expression in Equation (7.8) results in

$$\dot{x}_{Na,n(}t)=\frac{\left[\alpha_{H,n}^{Na}+x_{Na,n}(t)(1-\alpha_{H,n}^{Na})\right]^2}{H\left[\alpha_{H,n}^{Na}+x_{Na,n}(t)(1-\alpha_{H,n}^{Na})\right]^2+h\alpha_{H,n}^{Na}}\times\left[-F_L x_{Na,(n-1)}(t)-\right.$$
$$-F_L x_{Na,n(}t)\ +\frac{1}{[\alpha_{H,(n+1)}^{Na}+x_{Na,(n+1)}(1-\alpha_{H,(n+1)}^{Na})]}F_R x_{Na,(n+1)}(t)- \qquad (7.9)$$
$$\left.-\frac{1}{[\alpha_{H,n}^{Na}+x_{Na,n}(t)(1-\alpha_{H,n}^{Na})]}F_R x_{Na,n}(t)\right]$$

The model is then presented in a discrete form to allow it to be used with a computer software program. The solution is based on the first principle of gradient determination, $\dot{x}(t)=\frac{x(k+1)-x(k)}{\Delta t}$, where Δt is the sampling period at every k th moment. The model in Equation (7.9) then becomes

$$x_{Na,n}(k+1)=x_{Na,n}(k)+$$
$$+\Delta t\left[\frac{\left[\alpha_{H,n}^{Na}+x_{Na,n}(k)(1-\alpha_{H,n}^{Na})\right]^2}{H\left[\alpha_{H,n}^{Na}+x_{Na,n}(k)(1-\alpha_{H,n}^{Na})\right]^2+h\alpha_{H,n}^{Na}}\times\left\{F_L x_{Na,(n-1)}(k)-\right.\right.$$
$$-F_L x_{Na,n}(k)+\frac{1}{[\alpha_{H,(n+1)}^{Na}x_{Na,(n+1)}(k)(1-\alpha_{H,(n+1)}^{Na})]}F_R x_{Na,(n+1)}(k)- \qquad (7.10)$$
$$\left.\left.-\frac{1}{[\alpha_{H,n}^{Na}+x_{Na,n}(k)(1-\alpha_{H,n}^{Na})]}F_R x_{Na,n}(k)\right\}\right]$$

The following functions of parameters are identified from the complex model in Equation (7.10) and presented in a simplified form in the state space representation:

$$f_{1,n}(k,x_{Na,n}(k),\alpha_{H,n}^{Na})=\left[\frac{\Delta t\left[\alpha_{H,n}^{Na}+x_{Na,n}(k)(1-\alpha_{H,n}^{Na})\right]^2}{H\left[\alpha_{H,n}^{Na}+x_{Na,n}(k)(1-\alpha_{H,n}^{Na})\right]^2+h\alpha_{H,n}^{Na}}\right] \qquad (7.11)$$

$$f_{2,n}(k,x_{Na,n}(k),\alpha_{H,n}^{Na})=\frac{\Delta t}{[\alpha_{H,n}^{Na}+x_{Na,n}(k)(1-\alpha_{H,n}^{Na})]} \qquad (7.12)$$

$$f_{3,(n+1)}(k, x_{Na,(n+1)}(k), \alpha_{H,n}^{Na}, \alpha_{H,(n+1)}^{Na}) = \frac{\Delta t}{[\alpha_{H,n}^{Na} + x_{Na,(n+1)}(k)(1-\alpha_{H,(n+1)}^{Na})]} \quad (7.13)$$

The unknown variable is the separation factor, $\alpha_{H,n}^{Na} = \alpha_{H,n}^{Na}(k)$ over the stages $n = \overline{1,N}$. The state variables $x_{Na,n}(k)$ are known from the measured data of the sodium content in the liquid, and the control variable is the resin flow rate $F_R(k)$. The state vector is therefore given by $x_{Na,n}(k) = [x_{Na,1}(k), x_{Na,2}(k), ..., x_{Na,N}(k)]^T$ and the control variable is defined based on the process upflow period (T) in the following approach, $u(k) = F_R(k) = \frac{hd}{T}$; where h is the resin holdups in each stage, and d is the fractional exchange coefficient relating the exchanging ions in the resin and the liquid. The state space equations are defined as follows,

$$\begin{aligned}
x_{Na,1}(k+1) &= x_{Na,1}(k) + f_{1,1}(k)F_L x_{Na,f}(k) - f_{1,1}(k)F_L x_{Na,1}(k) + \\
&\quad + f_{1,1}(k) f_{3,2}(k)F_R(k)x_{Na,2}(k) - f_{1,1}(k) f_{2,1}(k)F_R(k)x_{Na,1}(k) \\
x_{Na,2}(k+1) &= x_{Na,2}(k) + f_{1,2}(k)F_L x_{Na,1}(k) - f_{1,2}(k)F_L x_{Na,2}(k) + \\
&\quad + f_{1,2}(k) f_{3,3}(k)F_R(k)x_{Na,3}(k) - f_{1,2}(k) f_{2,2}(k)F_R(k)x_{Na,2}(k) \\
x_{Na,3}(k+1) &= x_{Na,3}(k) + f_{1,3}(k)F_L x_{Na,2}(k) - f_{1,3}(k)F_L x_{Na,3}(k) + \\
&\quad + f_{1,3}(k) f_{3,4}(k)F_R(k)x_{Na,4}(k) - f_{1,3}(k) f_{2,3}(k)x_{Na,3}(k) \\
x_{Na,4}(k+1) &= x_{Na,4}(k) + f_{1,4}(k)F_L x_{Na,3}(k) - f_{1,4}(k)F_L x_{Na,4}(k) + \\
&\quad + f_{1,4}(k) f_{3,5}(k)F_R(k)x_{Na,5}(k) - f_{1,4}(k) f_{2,4}(k)F_R(k)x_{Na,4}(k) \\
x_{Na,5}(k+1) &= x_{Na,5}(k) + f_{1,5}(k)F_L x_{Na,4}(k) - f_{1,5}(k)F_L x_{Na,5}(k) + \\
&\quad + f_{1,5}(k) f_{3,6}(k)F_R(k)x_{Na,6}(k) - f_{1,5}(k) f_{2,5}(k)F_R(k)x_{Na,5}(k) \\
x_{Na,6}(k+1) &= x_{Na,6}(k) + f_{1,6}(k)F_L x_{Na,5}(k) - f_{1,6}(k)F_L x_{Na,6}(k) + 0 + \\
&\quad - f_{1,6}(k) f_{2,6}(k)F_R(k)x_{Na,6}(k)
\end{aligned} \quad (7.14)$$

In a matrix form the state space representation of the model is as follows,

$$x_{Na,n}(k+1) =$$

$$= \begin{bmatrix} (1-f_{1,1}F_L) & 0 & 0 & 0 & 0 & 0 \\ f_{1,2}F_L & (1-f_{1,2}F_L) & 0 & 0 & 0 & 0 \\ 0 & f_{1,3}F_L & (1-f_{1,3}F_L) & 0 & 0 & 0 \\ 0 & 0 & f_{1,4}F_L & (1-f_{1,4}F_L) & 0 & 0 \\ 0 & 0 & 0 & f_{1,5}F_L & (1-f_{1,5}F_L) & 0 \\ 0 & 0 & 0 & 0 & f_{1,6}F_L & (1-f_{1,6}F_L) \end{bmatrix} \times$$

$$\times x_{Na,n}(k) +$$

$$+\begin{bmatrix} -f_{1,1}f_{2,1} & f_{1,1}f_{3,2} & 0 & 0 & 0 & 0 \\ 0 & -f_{1,2}f_{2,2} & f_{1,2}f_{3,3} & 0 & 0 & 0 \\ 0 & 0 & -f_{1,3}f_{2,3} & f_{1,3}f_{3,4} & 0 & 0 \\ 0 & 0 & 0 & -f_{1,4}f_{2,4} & f_{1,4}f_{3,5} & 0 \\ 0 & 0 & 0 & 0 & -f_{1,5}f_{2,5} & f_{1,5}f_{3,6} \\ 0 & 0 & 0 & 0 & 0 & -f_{1,6}f_{2,6} \end{bmatrix}\times$$
$$\times x_{Na,n}(k)F_R(k)+\left[f_{1,1}F_L \quad 0 \quad \ldots \quad 0\right]^T x_{Na,f}(k)$$

(7.15)

where $f_{i,j} = f_{i,j}(k)$, $i=\overline{1,3}$ and $j=\overline{1,6}$

$x_{Na,f}(k)=x_f(k)=w(k)$ – is the disturbance input representing the changing sodium content in the feed concentration,

$x_{Na,n}(k)=\left[x_{Na,1}(k) \quad x_{Na,2}(k) \quad x_{Na,3}(k) \quad x_{Na,4}(k) \quad x_{Na,5}(k) \quad x_{Na,6}(k)\right]^T$.

In general form, the nonlinear model in Equation (7.15) can be expressed as

$$\begin{aligned} x_{Na}(k+1) = F(k,\alpha_H^{Na})x_{Na}(k) + \\ + G(k,\alpha_H^{Na})F_R(k)x_{Na}(k)+Q(k,\alpha_H^{Na})x_{Na,f}(k) \end{aligned} \qquad (7.16)$$

where

$F(k)=$

$$=\begin{bmatrix} (1-f_{1,1}F_L) & 0 & 0 & 0 & 0 & 0 \\ f_{1,2}F_L & (1-f_{1,2}F_L) & 0 & 0 & 0 & 0 \\ 0 & f_{1,3}F_L & (1-f_{1,3}F_L) & 0 & 0 & 0 \\ 0 & 0 & f_{1,4}F_L & (1-f_{1,4}F_L) & 0 & 0 \\ 0 & 0 & 0 & f_{1,5}F_L & (1-f_{1,4}F_L) & 0 \\ 0 & 0 & 0 & 0 & f_{1,6}F_L & (1-f_{1,6}F_L) \end{bmatrix}$$

$$G(k)=\begin{bmatrix} f_{1,1}f_{2,1} & -f_{1,1}f_{3,2} & 0 & 0 & 0 & 0 \\ 0 & f_{1,2}f_{2,2} & -f_{1,2}f_{3,3} & 0 & 0 & 0 \\ 0 & 0 & f_{1,3}f_{2,3} & -f_{1,3}f_{3,4} & 0 & 0 \\ 0 & 0 & 0 & f_{1,4}f_{2,4} & -f_{1,4}f_{3,5} & 0 \\ 0 & 0 & 0 & 0 & f_{1,5}f_{2,5} & -f_{1,5}f_{3,6} \\ 0 & 0 & 0 & 0 & 0 & f_{1,6}f_{2,6} \end{bmatrix}$$

$$Q(k)=\left[f_{1,1}F_L \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\right]^T$$

The model, Equation (7.15), has N unknown coefficients $\alpha_{H,n}^{Na}(k)$, $n=\overline{1,N}$. The measurements for $x_{Na,n}(k)$, $n=\overline{1,N}$ are known from the measurement data, this means that $x_{Na}(k+1)$ and $x_{Na}(k)$ are known. The control variable is also known

$F_R(k) = \frac{hd}{T}$, where $d = 2.300$, $h = 0.4906$ and $T = 0.6378$. When these known variables are substituted in the model Equation (7.15), only the values of the *separation factor* $\alpha_{H,n}^{Na}$ are unknown.

There are two possible approaches to solving the problem for estimation of the values of the separation factor,

1) *Direct method of using MATLAB* nonlinear function, *fsolve* in order to determine the unknown separation factor values, or
2) Using parameter *estimation method based on a Lagrange's functional*.

7.3. Nonlinear model parameter estimation of the separation factor

7.3.1. Nonlinear model parameter estimation of the separation factor – the direct method

The direct method used in solving the nonlinear parameter estimation problem is based on a MATLAB software $fsolve(\,)$ function. The function determines a root (zero) of a system of nonlinear equations. The $fsolve(\,)$ function has a number of variations in solving optimization problems. The variations are presented here below as given by Mathworks (2001):

$x = fsolve(\text{fun},x_0)$,	*starts at x_0 and tries to solve the equations* described in fun.
$x = fsolve(\text{fun},x_0,\text{options})$,	*minimizes with the optimization parameters* specified in the structure options.
$x = fsolve(\text{fun},x_0,\text{options},p_1,p_2,...)$,	*passes the problem-dependent parameters* p_1, p_2, etc., *directly to the function fun*. Pass an empty matrix for options to use the default values for options.
$[x,fval] = fsolve(\text{fun},x_0)$,	*returns the value of the objective function fun* at the solution x.
$[x,fval,exitflag] = fsolve(...)$,	*returns a value exitflag that describes the exit condition*.
$[x,fval,exitflag,output] = fsolve(...)$,	*returns a structure output that contains information about the optimization*.
$[x,fval,exitflag,output,Jacobian] = fsolve(...)$,	*returns the Jacobian of a function* at the solution x.

The basic $fsolve()$ function listed here above as $x = fsolve(\text{fun}, x_0, \text{options})$ was used since there were no parameters being passed to th function and the interest was simply to find the caluse of unknown parameter α_{Na} with the display options of the method applied and the total number of iterations used.

Optimization options used with $fsolve()$ function

$fsolve()$ uses a few algorithms, depending on the size of the optimization problem (Mathworks, 2001):

1) the Line-*Search* method,
2) the Large-*Scale* algorithm,
3) the Medium-*Scale* algorithm, and
4) the Large-*Scale* and *Medium-Scale* algorithms.

Optimization options parameters used by *fsolve* function apply to all algorithms, some are only relevant when using the Large-Scale algorithm, and others are only relevant when using the Medium-Scale algorithm. The Large-Scale algorithm option states a preference for which algorithm to be used. It is only a preference since certain conditions must be met to use the large-scale algorithm. Some of these optimization options are (Mathworks, 2001):

1) Diagnostics
   Print diagnostic information about the function to be minimized.
2) *Display*
   Level of display. '*off*' displays no output; '*iter*' displays output at each iteration; '*final*' (default) displays just the final output.
3) *Jacobian*
   If switched 'ON', *fsolve* uses a user-defined Jacobian (defined in *fun*), or Jacobian information (when using JacobMult), for the objective function. If 'OFF', *fsolve* approximates the Jacobian using finite differences.
4) *MaxFunEvals*
   Maximum number of function evaluations allowed.
5) *MaxIter*
   Maximum number of iterations allowed.
6) *TolFun*
   Termination tolerance on the function value.
7) *TolX*
   Termination tolerance on x.

Input and output arguments of the $fsolve()$ function

A number of input arguments are used with *fsolve* function, some have been presented above. The most common are *fun*, *options* and *vector parameters*, etc., more information on different input arguments can be obtained in Mathworks (2001).

The function returns the following arguments as its outputs: *iterations*, *funcCount*, *algorithm*, *cgiterations*, *stepsize*, and *firstorderopt*. These arguments are defined respectively to stand for, the number of iterations taken to reach the optimal solution, the number of function evaluations, the type of algorithm used, the number of *Preconditioned Conjugate Gradients* method (PCG) iterations (for large-scale algorithm only), the final step size taken (for medium-scale algorithm only) to reach optimal solution, and measure of first-order optimality (large-scale algorithm only) and for large scale problems, the first-order optimality is the infinity norm of the gradient.

7.3.2. The procedure for the direct method using the $fsolve()$ function

The procedure for calculation is divided into two according to the main program and the solution of the *fsolve* () function.

1. The main program:
   - 1.1) Declare the total number of the system trajectory points K according to the available measurements,
   - 1.2) Set the initial value of $\alpha_H^{Na} = \alpha_H^{Na}(0)$, the variable to be optimized,
   - 1.3) Declare the required options for minimization procedure using $optimset()$ function as described in Section 7.3.1.
   - 1.4) Call $fsolve()$ function to obtain calculated optimal value of the separation factor, $\alpha_H^{Na}(k)$.

2. The *fsolve* () function program:
   - 2.1) Declare the function name for the separation factor, *falpha*,
   - 2.2) Initialize the system parameters, h, H, F_L, d, the initial values of the input $F_R(k)$, the disturbance $x_f(k)$ and the states, $x_0(k) = x_0$,
   - 2.3) Project the input, the disturbance and the states for the full trajectory, $k = \overline{1, K}$,
   - 2.4) For each stage calculate the separation factor individually,
     - 2.4.1) for $k = \overline{1, K}$,

» calculate $f_{1,n}$, Equation (7.11),

» calculate $f_{2,n}$, Equation (7.12),

» calculate $f_{3,n}$, Equation (7.13),

2.4.2) Calculate the optimal separation factor α_H^{Na} and return its value to the *falpha* function.

3. End calculation.

7.3.3. The flowchart for nonlinear model parameter estimation

1) The main program flowchart (Figure 7.1)

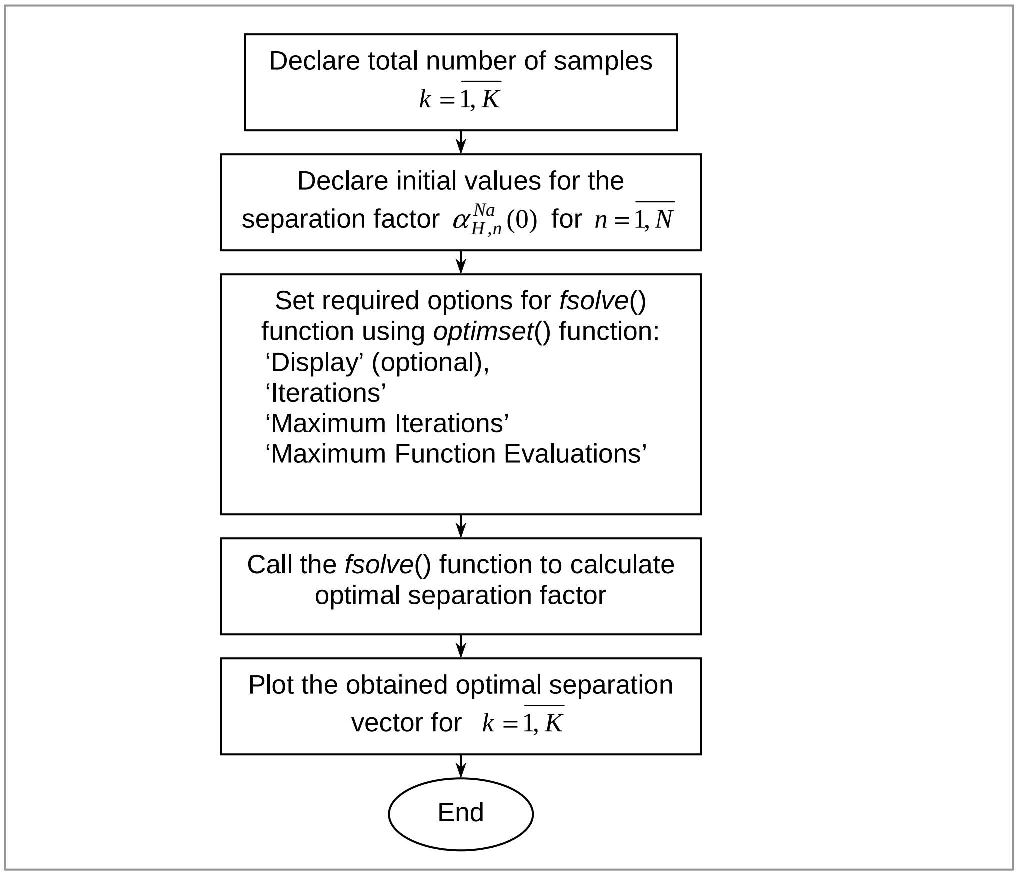

Declare total number of samples $k=\overline{1,K}$
Declare initial values for the separation factor $\alpha_{H,n}^{Na}(0)$ for $n=\overline{1,N}$
Set required options for *fsolve*() function using *optimset*() function: 'Display' (optional), 'Iterations' 'Maximum Iterations' 'Maximum Function Evaluations'
Call the *fsolve*() function to calculate optimal separation factor
Plot the obtained optimal separation vector for $k=\overline{1,K}$
End

Figure 7.1. **The main program flowchart for nonlinear model parameter estimation**

2) The *fsolve*() function calling routine (Figure 7.2)

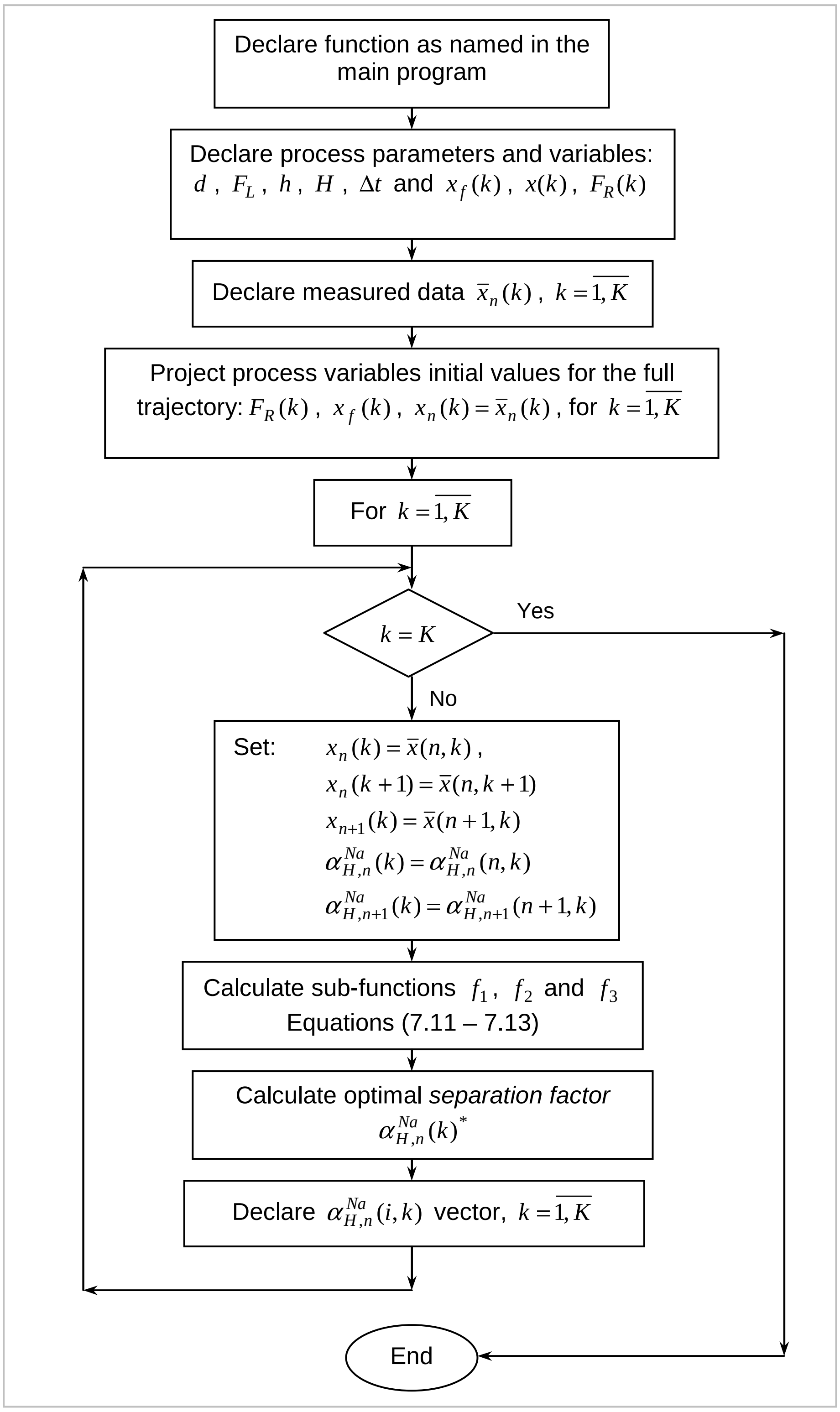

Declare function as named in the main program
Declare process parameters and variables: d, F_L, h, H, Δt and $x_f(k)$, $x(k)$, $F_R(k)$
Declare measured data $\bar{x}_n(k)$, $k=\overline{1,K}$
Project process variables initial values for the full trajectory: $F_R(k)$, $x_f(k)$, $x_n(k)=\bar{x}_n(k)$, for $k=\overline{1,K}$
For $k=\overline{1,K}$
$k=K$
Yes
No
Set: $x_n(k)=\bar{x}(n,k)$, $x_n(k+1)=\bar{x}(n,k+1)$, $x_{n+1}(k)=\bar{x}(n+1,k)$, $\alpha_{H,n}^{Na}(k)=\alpha_{H,n}^{Na}(n,k)$, $\alpha_{H,n+1}^{Na}(k)=\alpha_{H,n+1}^{Na}(n+1,k)$
Calculate sub-functions f_1, f_2 and f_3 Equations (7.11 – 7.13)
Calculate optimal *separation factor* $\alpha_{H,n}^{Na}(k)^*$
Declare $\alpha_{H,n}^{Na}(i,k)$ vector, $k=\overline{1,K}$
End

Figure 7.2. **The *fsolve*() function calling routine flowchart for nonlinear model parameter estimation**

7.3.4. The experiments of the nonlinear model parameter estimation problem using the *fsolve* MATLAB function

A limited number of experiments were performed under the direct method described in Chapter 6 due to complexity in the developed model. *Initial values of the separation factor*, the *disturbance input* and the *sampling period* were incremented and decremented to observe their influence on the overall performance of the process. The criterion for evaluating these experiments is based on the *fsolve* function *iteration index* and the *system processing time*.

The first set of experiments of the direct method (changing the initial values of the separation factor)

The first set of experiments is concerned with changing the *initial values of the separation factor*, Table 7.1. The results of this set of experiments are presented in Table 7.2 and Figures 7.3 and 7.4.

Table 7.1: Experiment for changing the system noise at constant measurement noise

Control input	Disturbance input	Separation factor (Ref.)	Changing initial separation factor values			
$F_R(k)$	$x_f(k)$ [l/min]	$\alpha_H^{Na}(0)$	Run 1	Run 2	Run 3	Run 4
1.1088	1.00	1.0	$0\times\alpha_H^{Na}$	$0.1\times\alpha_H^{Na}$	$0.5\times\alpha_H^{Na}$	$1.0\times\alpha_H^{Na}$

$$\alpha_H^{Na}(0) = [1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0]^T$$

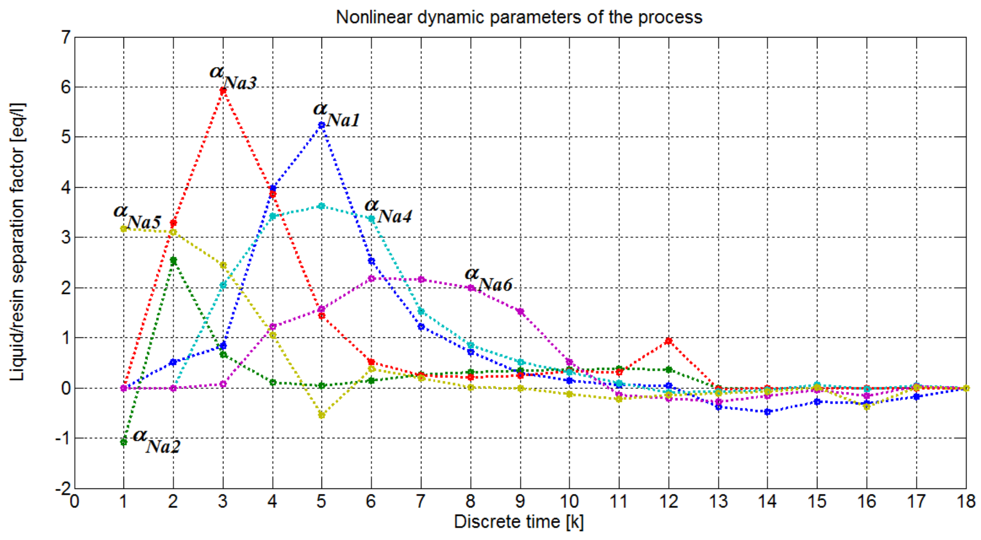


Figure 7.3. The best results of estimated separation factor values from Run 1 of Table 7.1.

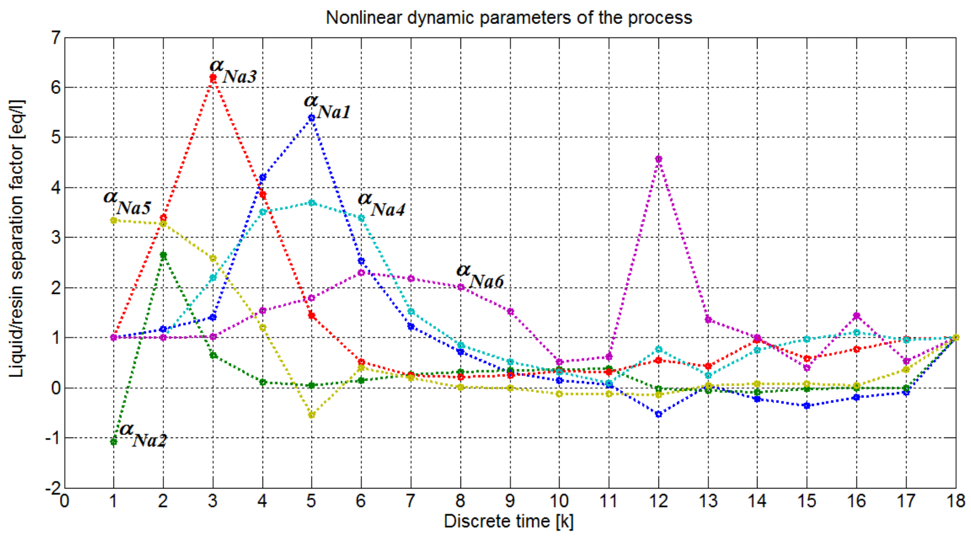


Figure 7.4. The worst case results for the estimated separation factor values from Run 4 of Table 7.1.

Table 7.2: Results of the first set of experiments for changing the system noise at constant measurement noise

Estimation criterion based on iteration value and the estimation run time for the first set of experiments		
Case 1: (First set of experiments)	Iteration value	Run time (t_{calc}) [sec]
Run 1	892	249.8920
Run 2	–	47.4210
Run 3	891	266.9290
Run 4	892	268.7320

– means optimizer is stuck in the local minimum

Results and discussion of the first set of experiments

The best results are received at separation factor initial conditions of zero, $\alpha_H^{Na}(0)=0$. At higher values of initial conditions, the system performs poorly in terms of the processing time, Table 7.2. The total number of iterations are the same for all the runs.

The second set of experiments of the direct method (changing the values of the disturbance input)

The second set of experiments is concerned with changing the *value of the disturbance input*, Table 7.3. The results of this set of experiments are presented in Table 7.4 and Figures 7.5 and 7.6.

Table 7.3: Experiment for changing the value of the disturbance input

Changing disturbance input		Sampling period	Control input	Separation factor
	$x_f(k)$	Δt [min]	$F_R(k)$ [l/min]	$\alpha_H^{Na}(0)$
Reference values	$x_f = 0.435$	2.7	1.1088	0
Run 1	$x_f = 0.055$	✓	✓	✓
Run 2	$x_f = 0.110$	✓	✓	✓
Run 3	$x_f = 0.550$	✓	✓	✓
Run 4	$x_f = 0.755$	✓	✓	✓
Run 5	$x_f = 1.00$	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

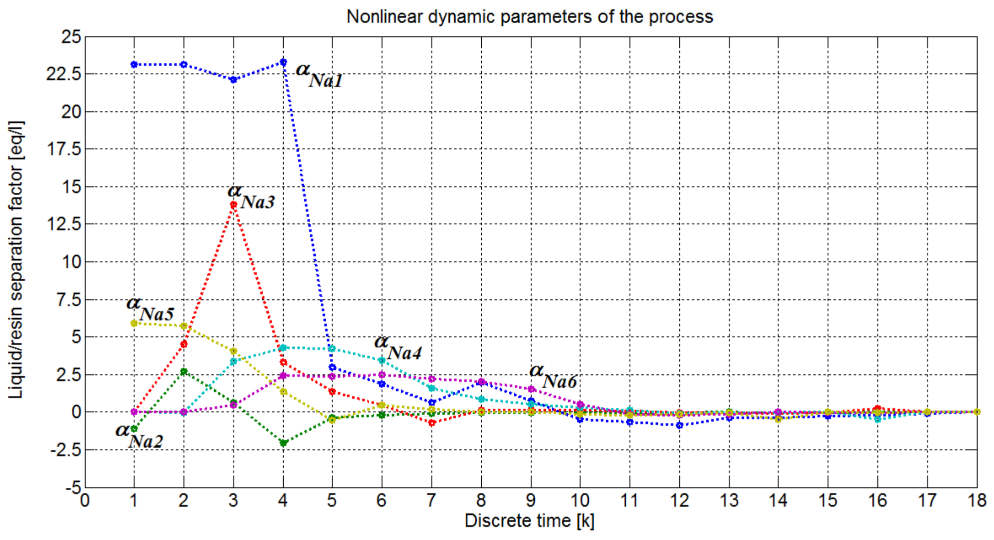


Figure 7.5. The worst case results for the estimated separation factor values from Run 1 of Table 7.3.

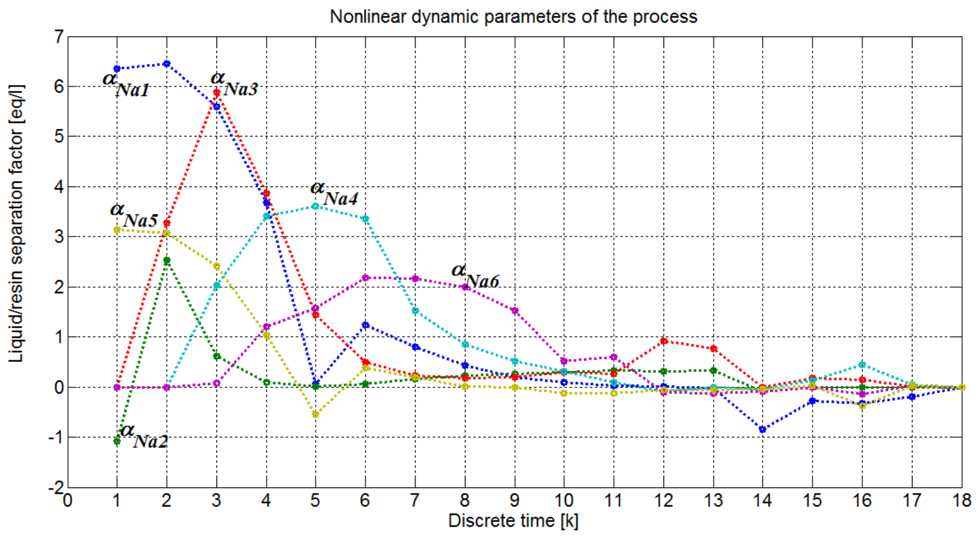


Figure 7.6. The best results of estimated separation factor values from Run 4 of Table 7.3.

Table 7.4: Results of the second set of experiments for changing the disturbance input value

Estimation criterion based on iteration value and the estimation run time for the second set of experiments		
Case 2: (Second set of experiments)	Iteration value	Run time (t_{calc}) [sec]
Run 1	891	239.1690
Run 2	–	66.2090
Run 3	892	236.0650
Run 4	892	230.3500
Run 5	892	229.8500

– means optimizer is stuck in the local minimum

Results and discussion of the second set of experiments

The best and the worst results for the changing disturbance input are presented in Figures 7.5 and 7.6. At lower values of the disturbance input, the estimation procedure tends to perform poorly and better results are obtained at higher values of the disturbance input in terms of processing time, Table 7.4. The first stage of the process is greatly affected by the disturbance input since this is where this input enters the process; this is demonstrated by the sharp climb of the sparation factor in all experiments, Figures 7.5 and 7.6.

The third set of experiments of the direct method (changing the values of the sampling period)

The third set of experiments is changing the *value of the sampling period*, Table 7.5. The results of this set of experiments are presented in Table 7.6 and Figures 7.7 and 7.8. Though the earlier experiment indicates that results for best results are obtained at disturbance of $x_f = 1.0$, this set of experiments considers the worse case disturbance because it should indicate the best sampling period at the worse case disturbance.

Table 7.5: Experiment of changing the value of the sampling period

Changing sampling period		Disturbance input	Control input	Separation factor
	Δt [min]	$x_f(k)$	$F_R(k)$ [l/min]	$\alpha_H^{Na}(0)$
Reference values	$\Delta t = 2.7\,\text{min}$	$x_f = 1.000$	1.1088	0
Run 1	$\Delta t = \Delta t$	✓	✓	✓
Run 2	$0.5 \times \Delta t$	✓	✓	✓
Run 3	1.00	✓	✓	✓
Run 4	0.50	✓	✓	✓
Run 5	$0.1 \times \Delta t$	✓	✓	✓

✓ means the value stays the same as in the previous mention in the table

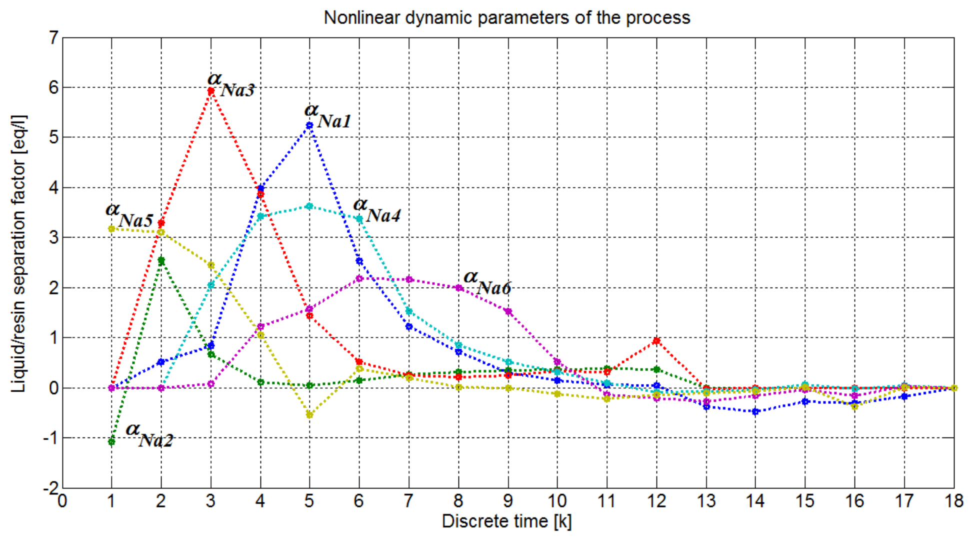


Figure 7.7. The worst case results of estimated separation factor values from Run 1 of Table 7.5.

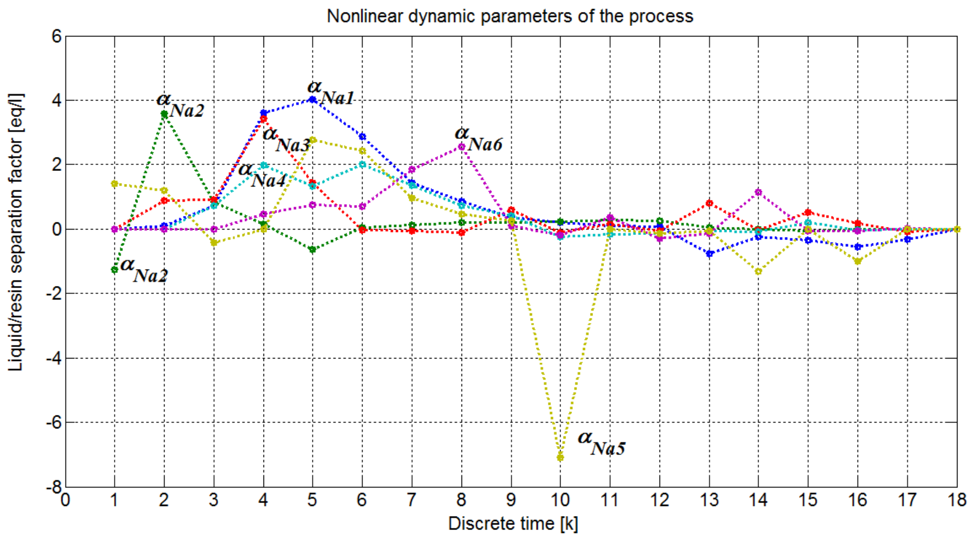


Figure 7.8. The best results of estimated separation factor values from Run 3 of Table 7.5.

Table 7.6: Results of the third set of experiments for changing the sampling period

Estimation criterion based on iteration value and the estimation run time for the third set of experiments		
Case 3: (Third set of experiments)	Iteration value	Run time (t_{calc}) [sec]
Run 1	892	251.3550
Run 2	–	226.1500
Run 3	892	244.9100
Run 4	893	246.4920
Run 5	–	169.4590

– means optimizer is stuck in the local minimum

Results and discussion of the third set of experiments

The experiment results for the changing sampling period's influence of the nonlinear model are presented in the Figures 7.7 and 7.8. The best results in terms of the evaluation criteria were obtained at the sampling period $\Delta t = 1.00\,\mathrm{min}$, Table 7.6. At lower sampling period or faster sampling, the estimation procedure performed poorly; this is linked to the measurement data used, which was sampled at higher values (Hendry, 1982a). There is an unexpected sharp decrement of the estimated values of the separation factor $\alpha_{Na} = 5$, the fifth stage around the sampling period $k = 10$; though this could not be clearly explained, it could be related to data being used and possibly the switching of techniques within $fsolve(\,)$ function.

7.3.5. Remarks on nonlinear parameter estimation using the *fsolve* function of MATLAB software

In calculating the nonlinear parameters of the model using the *direct method*, some problems were initially encountered with the usage of $fsolve(\,)$ function. These problems were not experienced when calculating a minimum based on a single dimension scalar, but were encountered when vectors were introduced. It was later observed that they related to initialization of the unknown variable, and the available data. It is recommended that care should be taken when initializing the unknown variable, more especially in the $mfile$. Reliable data also plays an important role in the success of the optimization procedure, if unreliable data is used, problems may arise. Data applied could have had some effect in some of the results that were poor.

7.4. Parameter estimation of the separation factor for the nonlinear model – using Lagrange's estimation technique

7.4.1. Formulation of the problem

Mathematically, the problem for estimation of the parameters $\alpha_{H,n}^{Na}(k)$, $n = \overline{1,N}$, $k = \overline{1,K}$ is formulated as follows: Find the values of the separation factor vector $\alpha_{H}^{Na}(k)$ such that the criterion, for $k = \overline{1,K}$:

$$J = \sum_{k=1}^{K} \left[\left\| e_{Na}(k) \right\|_Q^2 \right] = \sum_{k=1}^{K} \left\| x_{Na,meas}(k) - \hat{x}_{Na}(k) \right\|_Q^2 \tag{7.17}$$

is minimized under the model

$$x_{Na}(k+1) = F(k,\alpha_H^{Na})x_{Na}(k) + G(k,\alpha_H^{Na})F_R(k,\alpha_H^{Na}) + Q(k,\alpha_H^{Na})x_{Na,f}(k) \tag{7.18}$$

where

J – is the minimization criterion for calculating the error,

$e_{Na}(k)$ – is the error difference between the measured and the estimated states of the system,

$\hat{x}_{Na}(k)$ – is the vector of the process states calculated using the separation factor parameter estimates $\alpha_{H,n}^{Na}(k)$ for $n = \overline{1, N}$, and

$x_{Na,meas}(k)$ – is the vector of the measured data based on the sodium content of the liquid for the period.

The problem defined by Equations (7.17) and (7.18) is solved using the Lagrange's functional defined by the model equation and the cost function.

$$L_a = J + \lambda^T(k)\begin{bmatrix} -x_{Na}(k+1) + F(k,\alpha_H^{Na})x_{Na}(k) + \\ +G(k,\alpha_H^{Na})F_R(k)x_{Na}(k) + Q(k,\alpha_H^{Na})x_f(k)\end{bmatrix} \tag{7.19}$$

where

L_a – is the Lagrange function used for the minimization procedure,

$\lambda(k)$ – is the vector of Lagrange multipliers, $k = \overline{1, K}$ and $\lambda(k) \in R^N$.

According to the necessary conditions for optimality, the following conditions are to be met, for every moment $k = \overline{1, K}$

$$\frac{\partial L_a}{\partial \alpha_H^{Na}(k)} = e_{\alpha_H^{Na}}(k) = 0 \tag{7.20}$$

$$\frac{\partial L_a}{\partial \lambda(k)} = e_{\lambda}(k) = 0 \tag{7.21}$$

where

$e_{\alpha_H^{Na}}(k)$ – is the minimum error of the derivative of the Lagrange function based on the separation factor,

$e_{\lambda}(k)$ – is the minimum error of the derivative of the Lagrange function based on the Lagrange multipliers

A *gradient procedure* is used to obtain the optimal values of the unknown vectors $\alpha_H^{Na}(k)$ and $\lambda(k)$. The improved values of the unknown parameters are determined in the procedure by,

$$\overset{(j+1)}{\alpha_H^{Na}}(k) = \overset{(j)}{\alpha_H^{Na}}(k) - \Delta_{\alpha_H^{Na}} \frac{\partial L_a}{\partial \alpha_H^{Na}(k)} \tag{7.22}$$

$$\lambda^{(j+1)}(k) = \lambda^{j}(k) + \Delta_{\lambda} \frac{\partial L_a}{\partial \lambda(k)} \tag{7.23}$$

where

$\overset{(j+1)}{\alpha_H^{Na}}(k)$ – is the improved separation factor at the next iteration of the procedure,

$\lambda^{(j+1)}(k)$ – is the improved Lagrange multipliers at the next iteration of the procedure,

$\overset{(j)}{\alpha_H^{Na}}(k)$ – is the separation factor at the current iteration of the procedure,

$\lambda^{j}(k)$ – is the the Lagrange multiplier at the current iteration of the procedure,

$\Delta_{\alpha_{H,n}^{Na}}$ – is the gradient step for the *separation factor*, and

Δ_{λ} – is the gradient step for the *Lagrange multipliers*.

The derivatives Equations (7.20) and (7.21) are expressed from the Lagrangian Equation (7.19) as follows:

$$\frac{\partial L_a}{\partial \alpha_H^{Na}(k)} = \lambda^T(k)\left[\frac{\partial}{\partial \alpha_H^{Na}(k)}\left[\begin{array}{r} F(k,\alpha_H^{Na})x_{Na}(k) + G(k,\alpha_H^{Na})F_R(k)x_{Na}(k) + \\ + Q(k,\alpha_H^{Na})x_f(k) \end{array}\right]\right] = \\ = e_{\alpha_H^{Na}}(k) = 0 \qquad (7.24)$$

$$\frac{\partial L_a}{\partial \lambda(k)} = \left[\begin{array}{r} -x_{Na}(k+1) + F(k,\alpha_H^{Na})x_{Na}(k) + \\ + G(k,\alpha_H^{Na})F_R(k)x_{Na}(k) + Q(k,\alpha_H^{Na})x_f(k) \end{array}\right] = \\ = e_{\lambda}(k) = 0 \qquad (7.25)$$

The differentiation of Equation (7.24) is based on Equation (7.14) and contains two derivatives for each process stage since each equation of (7.14) contains two separation factor variables, $\alpha_{H,n}^{Na}$ and $\alpha_{H,(n+1)}^{Na}$, Equation (7.26) and Equation (7.27). It is thus necessary to differentiate the Lagrange's function in a vector-matrix format.

$$\frac{\partial L_{a\,n}}{\partial \alpha_{H,n}^{Na}(k)} = \left[\begin{array}{r} \frac{\partial f_{1,n}(k)}{\partial \alpha_{H,n}^{Na}(k)} F_L x_{(n-1)}(k) - \frac{\partial f_{1,n}(k)}{\partial \alpha_{H,n}^{Na}(k)} F_L x_n(k) + \\ + \frac{\partial}{\partial \alpha_{H,n}^{Na}(k)}\left[f_{1,n}(k) f_{3,(n+1)}(k) F_R(k) x_{(n+1)}(k)\right] - \\ - \frac{\partial}{\partial \alpha_{H,n}^{Na}(k)}\left[f_{1,n}(k) f_{2,n}(k) F_R(k) x_n(k)\right] \end{array}\right] \lambda_n(k) = \\ = e_{\alpha\,n,i}(k) = 0 \qquad (7.26)$$

$$\frac{\partial L_{a\,n}}{\partial \alpha^{Na}_{H,(n+1)}(k)} =$$

$$= \left[\begin{array}{l} \frac{\partial f_{1,n}(k)}{\partial \alpha^{Na}_{H,(n+1)}(k)} F_L x_{(n-1)}(k) - \frac{\partial f_{1,n}(k)}{\partial \alpha^{Na}_{H,(n+1)}(k)} F_L x_n(k) + \\ \qquad + \frac{\partial}{\partial \alpha^{Na}_{H,(n+1)}(k)} \left[f_{1,n}(k) f_{3,(n+1)}(k) F_R(k) x_{(n+1)}(k) \right] - \\ \qquad\qquad - \frac{\partial}{\partial \alpha^{Na}_{H,(n+1)}(k)} \left[f_{1,n}(k) f_{2,n}(k) F_R(k) x_n(k) \right] \end{array} \right] \lambda_n(k) = \tag{7.27}$$

$$= e_{\alpha(n+1),i}(k) = 0$$

where for $n = 1$, $x_{(n-1)}(k) = x_{(1-1)}(k) = x_f(k)$

The derivative of the Lagrange's function according to the vector of parameters $\alpha^{Na}_{H,n}$ is expressed by the derivatives of the matrices $F(k, \alpha^{Na}_{H,n})$, $G(k, \alpha^{Na}_{H,n})$ and $Q(k, \alpha^{Na}_{H,n})$ according to the vector of parameters, $\alpha^{Na}_{H,n}(k)$.

These derivatives can be expressed analytically following the rules for determination of derivatives of a matrix towards a vector. For this purpose the functions $f_{1,n}(k)$, $f_{2,n}(k)$ and $f_{3,n}(k)$ determined by Equations (7.13)–(7.13) can be expressed in the following manner:

$$f_{1,n}(k) = f_1(k, \alpha^{Na}_{H,n}), \; n = \overline{1, N} \tag{7.28}$$

$$f_{2,n}(k) = f_2(k, \alpha^{Na}_{H,n}), \; n = \overline{1, N} \tag{7.29}$$

$$f_{3,n}(k) = f_3(k, \alpha^{Na}_{H,n}, \alpha^{Na}_{H,n+1}), \; n = \overline{1, N} \tag{7.30}$$

Derivatives of the $G(k)$ matrix

The derivatives of these matrices are now given and simplified below,

$$\frac{\partial G(k)}{\partial \alpha^{Na}_{H,n}(k)} = \begin{bmatrix} \frac{\partial G_{11}}{\partial \alpha^{Na}_{H,1}(k)} & \frac{\partial G_{12}}{\partial \alpha^{Na}_{H,2}(k)} & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial G_{22}}{\partial \alpha^{Na}_{H,2}(k)} & \frac{\partial G_{23}}{\partial \alpha^{Na}_{H,3}(k)} & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \frac{\partial G_{55}}{\partial \alpha^{Na}_{H,5}(k)} & \frac{\partial G_{56}}{\partial \alpha^{Na}_{H,6}} \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial G_{66}}{\partial \alpha^{Na}_{H,6}(k)} \end{bmatrix} \tag{7.31}$$

The simplified equations for the derivatives of the $G(k)$ matrix are:

For the first row,

$$\frac{\partial G_{11}(k)}{\partial \alpha_{H,1}^{Na}(k)} = \frac{\partial(-f_{11} f_{21})}{\partial \alpha_{H,1}^{Na}(k)} = -f_{21} \frac{\partial f_{11}}{\partial \alpha_{H,1}^{Na}(k)} - f_{11} \frac{\partial f_{21}}{\partial \alpha_{H,1}^{Na}(k)}$$

$$\frac{\partial G_{12}(k)}{\partial \alpha_{H,2}^{Na}(k)} = \frac{\partial(f_{11} f_{32})}{\partial \alpha_{H,2}^{Na}(k)} = \frac{\partial f_{11}}{\partial \alpha_{H,2}^{Na}(k)} f_{32} + f_{11} \frac{\partial f_{32}}{\partial \alpha_{H,2}^{Na}(k)} = f_{11} \frac{\partial f_{32}}{\partial \alpha_{H,2}^{Na}(k)}$$

and all other derivatives are zero.

For the second row,

$$\frac{\partial G_{21}}{\partial \alpha_{H,1}^{Na}(k)} = 0$$

$$\frac{\partial G_{22}(k)}{\partial \alpha_{H,2}^{Na}(k)} = \frac{-\partial(f_{12} f_{22})}{\partial \alpha_{H,2}^{Na}(k)} = -\frac{\partial f_{12}}{\partial \alpha_{H,2}^{Na}(k)} f_{22} - f_{12} \frac{\partial f_{22}}{\partial \alpha_{H,2}^{Na}(k)} =$$

$$= -f_{22} \frac{\partial f_{12}}{\partial \alpha_{H,2}^{Na}(k)} - f_{12} \frac{\partial f_{22}}{\partial \alpha_{H,2}^{Na}(k)}$$

$$\frac{\partial G_{23}(k)}{\partial \alpha_{H,3}^{Na}(k)} = \frac{\partial(f_{12} f_{33})}{\partial \alpha_{H,3}^{Na}(k)} = \frac{\partial f_{12}}{\partial \alpha_{H,3}^{Na}(k)} f_{33} + f_{12} \frac{\partial f_{33}}{\partial \alpha_{H,3}^{Na}(k)} = f_{12} \frac{\partial f_{33}}{\partial \alpha_{H,3}^{Na}(k)}$$

all other derivatives are zero.

For the third row,

$$\frac{\partial G_{31}(k)}{\partial \alpha_{H,1}^{Na}(k)} = \frac{\partial G_{32}(k)}{\partial \alpha_{H,2}^{Na}(k)} = 0$$

$$\frac{\partial G_{33}(k)}{\partial \alpha_{H,3}^{Na}(k)} = \frac{-\partial(f_{13} f_{23})}{\partial \alpha_{H,3}^{Na}(k)} = -\frac{\partial f_{13}}{\partial \alpha_{H,3}^{Na}(k)} f_{23} - f_{13} \frac{\partial f_{23}}{\partial \alpha_{H,3}^{Na}} =$$

$$= -f_{23} \frac{\partial f_{13}}{\partial \alpha_{H,3}^{Na}(k)} - f_{13} \frac{\partial f_{23}}{\partial \alpha_{H,3}^{Na}(k)}$$

$$\frac{\partial G_{34}(k)}{\partial \alpha_{H,4}^{Na}(k)} = \frac{\partial(f_{13} f_{34})}{\partial \alpha_{H,4}^{Na}(k)} = \frac{\partial f_{13}}{\partial \alpha_{H,4}^{Na}(k)} f_{34} + f_{13} \frac{\partial f_{34}}{\partial \alpha_{H,4}^{Na}(k)} = f_{13} \frac{\partial f_{34}}{\partial \alpha_{H,4}^{Na}(k)}$$

$$\frac{\partial G_{35}(k)}{\partial \alpha_{H,5}^{Na}(k)} = \frac{\partial G_{36}(k)}{\partial \alpha_{H,6}^{Na}(k)} = 0$$

For the fourth row

$$\frac{\partial G_{41}(k)}{\partial \alpha_{H,1}^{Na}(k)} = \frac{\partial G_{42}(k)}{\partial \alpha_{H,2}^{Na}(k)} = \frac{\partial G_{43}(k)}{\partial \alpha_{H,3}^{Na}(k)} = 0$$

$$\frac{\partial G_{44}(k)}{\partial \alpha_{H,4}^{Na}(k)} = \frac{-\partial(f_{14} f_{24})}{\partial \alpha_{H,4}^{Na}(k)} = \frac{-\partial f_{14}}{\partial \alpha_{H,4}^{Na}(k)} f_{24} - f_{14} \frac{\partial f_{24}}{\partial \alpha_{H,4}^{Na}(k)} =$$

$$= -f_{24} \frac{\partial f_{14}}{\partial \alpha_{H,4}^{Na}(k)} - f_{14} \frac{\partial f_{24}}{\partial \alpha_{H,4}^{Na}(k)}$$

$$\frac{\partial G_{45}(k)}{\partial \alpha_{H,5}^{Na}(k)} = \frac{\partial(f_{14} f_{35})}{\partial \alpha_{H,5}^{Na}(k)} = \frac{\partial f_{14}}{\partial \alpha_{H,5}^{Na}(k)} f_{35} + f_{14} \frac{\partial f_{35}}{\partial \alpha_{H,5}^{Na}(k)} = f_{14} \frac{\partial f_{35}}{\partial \alpha_{H,5}^{Ns}(k)}; \frac{\partial G_{46}(k)}{\partial \alpha_{H,5}^{Na}(k)} = 0$$

The fifth row

$$\frac{\partial G_{51}(k)}{\partial \alpha_{H,1}^{Na}(k)} = \frac{\partial G_{52}(k)}{\partial \alpha_{H,2}^{Na}(k)} = \frac{\partial G_{53}(k)}{\partial \alpha_{H,3}^{Na}(k)} = \frac{\partial G_{54}(k)}{\partial \alpha_{H,4}^{Na}(k)} = 0$$

$$\frac{\partial G_{55}(k)}{\partial \alpha_{H,5}^{Na}(k)} = \frac{-\partial(f_{15} f_{25})}{\partial \alpha_{H,5}^{Na}(k)} = -\frac{\partial f_{15}}{\partial \alpha_{H,5}^{Na}(k)} f_{25} - f_{15} \frac{\partial f_{25}}{\partial \alpha_{H,5}^{Na}(k)}$$

$$\frac{\partial G_{56}(k)}{\partial \alpha_{H,6}^{Na}(k)} = \frac{\partial(f_{15} f_{36})}{\partial \alpha_{H,6}^{Na}(k)} = \frac{\partial f_{15}}{\partial \alpha_{H,6}^{Na}(k)} f_{36} + f_{15} \frac{\partial f_{36}}{\partial \alpha_{H,6}^{Na}(k)} = f_{15} \frac{\partial f_{36}}{\partial \alpha_{H,6}^{Na}(k)}$$

The sixth row

$$\frac{\partial G_{61}(k)}{\partial \alpha_{H,1}^{Na}(k)} = \frac{\partial G_{62}(k)}{\partial \alpha_{H,2}^{Na}(k)} = \frac{\partial G_{63}(k)}{\partial \alpha_{H,3}^{Na}(k)} = \frac{\partial G_{64}(k)}{\partial \alpha_{H,4}^{Na}(k)} = \frac{\partial G_{65}(k)}{\partial \alpha_{H,5}^{Na}(k)} = 0$$

$$\frac{\partial G_{66}(k)}{\partial \alpha_{H,6}^{Na}(k)} = \frac{-\partial(f_{16} f_{26})}{\partial \alpha_{H,6}^{Na}(k)} = -\frac{\partial f_{16}}{\partial \alpha_{H,6}^{Na}(k)} f_{26} - f_{16} \frac{\partial f_{26}}{\partial \alpha_{H,6}^{Na}(k)} = -f_{26} \frac{\partial f_{16}}{\partial \alpha_{H,6}^{Na}(k)} - f_{16} \frac{\partial f_{26}}{\partial \alpha_{H,6}^{Na}(k)}$$

The matrix $\frac{\partial G(k)}{\partial \alpha_{H,n}^{Na}(k)}$ is given by Equation (7.32):

$$\frac{\partial G(k)}{\partial \alpha_H^{Na}(k)} = \begin{bmatrix} -g_{211}-g_{121} & g_{132} & 0 & 0 & 0 & 0 \\ 0 & -g_{212}-g_{122} & g_{133} & 0 & 0 & 0 \\ 0 & 0 & -g_{213}-g_{123} & g_{134} & 0 & 0 \\ 0 & 0 & 0 & -g_{214}-g_{124} & g_{135} & 0 \\ 0 & 0 & 0 & 0 & -g_{215}-g_{125} & g_{136} \\ 0 & 0 & 0 & 0 & 0 & -g_{216}-g_{126} \end{bmatrix} \tag{7.32}$$

where

$g_{211} = f_{21}\frac{\partial f_{11}}{\partial \alpha_{H,1}^{Na}(k)}$; $g_{121} = f_{11}\frac{\partial f_{21}}{\partial \alpha_{H,1}^{Na}(k)}$; $g_{132} = f_{11}\frac{\partial f_{32}}{\partial \alpha_{H,2}^{Na}(k)}$; $g_{212} = f_{22}\frac{\partial f_{12}}{\partial \alpha_{H,2}^{Na}(k)}$; $g_{122} = f_{11}\frac{\partial f_{22}}{\partial \alpha_{H,2}^{Na}(k)}$; $g_{133} = f_{12}\frac{\partial f_{33}}{\partial \alpha_{H,3}^{Na}(k)}$
$g_{213} = f_{23}\frac{\partial f_{13}}{\partial \alpha_{H,3}^{Na}(k)}$; $g_{123} = f_{13}\frac{\partial f_{23}}{\partial \alpha_{H,3}^{Na}(k)}$; $g_{134} = f_{13}\frac{\partial f_{34}}{\partial \alpha_{H,4}^{Na}}$; $g_{214} = f_{24}\frac{\partial f_{14}}{\partial \alpha_{H,4}^{Na}(k)}$; $g_{124} = f_{14}\frac{\partial f_{24}}{\partial \alpha_{H,4}^{Na}(k)}$; $g_{135} = f_{14}\frac{\partial f_{35}}{\partial \alpha_{H,4}^{Na}(k)}$
$g_{215} = f_{25}\frac{\partial f_{15}}{\partial \alpha_{H,5}^{Na}(k)}$; $g_{125} = f_{15}\frac{\partial f_{25}}{\partial \alpha_{H,5}^{Na}(k)}$; $g_{136} = f_{15}\frac{\partial f_{36}}{\partial \alpha_{H,5}^{Na}(k)}$; $g_{216} = f_{26}\frac{\partial f_{16}}{\partial \alpha_{H6}^{Na}(k)}$; $g_{126} = f_{16}\frac{\partial f_{26}}{\partial \alpha_{H,6}^{Na}(k)}$.

The terms of the $\frac{\partial G(k)}{\partial \alpha_{H,n}^{Na}(k)}$ matrix may be generalized as follows:

$$-g_{21n} - g_{12n} = -f_{2,n}\frac{\partial f_{1,n}}{\partial \alpha_{H,n}^{Na}(k)} - f_{1,n}\frac{\partial f_{2,n}}{\partial \alpha_{H,n}^{Na}(k)},\ n = \overline{1,N} \tag{7.33}$$

$$g_{13(n+1)} = f_{1,n}\frac{\partial f_{3,n+1}}{\partial \alpha_{H,n+1}^{Na}(k)},\ n = \overline{1,N} \tag{7.34}$$

For the general equations in the derivative of the Lagrange equation with respect to the separation factor for n and $n+1$, the following expressions are generated. The elements of the derivatives of the matrices are expressed following Equation (7.15) using the argument that,

if $f(t) = \frac{f_1(t)}{f_2(t)}$ then $\frac{df(t)}{dt} = \frac{f_2(t)\frac{df_1(t)}{dt} - f_1(t)\frac{df_2(t)}{dt}}{[f_2(t)]^2}$:

$$\frac{\partial f_{1,n}(k)}{\partial \alpha_{H,n}^{Na}(k)} = \frac{\partial}{\partial \alpha_{H,n}^{Na}}\left(\frac{\Delta t\left[\alpha_{H,n}^{Na}(k) + x_{Na,n}(k)(1-\alpha_{H,n}^{Na}(k))\right]^2}{H\left[\alpha_{H,n}^{Na}(k) + x_{Na,n}(k)(1-\alpha_{H,n}^{Na}(k))\right]^2 + h\alpha_{H,n}^{Na}(k)}\right),$$

$$n = \overline{1,N} \tag{7.35}$$

$$\frac{\partial f_{2,n}(k)}{\partial \alpha_{H,n}^{Na}(k)} = \frac{\partial}{\partial \alpha_{H,n}^{Na}(k)}\left(\frac{\Delta t}{[\alpha_{H,n}^{Na}(k) + x_{Na,n}(k)(1-\alpha_{H,n}^{Na}(k))]}\right) =$$

$$= \frac{\Delta t(x_n(k)-1)}{\begin{bmatrix}(\alpha_{H,n}^{Na}(k))^2 + 2(\alpha_{H,n}^{Na}(k))x_{Na,n} - 2(\alpha_{Na,n}^{Na}(k))^2 x_{Na,n}(k) + x_{Na,n}^2(k) - \\ \qquad - 2(\alpha_{H,n}^{Na}(k))x_{Na,n}^2(k) + (\alpha_{H,n}^{Na}(k))^2 x_{Na,n}^2(k)\end{bmatrix}},$$

$$n = \overline{1,N} \tag{7.36}$$

$f_{3,n+1}(k)$ is a function of $\alpha_{H,(n+1)}^{Na}(k)$

$$\frac{\partial f_{3,(n+1)}(k)}{\partial \alpha_{H,(n+1)}^{Na}(k)} = \frac{\partial\left(\frac{\Delta t}{\alpha_{H,n}^{Na}(k) + x_{Na,(n+1)}(k)(1-\alpha_{H,(n+1)}^{Na}(k))}\right)}{\partial \alpha_{H,(n+1)}^{Na}(k)} =$$

$$= \frac{-\Delta t(-x_{Na,(n+1)}(k)}{\left[\alpha_{H,n}^{Na}(k) + x_{Na,(n+1)}(k)(1-\alpha_{H,(n+1)}^{Na}(k))\right]^2} =$$

$$= \frac{\Delta t(x_{Na,(n+1)}(k))}{\begin{bmatrix}(\alpha_{H,n}^{Na}(k))^2 + 2\alpha_{H,n}^{Na}(k)x_{Na,n+1}(k) - 2(\alpha_{H,n}^{Na}(k)x_{n+1}(k)\alpha_{H,n+1}^{Na}(k)) \\ \qquad + (x_{Na,n+1}(k))^2 - 2(x_{Na,n+1}(k))^2\alpha_{H,n+1}^{Na}(k) + (x_{Na,n+1}(k))^2(\alpha_{H,n+1}^{Na}(k))^2\end{bmatrix}} \tag{7.37}$$

Derivatives of the $F(k)$ matrix

$$\frac{\partial F(k)}{\partial \alpha_H^{Na}(k)} =$$

$$= \begin{bmatrix} \frac{-\partial f_{11}}{\partial \alpha_{H,1}^{Na}(k)} & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial f_{12}}{\partial \alpha_{H,1}^{Na}(k)} & \frac{-\partial f_{12}}{\partial \alpha_{H,2}^{Na}(k)} & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial f_{13}}{\partial \alpha_{H,2}^{Na}(k)} & \frac{-\partial f_{13}}{\partial \alpha_{H,3}^{Na}(k)} & 0 & 0 & 0 \\ 0 & 0 & \frac{\partial f_{14}}{\partial \alpha_{H,3}^{Na}(k)} & \frac{-\partial f_{14}}{\partial \alpha_{H,4}^{Na}(k)} & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial f_{15}}{\partial \alpha_{H,4}^{Na}(k)} & \frac{-\partial f_{15}}{\partial \alpha_{H,5}^{Na}} & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial f_{16}}{\partial \alpha_{H,5}^{Na}(k)} & \frac{-\partial f_{16}}{\partial \alpha_{H,6}^{Na}(k)} \end{bmatrix} \times F_L \qquad (7.38)$$

The diagonal elements of the matrix $F(k)$ are represented by the expressions,

$$F_{n,n} = 1 - f_{1,n} F_L \,, \; n = \overline{1, N} \qquad (7.39)$$

The corresponding derivatives are given by

$$\frac{\partial F_{n,n}}{\partial \alpha_{H,n}^{Na}(k)} = \frac{\partial \left(1 - f_{1,n} F_L\right)}{\partial \alpha_{H,n}^{Na}(k)} = \frac{\partial f_{1,n}}{\partial \alpha_{H,n}^{Na}(k)} F_L \qquad (7.40)$$

For separate elements of the matrix, considering them row by row; the derivatives for the off diagonal elements are all zeros:

1) First row,

$$\frac{\partial F_{1,1}(k)}{\partial \alpha_{H,1}^{Na}(k)} = \frac{-\partial f_{1,1}(k)}{\partial \alpha_{H,1}^{Na}(k)} F_L$$

2) Second row

$$\frac{\partial f_{1,2}(k)}{\partial \alpha_{H,1}^{Na}(k)} F_L = 0 \,; \; \frac{\partial F_{2,2}(k)}{\partial \alpha_{H,2}^{Na}} = \frac{-\partial f_{1,2}(k)}{\partial \alpha_{H,2}^{Na}(k)} F_L$$

3) Third row

$$\frac{\partial f_{1,3}(k)}{\partial \alpha_{H,2}^{Na}(k)} F_L = 0 \,; \; \frac{\partial F_{3,3}(k)}{\partial \alpha_{H,3}^{Na}(k)} = \frac{-\partial f_{1,3}(k)}{\partial \alpha_{H,3}^{Na}(k)} F_L$$

4) Fourth row

$$\frac{\partial f_{1,4}(k)}{\partial \alpha_{H,3}^{Na}(k)} F_L = 0 \,; \; \frac{\partial F_{4,4}(k)}{\partial \alpha_{H,4}^{Na}(k)} = \frac{-\partial f_{1,4}(k)}{\partial \alpha_{H,4}^{Na}(k)} F_L$$

5) Fifth row

$$\frac{\partial f_{1,5}(k)}{\partial \alpha_{H,5}^{Na}(k)} F_L = 0 \, ; \; \frac{\partial F_{5,5}(k)}{\partial \alpha_{H,5}^{Na}(k)} = \frac{-\partial f_{1,5}(k)}{\partial \alpha_{H,5}^{Na}(k)} F_L$$

6) Sixth row

$$\frac{\partial f_{1,6}(k)}{\partial \alpha_{H,5}^{Na}(k)} F_L = 0 \, ; \; \frac{\partial F_{6,6}(k)}{\partial \alpha_{H,6}^{Na}(k)} = \frac{-\partial f_{1,6}(k)}{\partial \alpha_{H,6}^{Na}(k)} F_L$$

The Lagrange function derivative in Equation (7.24) can be presented using matrices as in Equation (7.16).

$$\frac{\partial F(k)x(k)}{\partial \alpha_H^{Na}(k)} = \begin{bmatrix} \frac{\partial F_{11}(k)}{\partial \alpha_{H,1}^{Na}(k)} & \frac{\partial F_{12}(k)}{\partial \alpha_{H,2}^{Na}(k)} & \cdots & \frac{\partial F_{1N}(k)}{\partial \alpha_{H,N}^{Na}(k)} \\ \frac{\partial F_{21}(k)}{\partial \alpha_{H,1}^{Na}(k)} & \frac{\partial F_{22}(k)}{\partial \alpha_{H,2}^{Na}(k)} & \cdots & \frac{\partial F_{2N}(k)}{\partial \alpha_{H,N}^{Na}(k)} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial F_{N1}(k)}{\partial \alpha_{H,1}^{Na}(k)} & \frac{\partial F_{N2}(k)}{\partial \alpha_{H,2}^{Na}(k)} & \cdots & \frac{\partial F_{NN}(k)}{\partial \alpha_{H,N}^{Na}(k)} \end{bmatrix} x_{Na}(k) \tag{7.41}$$

Then the derivative of Equation (7.44) becomes

$$\frac{\partial F(k,\alpha_H^{Na})}{\partial \alpha_H^{Na}(k)} = \begin{bmatrix} \frac{\partial F_{1,1}(k)}{\partial \alpha_{H,1}^{Na}(k)} & 0 & 0 & \dots & 0 \\ 0 & \frac{\partial F_{2,2}(k)}{\partial \alpha_{H,2}^{Na}(k)} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \frac{\partial F_{N,N}(k)}{\partial \alpha_{H,N}^{Na}(k)} \end{bmatrix} \tag{7.42}$$

where every derivative is calculated on the basis of $\frac{\partial F_{n,n}(k)}{\partial \alpha_{H,n}^{Na}(k)}$, $n = \overline{1,N}$.

Calculation of the $Q(k)$ matrix derivatives,

$$\frac{\partial Q(k,\alpha_H^{Na})}{\partial \alpha_H^{Na}(k)} = \left[\partial F1{,}1(k)/\partial \alpha_{H,1}^{Na}(k) \quad 0 \quad \dots \quad 0\right]^T = \\ = \left[-\partial f1{,}1(k)/\partial \alpha_{H,1}^{Na}(k) \quad 0 \quad \dots \quad 0\right]^T \tag{7.43}$$

The Lagrange derivative is given by

$$\frac{\partial L_a(k)}{\partial \alpha_H^{Na}(k)} = \left[\frac{\partial F(k,\alpha_{H,n}^{Na})}{\partial \alpha_{H,n}^{Na}(k)} + \frac{\partial G(k,\alpha_{H,n}^{Na})}{\partial \alpha_{H,n}^{Na}(k)} F_R(k) x_{Na}(k) + \frac{\partial Q(k,\alpha_{H,n}^{Na})}{\partial \alpha_{H,n}^{Na}(k)} x_f(k)\right] \lambda(k) =$$

$$
= \left\{ \begin{array}{l}
\begin{bmatrix}
\frac{\partial F_1(k,\alpha_{H,1}^{Na})}{\partial \alpha_{H,1}^{Na}(k)} & 0 & \ldots & 0 \\
0 & \frac{\partial F_2(k,\alpha_{H,2}^{Na})}{\partial \alpha_{H,2}^{Na}(k)} & \ldots & 0 \\
\ldots & \ldots & \ldots & \\
0 & 0 & \ldots & \frac{\partial F_N(k,\alpha_{H,N}^{Na})}{\partial \alpha_{H,N}^{Na}(k)}
\end{bmatrix} x_{Na}(k) + \\
+ \begin{bmatrix}
-g_{211}-g_{121} & g_{132} & 0 & 0 & 0 & 0 \\
0 & -g_{212}-g_{122} & g_{133} & 0 & 0 & 0 \\
0 & 0 & -g_{213}-g_{123} & g_{134} & 0 & 0 \\
0 & 0 & 0 & -g_{214}-g_{124} & g_{135} & 0 \\
0 & 0 & 0 & 0 & -g_{215}-g_{125} & g_{136} \\
0 & 0 & 0 & 0 & 0 & -g_{216}-g_{126}
\end{bmatrix} F_R(k) x_{Na}(k) + \\
\qquad + \left[-\partial F_1(k) / \partial \alpha_{H,1}^{Na}(k) \quad 0 \quad \ldots \quad 0 \right]^T x_f(k)
\end{array} \right\} \tag{7.44}
$$

The first condition for optimality according to the Lagrange's variable is given by Equation (7.25).

7.4.2. Algorithm for the parameter estimation of the separation factor for the nonlinear model using the Lagrange method

1) Set all initial process model values,
2) Read all measured values, $x_{Na}(k) \in R^{6\times K}$,
3) Set initial values of $\lambda(k) \in R^{6\times K}$,
4) Set initial values of $\alpha_H^{Na} \in R^{6\times K}$,
5) Set $F_R(k) \in R^{1\times K}$,
6) Set gradient method parameters initial conditions,
7) Calculation of $\frac{\partial L_a(k)}{\partial \lambda(k)}$, Equation (7.25) using values from steps 1)– 3) above:

   7.1) Calculate the function $f_{1,n}(k)$, Equation (7.11), $f_{1,n}(k) \in R^{6\times K}$,

   7.2) Calculate the function $f_{2,n}(k)$, Equation (7.12), $f_{2,n}(k) \in R^{6\times K}$,

   7.3) Calculation of the function $f_{3,(n+1)}(k)$, Equation (7.13), $f_{3,(n+1)}(k) \in R^{6\times K}$,

   7.4) Form the matrix $F(k)$, Equation (7.16),

   7.5) Forming the matrix $G(k)$, Equation (7.16),

   7.6) Forming the matrix $Q(k)$, Equation 7.16),

   7.7) Calculate $\frac{\partial L_a(k)}{\partial \lambda(k)}$, Equation (7.25) and the error $e_\lambda(k)$.

8) Check the condition for optimality (for stopping calculations),

   8.1) if $e_\lambda(k) \le \varepsilon_\lambda$, stop the calculation, ε_λ is a small positive number used for stopping the calculation, and

9) Calculate the new (improved) value of λ

   $\lambda^{(j+1)}(k) = \lambda^j(k) + \Delta_\lambda e_\lambda(k)$.

10) Calculation of the derivative of the Lagrange $\frac{\partial L_a(k)}{\partial \alpha_H^{Na}(k)}$, Equation (7.24), using the improved value of the Lagrange multiplier $\lambda^{(j+1)}$ and the guessed values of the separation factor, $\alpha_H^{Na}(k)$,

    10.1) Calculate $\frac{\partial F_{n,n}(k)}{\partial \alpha_{H,n}^{Na}(k)}$ for $n,n = 1,1;\ 2,2;\ldots N,N$, $k = \overline{1,K}$,

    10.2) Form the matrix,

$$\frac{\partial F(k)}{\partial \alpha_H^{Na}(k)} = diag\left\{ \frac{\partial F_{1,1}(k)}{\partial \alpha_{H,1}^{Na}(k)}, \quad \frac{\partial F_{2,2}(k)}{\partial \alpha_{H,2}^{Na}(k)}, \quad \dots \quad \frac{\partial F_{N,N}(k)}{\partial \alpha_{H,N}^{Na}(k)} \right\}, \ k = \overline{1,K} .$$

10.3) Form the $Q(k)$ matrix

$$Q(k) = \left[-\partial F_{1,1}(k) / \partial \alpha_{H,1}^{Na}(k) \quad 0 \quad \dots \quad 0 \ \right]^T, \ k = \overline{1,K},$$

10.4) Calculate the derivatives of the matrix $\frac{\partial G(k)}{\partial \alpha_H^{Na}(k)}$,

10.5) Form the matrix $\frac{\partial G(k)}{\partial \alpha_H^{Na}(k)} \in R^{6\times 6}$, $k = \overline{1,K}$,

10.6) Calculate $\frac{\partial L_a(k)}{\partial \alpha_H^{Na}(k)} = e_\alpha(k)$, $k = \overline{1,K}$.

11) Check the condition for optimality,

11.1) if $e_\alpha(k) < \varepsilon_\alpha$, then $\alpha_H^{Na(j+1)}(k) = \alpha_H^{Na(j)}(k)$,

11.2) if not, calculate the new value of $\alpha_H^{Na}(k)$.

12) Calculate the new value of $\alpha_H^{Na}(k)$

$$\alpha_H^{Na(j+1)}(k) = \alpha_H^{Na(j)} - \Delta_\alpha e_\alpha(k), \ k = \overline{1,K} .$$

13) Check if maximum number of iterations (M) has been reached

13.1) if $j = M$, then stop the calculation,

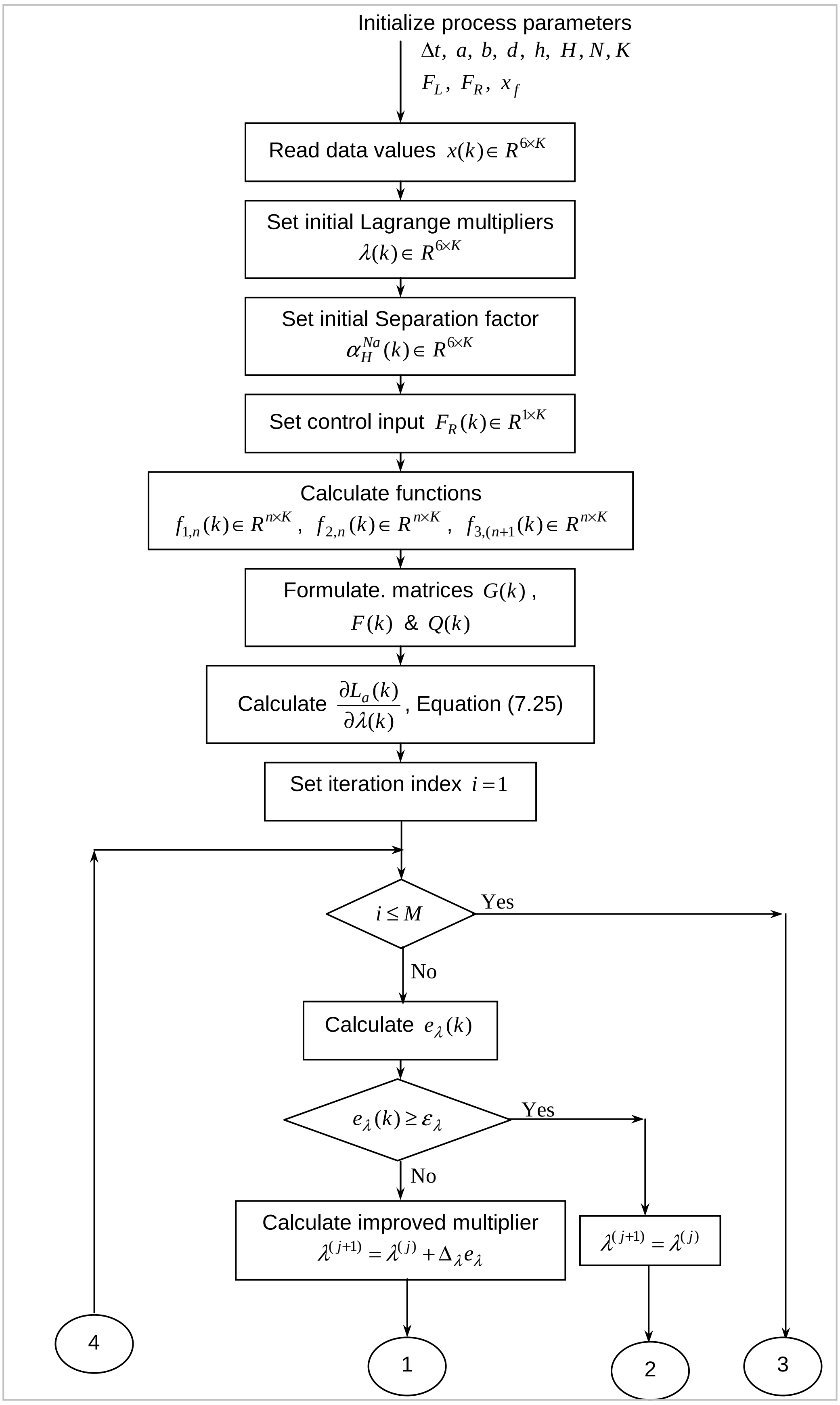
13.1) if not, continue calculation, $j = j + 1$,

14) Plot the calculated parameter values received, $\alpha_H^{Na}(k) \in R^{6\times K}$.

7.4.3. Flowchart of the nonlinear parameter estimation using Lagrange method

Figure 7.9 presents the flowchart of the developed Lagrange method for the nonlinear model.

Due to the complex nature of the developed Lagrange method; this method would dictate further complex program development. As it has been seen from the direct method, there are a number of issues that need further interrogation and that has meant that this method could not be investigated further beyond the development stage. It is suggested this part of the work be considered as a possible future study where indepth and experimental interrogation would be considered with the intention of producing better results or analysis.



Initialize process parameters
$\Delta t,\ a,\ b,\ d,\ h,\ H, N, K$
$F_L,\ F_R,\ x_f$
Read data values $x(k) \in R^{6\times K}$
Set initial Lagrange multipliers $\lambda(k) \in R^{6\times K}$
Set initial Separation factor $\alpha_H^{Na}(k) \in R^{6\times K}$
Set control input $F_R(k) \in R^{1\times K}$
Calculate functions $f_{1,n}(k) \in R^{n\times K}$, $f_{2,n}(k) \in R^{n\times K}$, $f_{3,(n+1)}(k) \in R^{n\times K}$
Formulate. matrices $G(k)$, $F(k)$ & $Q(k)$
Calculate $\frac{\partial L_a(k)}{\partial \lambda(k)}$, Equation (7.25)
Set iteration index $i = 1$
$i \le M$
Yes
No
Calculate $e_\lambda(k)$
$e_\lambda(k) \ge \varepsilon_\lambda$
Yes
No
Calculate improved multiplier $\lambda^{(j+1)} = \lambda^{(j)} + \Delta_\lambda e_\lambda$
$\lambda^{(j+1)} = \lambda^{(j)}$
4
1
2
3

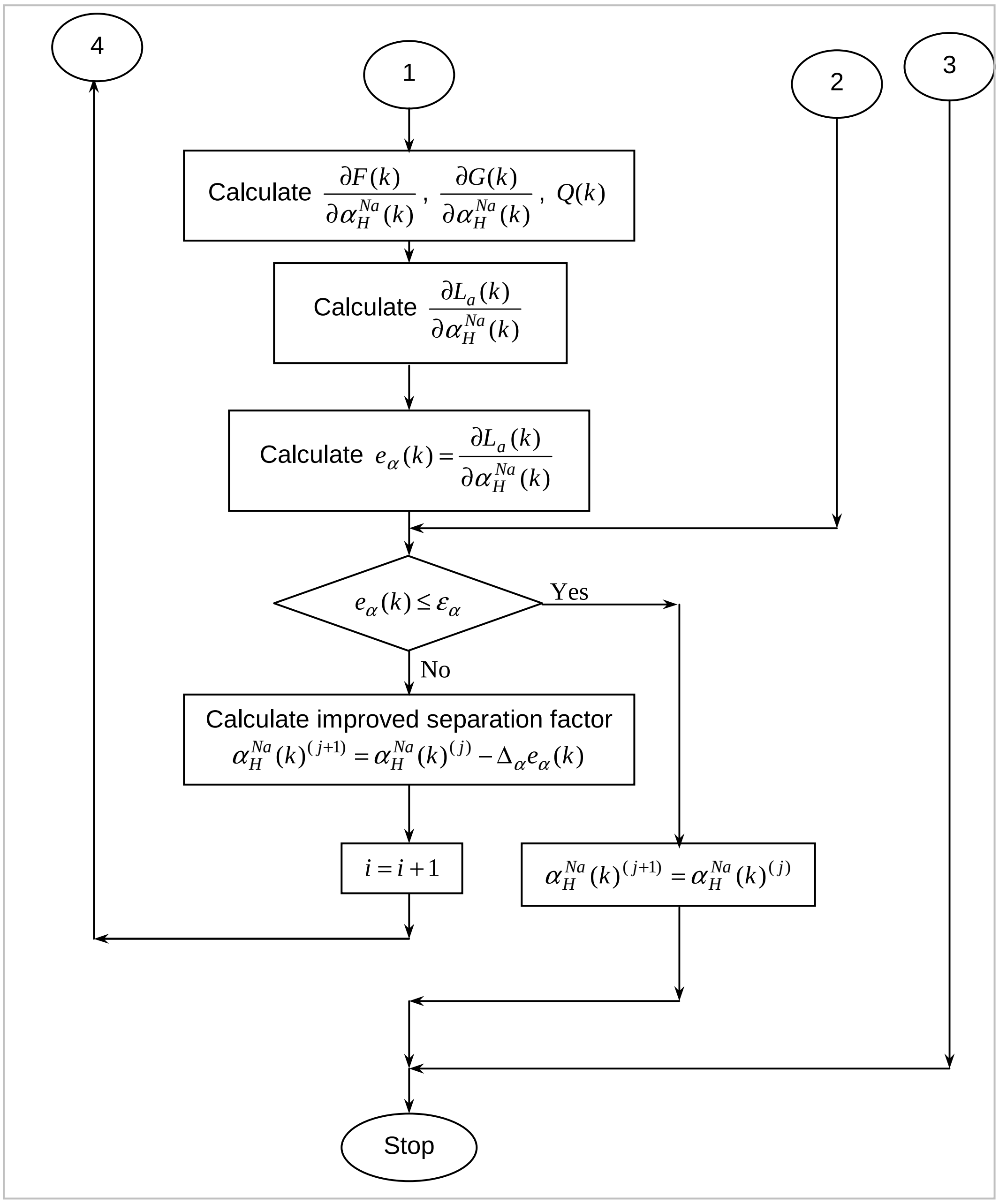

4
1
2
3
Calculate $\frac{\partial F(k)}{\partial \alpha_H^{Na}(k)}$, $\frac{\partial G(k)}{\partial \alpha_H^{Na}(k)}$, $Q(k)$
Calculate $\frac{\partial L_a(k)}{\partial \alpha_H^{Na}(k)}$
Calculate $e_\alpha(k) = \frac{\partial L_a(k)}{\partial \alpha_H^{Na}(k)}$
$e_\alpha(k) \le \varepsilon_\alpha$
Yes
No
Calculate improved separation factor $\alpha_H^{Na}(k)^{(j+1)} = \alpha_H^{Na}(k)^{(j)} - \Delta_\alpha e_\alpha(k)$
$i = i + 1$
$\alpha_H^{Na}(k)^{(j+1)} = \alpha_H^{Na}(k)^{(j)}$
Stop

Figure 7.9. The flowchart of the nonlinear parameter estimation using the Lagrange method

7.5. Conclusion

This chapter presented the approach to the solution of the parameter estimation problem for the model of the continuous countercurrent ion exchange process (CCIX) nonlinear towards its parameters. The nonlinear model is developed based on the separation factor between the two core exchanging ions of the CCIX process, Na^+ and H^+. The separation factor defines the behaviour of the ion exchange process at equilibrium. Once the model has been developed, unknown parameters need to be identified, and two methods of solution are proposed to solve the nonlinear model parameter estimation problem, 1) the *direct method that uses a* MATLAB *software program optimization function* called $fsolve()$, and 2) *the Lagrange optimization method*.

Though the experiments of the nonlinear model produced some relevant results, there were a number of issues that may need interrogation in the future, more especially the usage of the *fsolve*() function. The usage of this function is further complicated by the fact that the design of ion exchange process has stages that have a direct influence on one another. Every other stage is influenced by the previous stage and these stages are interdependent via the state vector. It became impossible to run the experiments considering only one particular stage at a time. It also seems that the *fsolve*() function works best with scalar values and since the model being used is vectors/matrix based, the *fsolve*() function could not produce expected result. This was tested with scalar values and calculations were quickly produced, but in introducing vectors, the computation period was extremely long and in most cases, the optimization procedure had to be terminated.

Conclusion from these observations is that when using *fsolve*() function, one needs to take note of the model structure and decide if it is worth using or not. *fsolve*() function may not have been the best option for the nonlinear model developed here due to its complexity.

Once again, due to the complex nature of the nonlinear model for the CCIX plant developed, the Lagrange method source code was not written. It may be considered in the furure developments of the project.

The next chapter, Chapter 8 deals with the state estimation problem solved using a bilinear observer design. Once the unknown model parameters have been estimated, it is possible to use them in the developed model to determine unknown states of the model. A Luenberger type observer is designed and solved using pole placement method.

CHAPTER EIGHT

METHOD FOR DESIGN OF A BILINEAR OBSERVER

CHAPTER EIGHT

8. METHOD FOR DESIGN OF A BILINEAR OBSERVER

8.1. Introduction

This chapter deals with the solution of the state estimation problem developed for the continuous countercurrent ion exchange (CCIX) process. The states of the plant were not measurable due to the plant design, except for the last stage of the column. The liquid inflow to the column at the first stage is considered as the system disturbance, and the outflow from the last stage is considered as the system output variable. The state values are needed for the state space control of the process. It is therefore necessary to estimate the unmeasured states. The state estimation problem is solved by two different approaches; the *observer design* (with and without the inflow disturbance) and the *Kalman filter*. This chapter, Chapter 8 deals with the observer design solution and the Kalman filter solution is covered in the next chapter, Chapter 9. The process model used in implementing these solutions is that of a bilinear system as generated in Chapter 5.

The *observer design* method used considers two possible solutions, the solution where the disturbance is *not included* in the model (the system is assumed to have no external disturbance); and secondly, the case where the disturbance is assumed constant for a long period of time. Another focal point of the method of solution is the behaviour of the input signal based on a given input trajectory of the system. In the first case, the input signal varies with time (this is more proper for a closed loop system that includes an observer design as part of the solution for designing a controller of the system); and the second part considers the input signal as constant over the given input signal trajectory.

The chapter covers the following topics: Section 8.2 is the procedure for the design of the observer to estimate the unknown system states; the observer design for the CCIX bilinear model with and without the system disturbance in Section 8.3; determination of the observer matrix based on constant input signal; the application of the pole placement method for design of the desired observer matrix and the accompanying procedure; the usage of determinants as means of solving the pole placement problem in Section 8.6; Section 8.7, MATLAB/SIMULINK software programs and the accompanying algorithms as part of the solution of the pole placement problem to determine the observer gain matrix; and finally the results from the MATLAB and SIMULINK programs, Section 8.7. The chapter ends with concluding remarks.

8.2. Procedure for design of the observer for solution of the state estimation problem

The main aim of the design of the state observer is to minimize the error between the actual process states $x(t)$ and the estimated states $\hat{x}(t)$. In designing the observer, generally, two conditions must be met:

1) *minimizing the error* between the actual state variables and the estimated state variables, i.e.,

$$e(t) = x(t) - \hat{x}(t) \rightarrow \min\ ,\ t = 0, 1, 2, ... \qquad (8.1)$$

2) *the error rate of change* (derivative of the error) must stay zero under every input and state variables, in some areas determined by the physical limitations of the process, Equation (8.3),

$$\frac{de(t)}{dt} = 0\ ,\ t = 0, 1, 2, ... \qquad (8.2).$$

The first condition can be determined for various initial conditions of the state variables for the state and for the estimated state. Two cases are considered:

1) if the estimated state $\hat{x}(t_0)$ at the current computation is such that it coincides with the state vector $x(t_0)$ at some initial moment in time t_0, then this coincidence is necessary to be kept for the rest of moments in time that follow;

$$\left| \begin{array}{ll} if\ \ at\ \ \ t = t_0, & \hat{x}(t_0) = x(t_0) \\ then\ \text{for all}\ t > t_0, & \hat{x}(t) = x(t) \end{array} \right. \ ,\ t = 1, 2, 3, ... \qquad (8.3)$$

2) if the estimated state $\hat{x}(t_0)$ does not coincide with the initial state $x(t_0)$, i.e., $\hat{x}(t_0) \neq x(t_0)$, then, the error $e(t)$ has to tend to zero as $t \rightarrow \infty$,

$$\lim_{t\rightarrow\infty} e(t) = \lim_{t\rightarrow\infty} [x(t) - \hat{x}(t)] = 0 \qquad (8.4)$$

In the considered case of the bilinear process model, unlike in the linear case, the error dynamics are not totally independent of the input and the state due to the bilinear term (Khalil, et al., 2015; Komatsu & Takata, 2009; Williamson, 1977). The procedure for design of the observer matrices is developed through fulfillment of the conditions (Equation 8.1) and (Equation 8.4). The procedure for the design of the observer is divided into two cases, 1) the case where there is no disturbance signal $x_f(t) = 0$, and 2) the case where the disturbance signal is assumed to be constant for long periods of time, as this will be the most practical case in the real plant operation (Ahmed–Ali, et al., 2017; Trinh, et al., 2006; Misawa & Hedrick, 1989).

The system disturbance is described by $x_f(t) = x_{n-1}(t)$, for $n = 1$, that is, the disturbance signal is the input concentration $x_{in}(t)$ at the first stage of the column.

8.3. Observer design for the continuous countercurrent ion exchange (CCIX) process bilinear model without a disturbance

8.3.1. Derivation of the observer equation

Consider the case where the model given in Chapter 5, Data and Model Reformulation, Equation (5.8), if the model equation has no disturbance vector, i.e., $x_f(t) = 0$, the model equation is then represented by

$$\dot{x}(t) = Ax(t) + B_1 x(t) F_R(t) + BF_R(t),\ x(0) = x_0 = 0 \tag{8.5}$$

$$z(t) = Cx(t)$$

where

$A \in R^{n\times n}$ – is system state matrix,

$B \in R^{m\times n}$ – is the system state matrix,

$B_1 \in R^{n\times n}$ – is the system matrix,

$C \in R^{l\times n}$ – is the output matrix,

$x(t) \in R^{n\times l}$ – is the state vector,

$x_f(t) \in R^{l}$ – is the disturbance vector,

$F_R(t) \in R^{l}$ – is the control input signal,

$z(t) \in R^{l}$ – is the output vector, and

$x(0) = x_0$ – is the initial state of the system.

The observer model for the plant model Equation (8.5) is considered to be of the form

$$\dot{\hat{x}}(t) = L_1\hat{x}(t) + L_2\hat{x}(t)F_R(t) + L_3F_R(t) + Lz(t),\ \hat{x}(0) = x_0 \tag{8.6}$$

$$z(t) = Cx(t)$$

where

$\hat{x}(t) \in R^{N\times 1}$ – is the estimated state vector,

$\hat{x}(0) \in R^{N\times 1}$ – is the initial estimate state vector,

$L_1 \in R^{N\times N}$ – is the state matrix of the observer,

$L_2 \in R^{N \times N}$ – is the bilinear term matrix of the observer,

$L_3 \in R^{N \times 1}$ – is the control input matrix of the observer, and

$L \in R^{1 \times N}$ – is the output (the observation) matrix.

In this first case, the process of the observer design involves determination of the matrices, L_1, L_2, L_3, and L in such a way that the two error requirements, Equations (8.2)–(8.4) representing the error between the state and the estimated state are met. The derivative of the error equation is determined by,

$$\dot{e}_x(t) = \left[\dot{x}(t) - \dot{\hat{x}}(t)\right] = 0 \tag{8.7}$$

$$\begin{aligned} \dot{e}_x(t) = \left[Ax(t) + B_1 x(t) F_R(t) + BF_R(t)\right] - \\ - \left[L_1 \hat{x}(t) + L_2 \hat{x}(t) F_R(t) + L_3 F_R(t) + Lz(t)\right] = 0 \end{aligned} \tag{8.8}$$

$$\Rightarrow \dot{e}_x x(t) = \begin{bmatrix} (A - LC)x(t) - L_1 \hat{x}(t) + B_1 x(t) F_R(t) - L_2 \hat{x}(t) F_R(t) + \\ + (B - L_3) F_R(t) \end{bmatrix} =$$

$$= \begin{bmatrix} (A - LC)x(t) - L_1(x(t) - e(t)) + B_1 x(t) F_R(t) - L_2(x(t) - \\ - e(t)) F_R(t) + (B - L_3) F_R(t) \end{bmatrix} =$$

$$= \begin{bmatrix} (A - LC - L_1)x(t) + (B_1 - L_2)x(t) F_R(t) - (L_1 + L_2 F_R(t)) e(t) + \\ + (B - L_3) F_R(t) \end{bmatrix} = 0 \tag{8.9}$$

Conditions (8.2) and (8.3) are applied to Equation (8.9). The observer will produce the exact state estimate if $e(t) = 0$ and $\dot{e}_x(t) = 0$. In order for $\dot{e}_x(t)$ to be equal to zero, all terms on the right side of Equation (8.9) have to be equal to zero.

From the two required conditions for the error dynamics, the following equations can be derived from the Equation (8.9), considering each term for the Equation (8.9) to be zero. In solving this equation, $x(t)$ cannot be zero which means that $(A - LC - L_1)$ must be zero; the bilinear term $x(t)F_R(t)$ can also not be zero which means that $(B_1 - L_2)$ must be zero; $F_R(t)$ also, cannot be zero and therefore $(B - L_3)$ must be zero; and finally the error $e(t) = e_x(t)$ must be zero which means that the term $(L_1 + L_2 F_R(t))$ cannot be zero, Equations (8.10)–(8.13). The first three equations can be solved using the first condition, Equation (8.2) and the last equation can be solved using the second condition, Equation (8.3).

$$A - LC - L_1 = 0 \tag{8.10}$$

$$B_1 - L_2 = 0 \tag{8.11}$$

$$B - L_3 = 0 \tag{8.12}$$

$$\left. \begin{array}{l} L_1 + L_2 F_R(t) \neq 0\,, \\ \end{array} \right\} \forall \ t, t = t_0 \tag{8.13}$$

The solutions to the Equations (8.10) to (8.13) are given as:

$$L_2 = B_1 \tag{8.14}$$

$$L_3 = B \tag{8.15}$$

$$L_1 = A - LC \tag{8.16}$$

And from equations (8.13) and (8.16)

$$(L_2 F_R(t) + A - LC) \times e(t) \to 0 \tag{8.17}$$

according to condition Equation (8.3).

In order to determine the L_1 state observer matrix, firstly, the observer output matrix L must be obtained. L is determinable from the second condition where $[B_1 F_R(t) + A - LC] \times e(t) = 0$, as the error $e(t)$ and the rate of change of the error $\dot{e}(t)$ have to be zero for every moment of time. For this condition the requirement is that;

$$\begin{aligned} \dot{e}(t) &= \left[\dot{x}(t) - \dot{\hat{x}}(t)\right] = 0 \\ \dot{e}(t) &= \left[Ax(t) + B_1 x(t) F_R(t) + BF_R(t)\right] - \\ &\qquad - \left[L_1 \hat{x}(t) + L_2 \hat{x}(t) F_R(t) + L_3 F_R(t) + Lz(t)\right] = \\ &= \begin{bmatrix} Ax(t) - LCx(t) - L_1 \hat{x}(t) + B_1 x(t) F_R(t) - L_2 \hat{x}(t) F_R(t) + \\ + (B - L_3) F_R(t) \end{bmatrix} = 0 \end{aligned} \tag{8.18}$$

Using Equations (8.15)–(8.17), Equation (8.18) can be rewritten as

$$\begin{aligned} \dot{e}(t) &= \begin{bmatrix} (A - LC)x(t) - (A - LC)\hat{x}(t) + \\ + (B_1 x(t) - B_1 \hat{x}(t)) F_R(t) + (B - B) F_R(t) \end{bmatrix} = \\ &= (A - LC)(x(t) - \hat{x}(t)) + F_R(t) B_1 (x(t) - \hat{x}(t)) = 0 \end{aligned} \tag{8.19}$$

The error derivative from the Equation (8.19) can be expressed as follows

$$\begin{aligned} \dot{e}(t) &= \left[A - LC + F_R(t) B_1\right](x(t) - \hat{x}(t)) = \\ &= \left[A - LC + F_R(t) B_1\right] e(t) = 0 \end{aligned} \tag{8.20}$$

where $A_{err} = [A - LC + B_1 F_R(t)]$ – is the error matrix in Equation (8.20).

Equation (8.20) has to converge to zero and the requirement to satisfy this condition is that the term $[(A - LC) + F_R(t) B_1]$ is to be a *stable constant matrix*, i.e., *real part of the poles of the characteristic equation* of the error matrix in Equation (8.20) *must be negative*. After determining the poles of the desired characteristic equation of the observer it becomes possible to determine the value of L, which at this stage is still unknown.

This bilinear case differs from the linear case in that the coefficients of the observer have to be calculated for every moment of time in real-time because the value of the input $F_R(t)$ will change with time, depending on the input concentration of the process. This means that the *condition for stability* has to be determined for every moment of time. This will result in increased time for calculation of the estimates a bit.

The final equations of the observer are

$$\dot{\hat{x}}(t) = (A - LC)\hat{x}(t) + B_1\hat{x}(t)F_R(t) + BF_R(t)\,,\ \hat{x}(t_0) = \hat{x}_0 \tag{8.21}$$

or

$$\dot{\hat{x}}(t) = A\hat{x}(t) + B_1\hat{x}(t)F_R(t) + BF_R(t) + Lz(t)\,,\ \hat{x}(t_0) = \hat{x}_0 \tag{8.22}$$

The Equations (8.20) and (8.22) are used further for simulation of the observer behaviour under various control input values. The results are given in the last section of this chapter, Section 8.6.

8.3.2. Procedure for design of the observer gain matrix L

The procedure for design of the observer matrix L is built on the basis of the pole placement method using Equation (8.20) and following the requirement that the error matrix $A_{err} = [A - LC + F_R(t)B_1]$ has to be a stable matrix. Two cases are considered:

1) the case where the input signal varies with time, this is more of the practical approach, i.e., values of the control input coming from the controller will vary based on the process behaviour at the time; and
2) the case where the input is considered constant in time.

8.3.3. Determination of the gain matrix L based on every moment of time calculations with varying input signal

1) Trajectory of the control (input) is given, $u(t) = F_R(t)$, $t = 0, 1, 2, ..., t_f$
2) The observer error matrix $A_{err} = \left[A - L(t)C + B_1F_R(t)\right]$ is derived for every moment of time $t = t_0 : t_f$, where $L(t)$ is the unknown observer matrix at every moment of time based on $F_R(t)$ value; $L(t)$ is considered a constant matrix for each separate moment of time, as follows:
   2.1) Specify the input signal $u(t) = F_R(t)$

2.2) Derivation of the determinant of the observer error matrix A_{err} given by $\det_{error}(s) = |sI - A_{err}|$ and formation of the characteristic equation of the observer,

2.3) Calculation of the observer matrix $L(t)$

2.3.1) Determination of the desired poles and formation of the desired characteristic equation, $\det_{des}(s)$

2.3.2) Comparison of the two characteristic equations and calculation of the observer matrix $L(t) = const$ for $t = 0, 1, 2, ..., t_f$

8.3.4. Determination of the gain matrix L based on constant input signal

The same algorithm as from Section 8.3.2.1 is used, but in this case the process input $F_R(t)$ is constant, i.e., $u(t) = F_R = const$ for $t = 0, 1, 2, ..., t_f$. The procedures for calculation of the observer matrix are described in Section 8.6. MATLAB/SIMULINK software programs have been developed for both cases and the program listing presented in APPENDICES D1–D3.

8.4. Observer design for the CCIX (continuous countercurrent ion exchange) process bilinear model in the presence of a constant disturbance

The second case considered for solving state estimation problem for the continuous countercurrent ion exchange (CCIX) process is the case where the system disturbance is included in the model equation. In this model formulation one of the main assumptions is that the process experiences a piecewise constant disturbance. The disturbance is considered as the change of the concentration of the liquid that enters the first stage (the most bottom stage) of the process column in the CCIX. This disturbance has been modelled as:

$$x_f(t) = x_{in}(t) \tag{8.23}$$

In this case, the disturbance is no longer zero. The process model equation now becomes

$$\dot{x}(t) = Ax(t) + B_1x(t)F_R(t) + BF_R(t) + Wx_f(t) \tag{8.24}$$

The observer equation for the process model given in Equation (8.24) is described by

$$\dot{x}(t) = L_1\hat{x}(t) + B_1\hat{x}(t)F_R(t) + L_3F_R(t) + L_4\hat{x}_f(t) + Lz(t) \tag{8.25}$$

Based on the condition for error dynamics of the observer estimated states and the model states, Equations (8.24) and (8.25), the condition of zero error for the rate of change of the error as expressed by Equation (8.7) $\dot{e}(t) \to 0$ simplifies to

$$\begin{aligned}\dot{e}(t) = &\left[Ax(t) + B_1 x(t) F_R(t) + BF_R(t) + Wx_f(t)\right] - \\ &-\left[L_1 \hat{x}(t) + L_2 \hat{x}(t) F_R(t) + L_3 F_R(t) + L_4 x_f(t) + Lz(t)\right]\end{aligned} \quad (8.26).$$

Generally the output $z(t)$ can be expressed using the output matrix and state variables as $z(t) = Cx(t)$. This equation is then substituted in the error Equation (8.26). The resulting error equation is given by

$$\begin{aligned}\dot{e}(t) = &\left[(A - LC)x(t)\right] + \left[(B - L_3)F_R(t)\right] + \left[B_1 x(t) F_R(t) + Wx_f(t)\right] + \\ &+\left[-L_1 \hat{x}(t) - L_2 \hat{x}(t) F_R(t) - L_4 x_f(t)\right]\end{aligned} \quad (8.27)$$

Given that the error difference is defined by $e(t) = x(t) - \hat{x}(t)$; this means that $\hat{x}(t) = x(t) - e(t)$. The Equation (8.27) can therefore be expressed as

$$\begin{aligned}\dot{e}(t) = &\left[(A - LC)x(t)\right] + \left[(B - L_3)F_R(t)\right] + \left[B_1 x(t) F_R(t) + Wx_f(t)\right] + \\ &+\left[-L_1 x(t) + L_1 e(t) - L_2 x(t) F_R(t) + L_2 e(t) F_R(t) - L_4 x_f(t) + L_4 e(t)\right]\end{aligned} \quad (8.28)$$

The first condition for observer design states that $e(t) \to 0$ for $t \to \infty$ and therefore the Equation (8.28) is simplified to

$$\begin{aligned}\dot{e}(t) = &\left[(A - LC - L_1)x(t)\right] + \left[(B - L_3)F_R(t)\right] + \left[(B_1 - L_2)x(t)F_R(t)\right] + \\ &+\left[(W - L_4)x_f(t)\right] + \left[L_1 + L_2 F_R(t) + L_4\right]e(t)\end{aligned} \quad (8.29)$$

The second condition for the error dynamics requires that $\dot{e}(t) = 0$ and then it follows that each term in Equation (8.29) must be zero for this condition to be true. From this requirement the following equations, Equations (8.30) to (8.33) can be generated. It is important to note that $x(t)$ and $F_R(t)$ cannot be zero in this case for the condition to hold, and therefore

$$(A - LC - L_1) = 0 \quad (8.30)$$

$$(B - L_3) = 0 \quad (8.31)$$

$$(B_1 - L_2) = 0 \quad (8.32)$$

$$(W - L_4) = 0 \quad (8.33)$$

From Equations (8.30) to (8.33), the observer matrices are determined, where

$$L_1 = A - LC \quad (8.34)$$

$$L_2 = B_1 \quad (8.35)$$

$$L_3 = B \quad (8.36)$$

$$L_4 = W \tag{8.37}$$

In the same fashion as the case where the disturbance is not considered, the observer synthesis is reduced into the choice of the observer matrix L as clearly seen from the Equations (8.35) to (8.37). In designing the observer, one has to make sure that the observer poles are on the far left section in a complex plane for stability. The observer matrix, matrix L needs to be determined from the second condition by substituting the matrices backwards using Equations (8.35) to (8.37).

$$\begin{aligned}\dot{e}(t) &= \left[\frac{dx(t)}{dt} - \frac{d\hat{x}(t)}{dt}\right] = 0 = \\ &= \left[Ax(t) + B_1 x(t) F_R(t) + BF_R(t) + Wx_f(t)\right] - \\ &\quad - \left[(A - LC)\hat{x}(t) + B_1 \hat{x}(t) F_R(t) + BF_R(t) + Wx_f(t) + LCx(t)\right]\end{aligned} \tag{8.38}$$

$$\begin{aligned}\dot{e}(t) = \left[(A-LC)x(t) - (A-LC)\hat{x}(t)\right] + \left[(B_1 x(t) F_R(t) - B_1 \hat{x}(t) F_R(t))\right] + \\ + \left[BF_R(t) - BF_R(t)\right] + \left[Wx_f(t) - Wx_f(t)\right] = 0\end{aligned} \tag{8.39}$$

$$\begin{aligned}\dot{e}(t) = \left[(A-LC)(x - \hat{x}(t))\right] + \left[F_R(t) B_1 x(t) - F_R(t) B_1 \hat{x}(t)\right] + \\ + \left[W(x_f(t) - x_f(t))\right] = 0\end{aligned} \tag{8.40}$$

$$\dot{e}(t) = \left[(A-LC)(x(t) - \hat{x}(t))\right] + \left[F_R(t) B_1 (x(t) - \hat{x}(t))\right] = 0 \tag{8.41}$$

$$\dot{e}(t) = \left[(A-LC)e(t) + F_R(t) B_1 e(t)\right] = \left[A - LC + B_1 F_R(t)\right] e(t) = 0 \tag{8.42}$$

where $F_R \in R^1$ – is a constant input,

For Equation (8.42) to converge to zero, the requirement is that the term $(A - LC) + B_1 F_R(t)$ must be a stable matrix. From this requirement,

1) the entries for the observer matrix L can be determined,
2) but again, for every moment of time in which it is accepted that the matrix

$$[(A - LC) + B_1 F_R] = const \tag{8.43}$$

Comparison of the Equations (8.21) and (8.43) shows that the *condition for stability* of the observer for both cases, – without or with a presence of the constant system disturbance is the same, and this allows the same design method of the observer gain matrix L to be used for both cases.

The equations of the observer for the case with a constant disturbance are Equation (8.44) or Equation (8.45)

$$\dot{\hat{x}}(t) = A\hat{x}(t) + B_1 \hat{x}(t) F_R(t) + BF_R(t) + Wx_f(t) + Lz(t)\,, \ x(t_0) = x_0 \tag{8.44}$$

$$\dot{\hat{x}}(t) = (A - LC)\hat{x}(t) + B_1 F_R(t) \hat{x}(t) + BF_R(t) + Wx_f(t)\,, \ \hat{x}(t_0) = \hat{x}_0 \tag{8.45}$$

Simulation based on the equations are performed and described in the results section of this chapter, Section 8.9., and the associated software programs used are presented in APPENDIX D.1–D.3 The same procedure for calculation of the observer gain matrix L as used in the case of no disturbance, is also used for the solution that considers the disturbance with a varying and a none varying input signal.

From these interpretations, the observer design problem can be summarized as a problem of choosing or selecting the observer gain matrix L such that the error rate dynamics goes to zero (Conticelli & Bicchi, 2000; Raghavan & Hedrick, 1994). Firstly, the developed MATLAB software program is used to find the model matrices and then the SIMULINK environment used to determine the error rate dynamics based on matrices values obtained from MATLAB workspace.

8.5. Pole placement method for design of the observer matrix

8.5.1. Procedure for design of the observer gain matrix

The procedure for design of the matrix L is built on the basis of the pole placement method. The solution for the procedure is given in two folds, 1) the case where the input is assumed constant for the entire process time trajectory, and 2) the case where the input continually changes over time over the full trajectory. The more realistic approach into the real plant behaviour is the one of the assumption that the input is varying in time as the plant operation progresses. The solution is derived from the stability requirement of the error rate dynamics, Equation (8.42). This requirement translates to the pole placement procedure for system stability, i.e., the real parts of the poles of the characteristic equation of the observer error must be on the negative side of the Cartesian plane, but not far from the imaginary line. This is the reason why the solution is also considered the pole placement method. This method requires that the observer error matrix be calculated every moment in time, and in real-time, because the value of the input signal will be also changing at every moment. Two characteristic equations of the observer are needed for the solution, one from Equation (8.43) and the desired one, Equations (8.47) and (8.48) respectively.

$$\det_{err}(s) = \left| sI - \left[A - LC + F_R(t)B_1 \right] \right| = \left| sI - A_{err} \right| \qquad (8.46),$$

$$\begin{aligned} &\det_{des}(s) = (s+p_1)(s+p_2)\ldots(s+p_N) = 0 \quad or \\ &\det_{des}(s) = (s+p)^N = 0 \end{aligned} \qquad (8.47)$$

where

s – is the Laplace variable,

$L = [l_1 \quad l_2 \quad ... \quad l_N]^T \in R^{N \times 1}$ - are the elements of the observer gain matrix,

$p_1, p_2, ..., p_N$ – are the desired poles for every stage of the process,

p – is the desired pole, of equal value for all column stages, and

$A_{err} = [A - LC + F_R(t)B_1]$.

The desired characteristic equation is solved by placing all the N necessary poles at the negative side of the Cartesian plane, Equation (8.47) and then determining the corresponding polynomial equation which is to be equated to the characteristic Equation (8.46) solved using the determinant derivation.

8.5.2. Derivation of the determinant equations

The *observer error characteristic equation*, Equation (8.46) is derived for the considered case of the ion exchange process as follows:

$$\det_{error}(s) = \det\left\{ \begin{array}{l} \begin{bmatrix} s & 0 & ... & 0 & 0 \\ 0 & s & 0 & ... & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & ... & ... & s \end{bmatrix} - \\ -\begin{bmatrix} -a_{11} & 0 & ... & ... & ... & 0 \\ a_{21} & -a_{22} & 0 & ... & ... & 0 \\ 0 & a_{32} & -a_{33} & 0 & ... & 0 \\ 0 & ... & ... & \ddots & 0 & 0 \\ 0 & ... & ... & ... & a_{65} & -a_{66} \end{bmatrix} + \\ +[l_1 \quad l_2 \quad ... \quad l_6]^T \times [0 \quad 0 \quad ... \quad 1] - \\ -F_R(t) \times \begin{bmatrix} -b_{11} & b_{12} & 0 & ... & 0 & 0 \\ 0 & -b_{22} & b_{23} & 0 & ... & 0 \\ 0 & 0 & \ddots & 0 & ... & 0 \\ 0 & ... & ... & \ddots & -b_{55} & b_{56} \\ 0 & ... & ... & ... & 0 & -b_{66} \end{bmatrix} \end{array} \right\} \quad (8.48)$$

where $l_i, \ i = \overline{1,6}$ – are the parameters of the observer matrix L and these matrix elements are unknown.

In evaluating Equation (8.48), the resulting simplified expression of the $\det_{error}(s)$ is a $R^{6\times6}$ matrix given by Equation (8.49),

$$\det_{error}(s) = \det\{[sI - A_{err}]\} \in R^{6\times6} \quad (8.49)$$

where

$$\det_{err}(s) =$$

$$= \begin{vmatrix} s+a_{11}-g_{11} & g_{12} & 0 & 0 & 0 & l_1 \\ -a_{21} & s+a_{22}-g_{22} & g_{23} & 0 & 0 & l_2 \\ 0 & -a_{32} & s+a_{33}-g_{33} & g_{34} & 0 & l_3 \\ 0 & 0 & -a_{43} & \ddots & \ddots & l_4 \\ 0 & 0 & 0 & \ddots & \ddots & (l_5+g_{56}) \\ 0 & 0 & 0 & 0 & -a_{65} & (s+a_{66}-g_{66}+l_6) \end{vmatrix}$$

with $g_{ii} = F_R(t)b_{ii}$, $i = \overline{1, N}$, i.e., $i = \overline{1,6}$,

$g_{ij} = F_R(t)b_{ij}$, $j = \overline{2,5}$

The resulting determinant can be represented as a sum of sub-determinants considering its first row which further aids to simplifying the calculation, Equations (8.50) and (8.51),

$$\det_{err}(s) = \det_1(s) + \det_2(s) + \det_6(s) = \det_{obs}(t) \tag{8.50}$$

$$\det_1(s) = (s+a_{11}-g_{11})\times$$

$$\times \begin{vmatrix} s+a_{22}-g_{22} & g_{23} & 0 & 0 & l_2 \\ -a_{32} & s+a_{33}-g_{33} & g_{34} & 0 & l_3 \\ 0 & -a_{43} & \ddots & \ddots & l_4 \\ 0 & 0 & \ddots & \ddots & (g_{56}+l_5) \\ 0 & 0 & 0 & -a_{65} & (s+a_{66}-g_{66}+l_6) \end{vmatrix} \tag{8.51}$$

$$\det_2(s) = -g_{12}\times$$

$$\times \begin{vmatrix} -a_{21} & g_{23} & 0 & 0 & l_2 \\ 0 & s+a_{33}-g_{33} & g_{34} & 0 & l_3 \\ 0 & -a_{43} & s+a_{44}-g_{44} & g_{45} & l_4 \\ 0 & 0 & \ddots & \ddots & (g_{56}+l_5) \\ 0 & 0 & 0 & -a_{65} & (s+a_{66}-g_{66}+l_6) \end{vmatrix} \tag{8.52}$$

$$\det_6(s) = -l_1\times$$

$$\times \begin{vmatrix} -a_{21} & s+a_{22}-g_{22} & g_{23} & 0 & 0 \\ 0 & -a_{32} & s+a_{33}-g_{33} & g_{34} & 0 \\ 0 & 0 & -a_{43} & s+a_{44}-g_{44} & g_{45} \\ 0 & 0 & 0 & -a_{54} & s+a_{55}-g_{55} \\ 0 & 0 & 0 & 0 & -a_{65} \end{vmatrix} \tag{8.53}$$

Derivation of the determinant of the observer error matrix continues in the same fashion until all sub-determinants are calculated, $\det_1(s)$, $\det_2(s)$ and $\det_6(s)$, and the characteristic equation of the observer is obtained. The sub-determinant equations are presented considering their first rows as follows:

$$\begin{aligned}\det_1(s) &= (s+a_{11}-g_{11})\times \\ &\times\left[(s+a_{22}-g_{22})\times\det_{11}(s)-g_{23}\times\det_{12}(s)+l_2\times\det_{16}(s)\right]\end{aligned} \quad (8.54)$$

$$\det_2(s) = g_{12}\left[a_{21}\times\det_{21}(s)-g_{23}\times\det_{22}(s)+l_2\times\det_{26}(s)\right] \quad (8.55)$$

$$\begin{aligned}\det_6(s) &= l_1\times \\ &\times\left[-a_{21}\times\det_{61}(s)-(s+a_{22}-g_{22})\times\det_{62}(s)+g_{23}\times\det_{63}(s)\right]\end{aligned} \quad (8.56)$$

This procedure is continued until one equation written according to the powers of the Laplace variable is obtained. This final equation is used for calculation of $L(t)$ at every moment of time $t=\overline{0,t_f}$. The desired characteristic equation is calculated for a real negative desired pole $p_i=5$, $i=\overline{1,N}$. Determination of this equation is via a MATLAB software *system functions*, *simplify*( ) and *expand*( ) functions,

$$\left.\begin{array}{l} syms\quad s \\ ce = simplify\,((s+\ p\,)\verb|^|\,N\,) \\ p = expand\,(\,ce\,) \end{array}\right\} \quad (8.57)$$

where ce stands for characteristic equation.

After evaluating both characteristic equations, they are equated to determine the values of $l_i(t)$, $t=\overline{0,t_f}$ for $l_i, i=\overline{1,N}$.

$$\begin{aligned}\det_{des}(s) &= s^6+30s^5+375s^4+2500s^3+9375s^2+18750s+15625= \\ &=\det_{obs}(l_i),\ i=\overline{1,6}\end{aligned} \quad (8.58)$$

Calculation of the elements of $L(t)$ can be based on mathematical derivation or numerical solution of Equation (8.58).

8.5.3. Algorithm for the observer design for the CCIX using the pole placement method for determination of the L matrix

The algorithm for calculation of the observer gain matrix based on the derivation in Subsection 8.5.2 is given by Figures 8.1 and 8.2. Its procedure is as follows:

1) Give a trajectory of $F_R(t), t=\overline{0,t_f}$,
2) Select the input signal $F_R=F_R(t)$ – for a given time $t, t=\overline{0,t_f}$

3) Form the mathematical expression of the determinant of the observer error $\det_{error}(s),$ with unknown observer gain matrix L, $L(t) = [l_1(t), l_2(t), ..., l_N(t)]^T$,
4) Form the mathematical expression of the desired determinant $\det_{des}(s)$,
5) Compare the two determinants and calculate the gain matrix elements, $l_i(t),\ i = \overline{1, N}$.

The solution to step 5) can be obtained numerically or analytically, e.g., using $fsolve()$ function in MATLAB software program provides a numerical solution. The thesis follows the analytical technique of back substitution between the two characteristic equations. The derivation is expressed in Figure 8.3.

8.6. Algorithm and MATLAB software programs developed for calculation and verification of the observer matrix L

8.6.1. Simulation procedure

The developed algorithms for design of the observer are used, first to calculate the observer matrices, and second, to validate the performance of the observer in a case of the closed loop system that consists of the models of the process, the observer and a controller. These tasks are performed by the algorithms developed in MATLAB software implementing the following simulation procedure APPENDIX D1–D3. Steps for the simulation follow three stages:

Stage I – MATLAB workspace program for determining model parameters and determinants

1) Process model parameters are obtained using provided experimental data,
2) The observer error determinant is calculated following the procedure in Section 8.5.2,
3) The desired determinant is calculated,
4) Then the observer gain matrix $L(t)$ unknown entries l_i, $i = \overline{1, N}$ are derived using determinant of the observer error, and the desired one.

Stage II – SIMULINK program is used to simulate the closed loop system,

1) The values of the observer matrix L are sent to the SIMULINK environment, Figure 8.3,
2) The simulation is run for the given period of time,

3) The error and the derivative of the error of the observer are calculated,

4) If the errors are greater than some given small numbers, ε_1 and ε_2, the procedures in stage I and stage II are repeated,

5) If the errors are acceptable the procedure stops and the results can be plotted.

Stage III – MATLAB workspace program used for extracting trajectory values and plotting these values:

1) This program is used to extract all trajectory data points as calculated in the Stage II using SIMULINK,

2) Trajectories are displayed as shown in Figure 8.4–Figure 8.21.

The proposed algorithm for design, Figure 8.1 can be used for real-time control too, when the control action is changing, but having constant values in the sampling periods for the calculation of the observer gain matrix L, this is demonstrated by Figure 8.3.

In the real-time case, the algorithm is run for every new value of the control signal and the obtained value of L is directly used for calculation of the estimated state values. In turn, these values are used for calculation of the control action. The structure of the algorithm is similar as in Figure 8.1., but the difference is that the real-time system replaces the SIMULINK block, Figure 8.2, and the calculated observer gain matrix is directly used to produce the real-time state estimates.

The MATLAB software program developed for process simulation is in three stages; the first part is used to calculate the model parameters based on the experimental data, and to calculate the observer matrix $L(t)$. Once observer model parameters are determined, the second phase of the program is to simulate the system error based on specified control input value for the full trajectory of the system until the *error rate dynamics* requirement is met. SIMULINK environment in the MATLAB software is used for this part, Figure 8.3.

From the SIMULINK trajectories, values of the state vector and the estimated state vector are generated. The program is then run at different *input signals*, and *estimated state vector* initial conditions as shown in the results section. A few runs are presented in the results section. The last part of the software developed is the MATLAB program in MATLAB environment used to extract both the state vectors from the SIMULINK environment. The same program is used to plot the two trajectories.

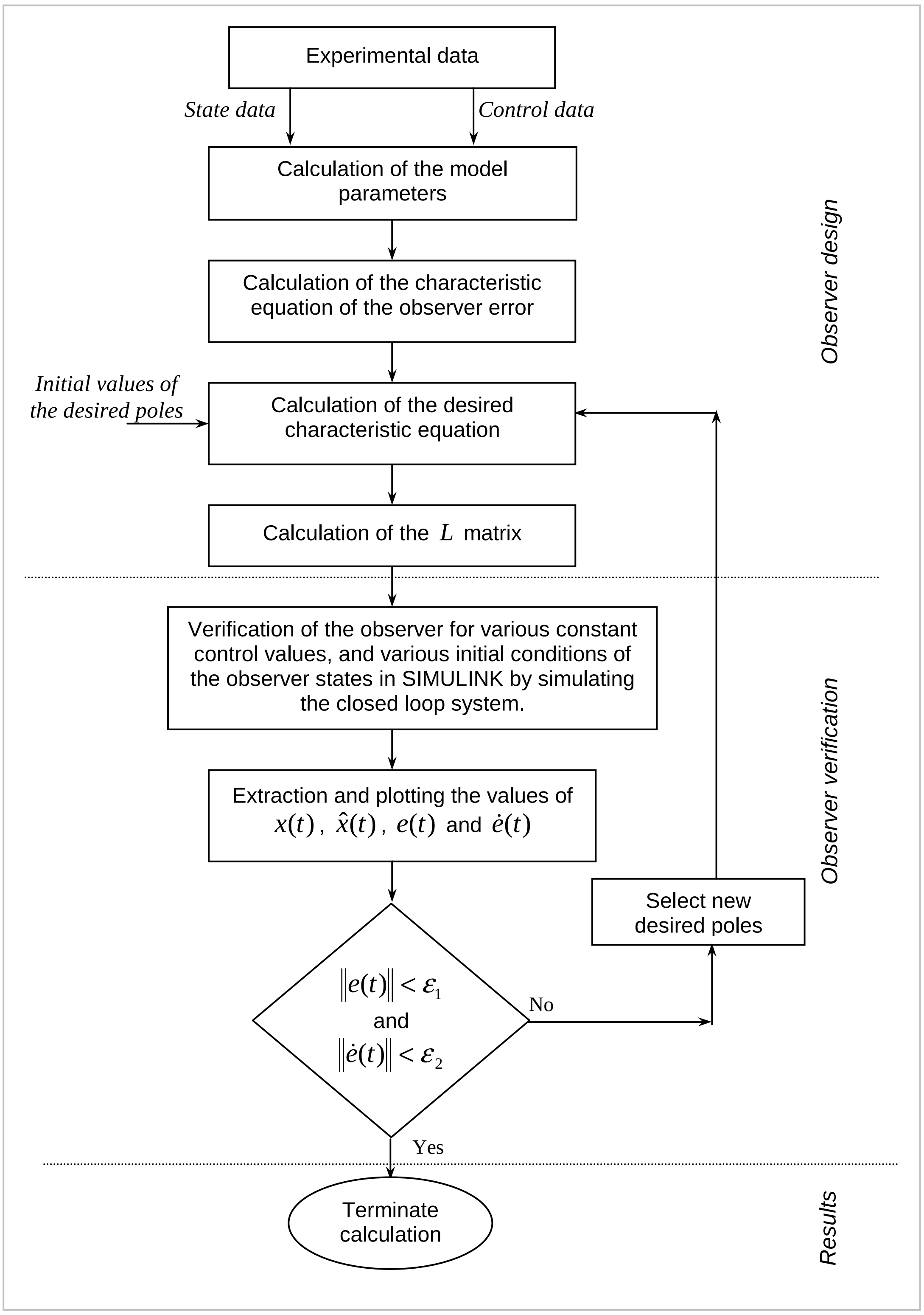
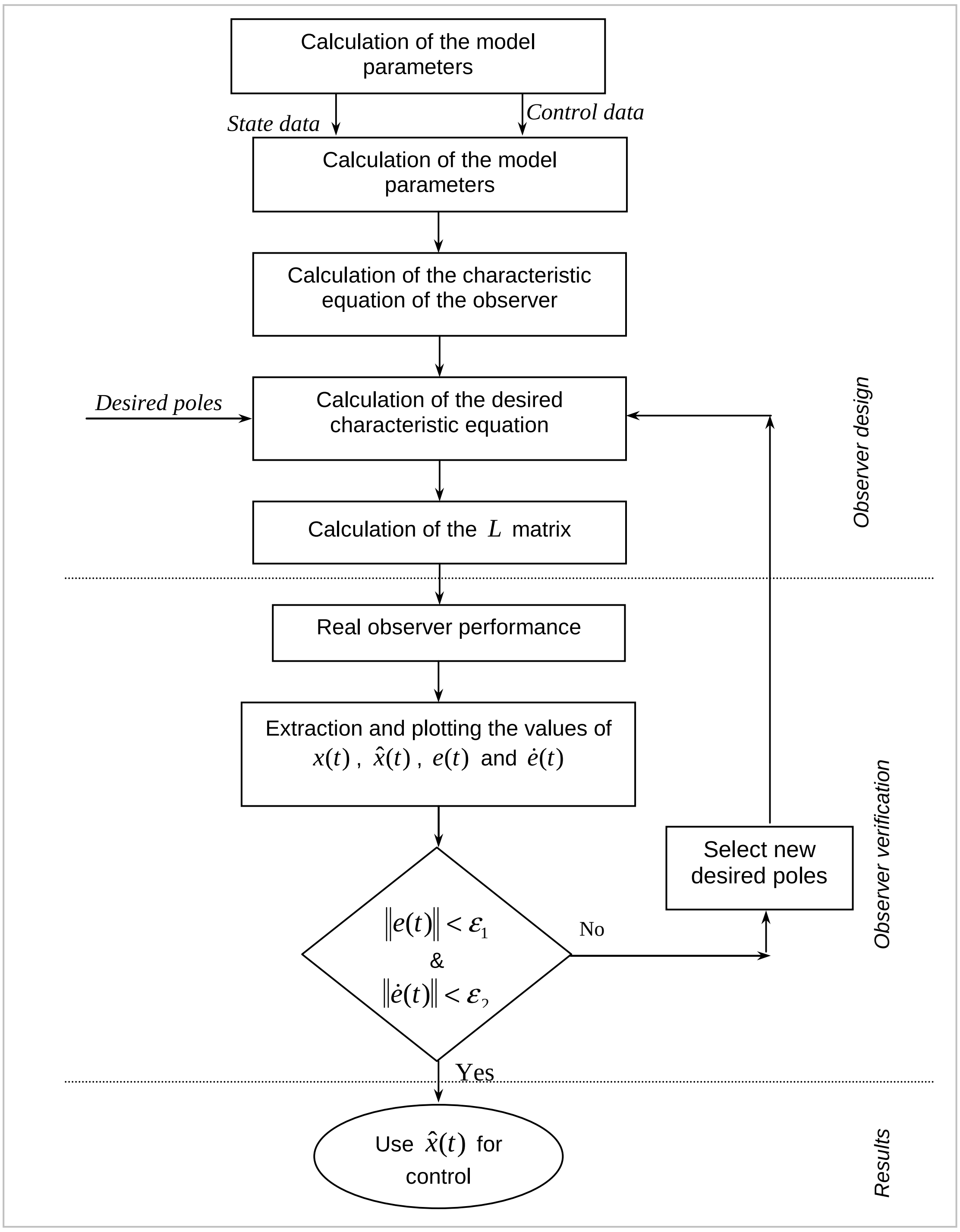

Experimental data
State data
Control data
Calculation of the model parameters
Calculation of the characteristic equation of the observer error
Initial values of the desired poles
Calculation of the desired characteristic equation
Calculation of the L matrix
Verification of the observer for various constant control values, and various initial conditions of the observer states in SIMULINK by simulating the closed loop system.
Extraction and plotting the values of $x(t)$, $\hat{x}(t)$, $e(t)$ and $\dot{e}(t)$
$\|e(t)\| < \varepsilon_1$ and $\|\dot{e}(t)\| < \varepsilon_2$
No
Yes
Select new desired poles
Terminate calculation
Observer design
Observer verification
Results

Figure 8.1. Algorithm for the design process



Calculation of the model parameters
State data
Control data
Calculation of the model parameters
Calculation of the characteristic equation of the observer
Desired poles
Calculation of the desired characteristic equation
Calculation of the L matrix
Observer design
Real observer performance
Extraction and plotting the values of $x(t)$, $\hat{x}(t)$, $e(t)$ and $\dot{e}(t)$
Select new desired poles
$\|e(t)\| < \varepsilon_1$ & $\|\dot{e}(t)\| < \varepsilon_2$
No
Yes
Observer verification
Use $\hat{x}(t)$ for control
Results

Figure 8.2. Algorithm for the real-time observer design process

8.7. Application of the design to the continuous countercurrent ion exchange (CCIX) process

8.7.1. Data used for experiments

Concentration measurements data from (Hendry, 1982a, Hendry, 1982b, Randall, 1984) were used to estimate state variables of the newly developed bilinear model of the continuous countercurrent ion exchange (CCIX) process. The results are based on normalized data of a six stages CCIX process column.

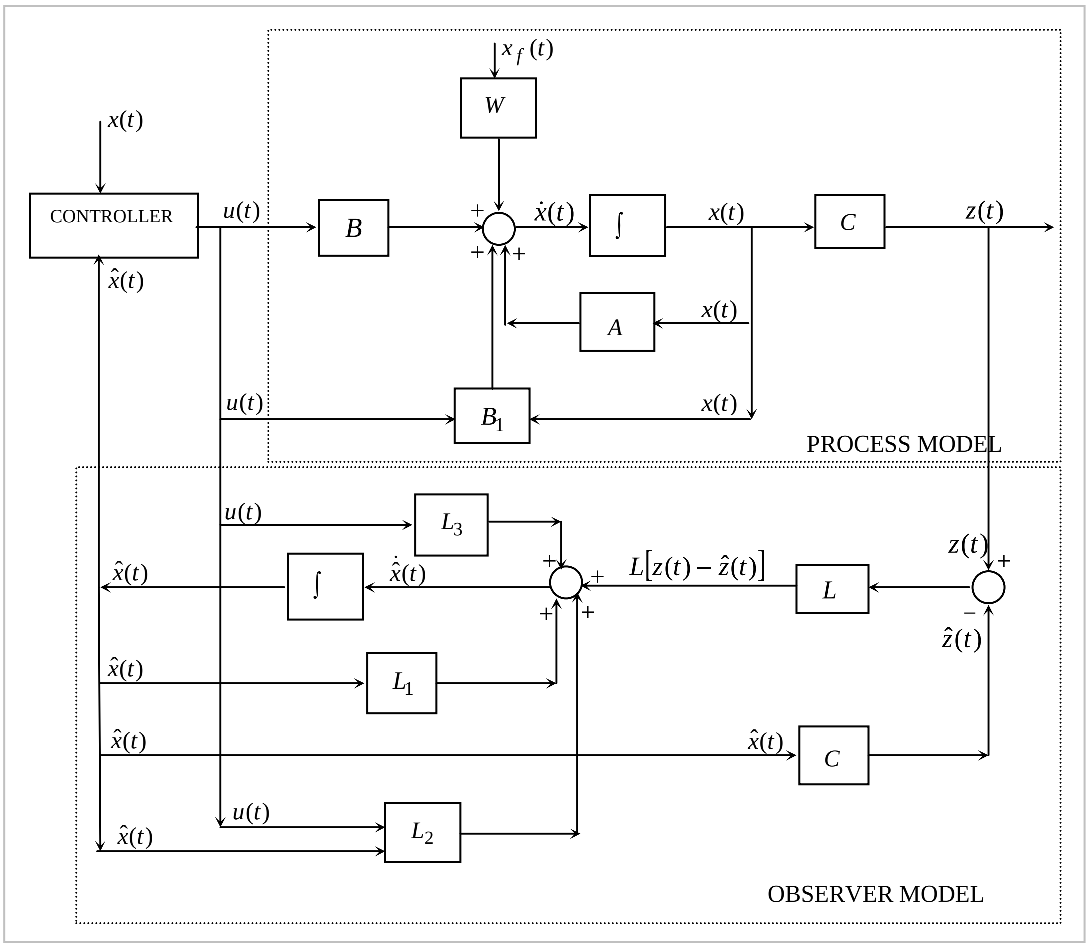


Figure 8.3. Simulation diagram of the process and observer system

Process parameters were calculated for the input flow rate $F_L = 2000 m^3/h$, the upflow period of $T = (17/60)h$, resin liquid constant ratio $d = 2/3$, the liquid holdups $42.809 l$ and the resin holdups of $32.93 l$. In the diagram the input signal $F_R(t)$ is represented by $u(t)$. Original data measured from the University of Cape Town (UCT) project [Hendry, 1982a] is presented in Table 8.1.

Table 8.1: CCIX data showing concentration in each stage of the cation column as per UCT project (Hendry, 1982a & Henry, 1982b)

Stage H^+ fractional change in liquid concentration using data obtained from UCT Project (Hendry, 1982b ,Volume 4)					
Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
0.221	0.000	0.000	0.000	0.000	0.000
0.577	0.140	0.040	0.000	0.000	0.000
0.730	0.314	0.066	0.004	0.000	0.000
0.847	0.523	0.184	0.035	0.003	0.000
0.920	0.656	0.295	0.082	0.020	0.000
0.936	0.766	0.454	0.168	0.052	0.001
0.968	0.842	0.601	0.277	0.113	0.024
0.974	0.886	0.690	0.361	0.167	0.033

0.981	0.900	0.758	0.440	0.207	0.063
0.989	0.933	0.804	0.522	0.340	0.124
0.988	0.958	0.877	0.698	0.474	0.167
0.997	0.963	0.881	0.784	0.547	0.233
1.000	0.982	0.951	0.899	0.780	0.482
1.000	0.974	0.965	0.931	0.860	0.539
1.000	0.991	0.972	0.966	0.899	0.672
1.000	0.994	0.981	0.966	0.905	0.779
1.000	0.993	1.000	0.991	0.975	0.940
1.000	0.993	0.988	0.991	0.973	0.972

This data set shows the concentration in *fractional change* of sodium ions, H^+ as determined from the equation,

$$FC_n = \frac{I.F.H^+_n - I.F.H^+_{initial}}{I.F.H^+_{final} - I.F.\cdot^+_{initial}} \qquad (8.59)$$

where

FC_n – is the fractional change (Hendry, 1982b)

$I.F.H^+_n$ – ion fraction of H^+ ions of the current measurement at the n^{th} cycle from step change moment (Hendry, 1982b),

$I.F.H^+_{initial}$ – is the initial ion fraction of the H^+ ions in the stage (Hendry, 1982b),

$I.F.H^+_{final}$ – is the final ion fraction of the H^+ ions in the stage (Hendry, 1982b).

Calculation of values for model matrices are based on data from the UCT project, Table 8.1., which were obtained using a six stage column ion exchange process (Hendry, 1982a; Hendry, 1982b). MATLAB software programs were run using the values to determine the unknown process parameters and were calculated to be:

$$A = \begin{bmatrix} -26.406 & 0 & 0 & 0 & 0 & 0 \\ 28.921 & -28.921 & 0 & 0 & 0 & 0 \\ 0 & 31.966 & -31.966 & 0 & 0 & 0 \\ 0 & 0 & 35.726 & -35.726 & 0 & 0 \\ 0 & 0 & 0 & 37.959 & -37.959 & 0 \\ 0 & 0 & 0 & 0 & 43.382 & -43.382 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} -0.013 & 0.011 & 0 & 0 & 0 & 0 \\ 0 & -0.012 & 0.009 & 0 & 0 & 0 \\ 0 & 0 & -0.010 & 0.006 & 0 & 0 \\ 0 & 0 & 0 & -0.007 & 0.0054 & 0 \\ 0 & 0 & 0 & 0 & -0.006 & 0.002 \\ 0 & 0 & 0 & 0 & 0 & -0.002 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.0400 & 0.0500 & 0.1000 & 0.1000 & 0.0500 & 0.3500 \end{bmatrix}^T,$$

$W = \begin{bmatrix} -26.4065 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ and $C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$

The observer gain matrix evaluated with the constant input signal value and model parameters from the model matrices is shown in Equation (8.61).

$$L = 1 \times 10^5 \begin{bmatrix} 0.782 & 1.975 & 1.235 & 0.024 & 0.020 & 0.0016 \end{bmatrix}^T \qquad (8.60)$$

8.7.2. Evaluation of the observer performance

Multiple simulation sets were performed with different *input signal values*, *initial conditions for the observer* and the same *initial conditions for the process*, as per Table 8.2. A selected sample of the runs is presented here. The steps followed in the simulation procedure are also presented.

Table 8.2: **Input and initial conditions values used in trajectory runs for the process and the observer**

	Trajectory for process states and observer estimated states				
	Run 1	Run 2	Run 3	Run 4	Run 5
Input value	0.0	1.0	5.0	10.0	20.0
Process initial conditions	1.0	1.0	1.0	1.0	1.0
Observer initial conditions					
Set 1	– 1.0	– 1.0	– 1.0	– 1.0	– 1.0
Set 2	0.0	0.0	0.0	0.0	0.0
Set 3	0.5	0.5	0.5	0.5	0.5
Set 4	1.0	1.0	1.0	1.0	1.0
Set 5	10.0	10.0	10.0	10.0	10.0
Set 6	20.0	20.0	20.0	20.0	20.0

8.8. Results and discussion

The following graphs show the behaviour of the process states (from the model) versus that of the estimated states (observer) based on SIMULINK simulation runs with a constant input signal of 1.0, constant initial conditions of the states of the model at 1.0, and the changing observer initial conditions of -1.0, 0, 1/2, 1.0, 5.0 and 10.0 for states; and they are the same for every stage of the column.

The observer has shown that it converges within a reasonable time and maintains the error rate close to zero for the rest of the observed period, Figures 8.4–8.21, Dube & Tzoneva (2016). The error (Figures 8.5, 8.8, 8.11, 8.14, 8.17, 8.20), and error rate (Figures 8.6, 8.9, 8.12, 8.15, 8.18 and 8.21) are also presented. The observer has also shown to be very sensitive to the input signal. The input signal values should also be kept within normalized values (0–1.0), otherwise overshoot will be experienced in some process stages. The observer gain matrix values tend to be very high; this could be associated with the determinants calculation that involves a very large number of computations.

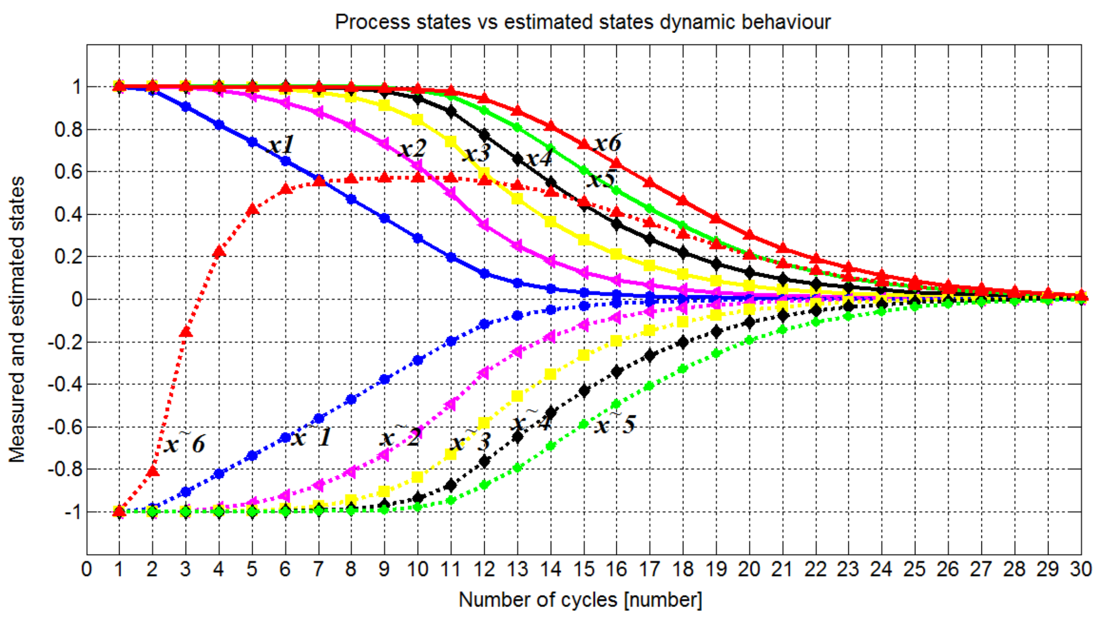


Figure 8.4. **Model states and estimated states for observer initial conditions of –1.0 and model initial conditions of 1.0 using a constant control input of 1.0.**

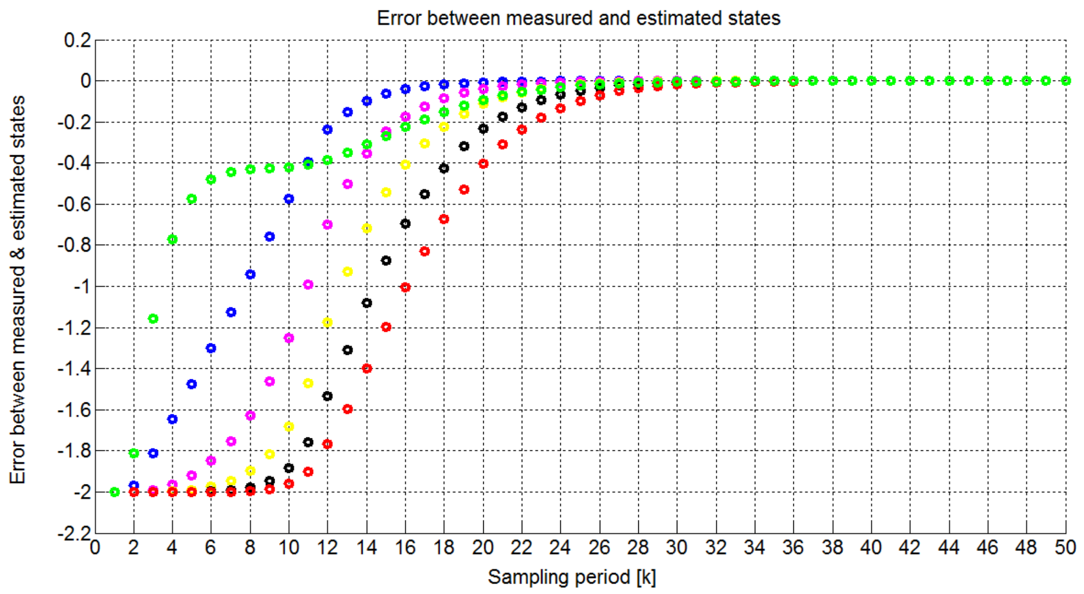


Figure 8.5. **Error between process states and estimated states for observer initial conditions of –1.0 and model initial conditions of 1.0 using a constant control input of 1.0.**

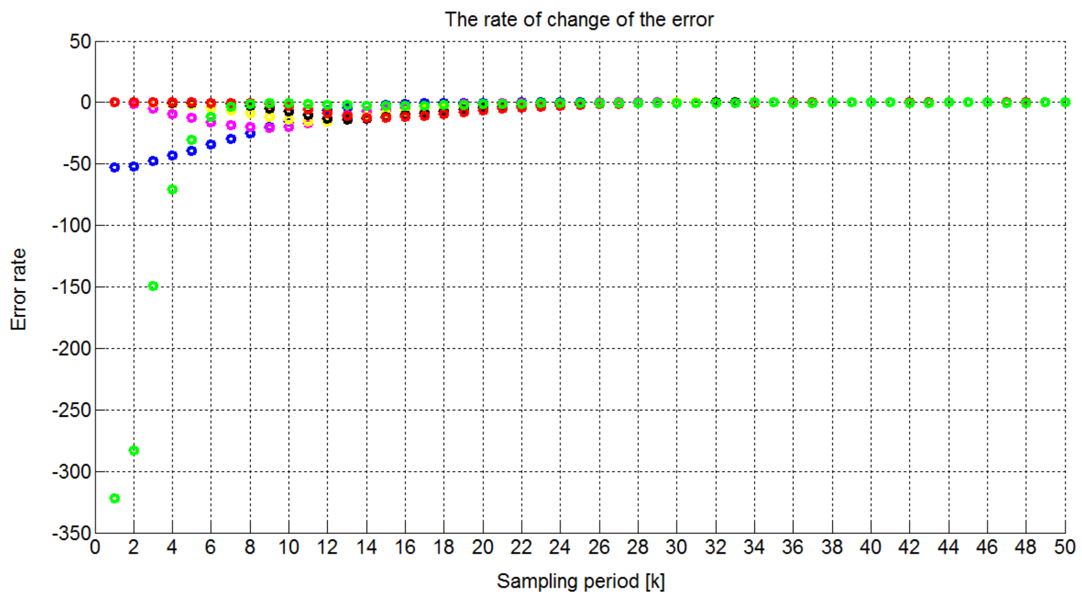


Figure 8.6. Rate of change of the error for observer initial conditions of –1.0 and model initial conditions of 1.0 using a constant control input of 1.0.

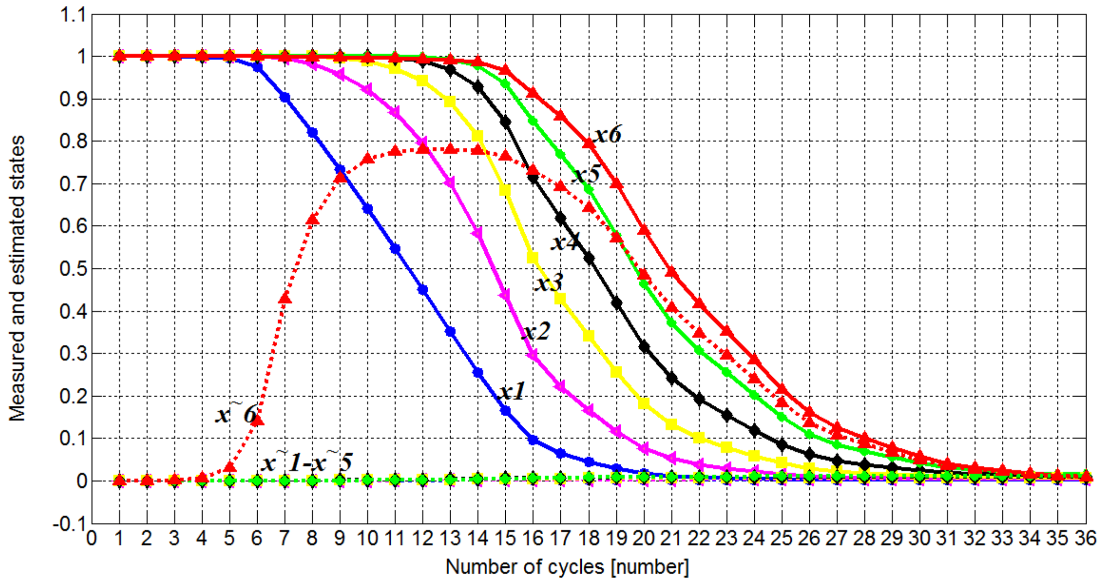


Figure 8.7. Model states and estimated states for observer initial conditions of 0.0 and model initial conditions of 1.0 using a constant control input of 1.0.

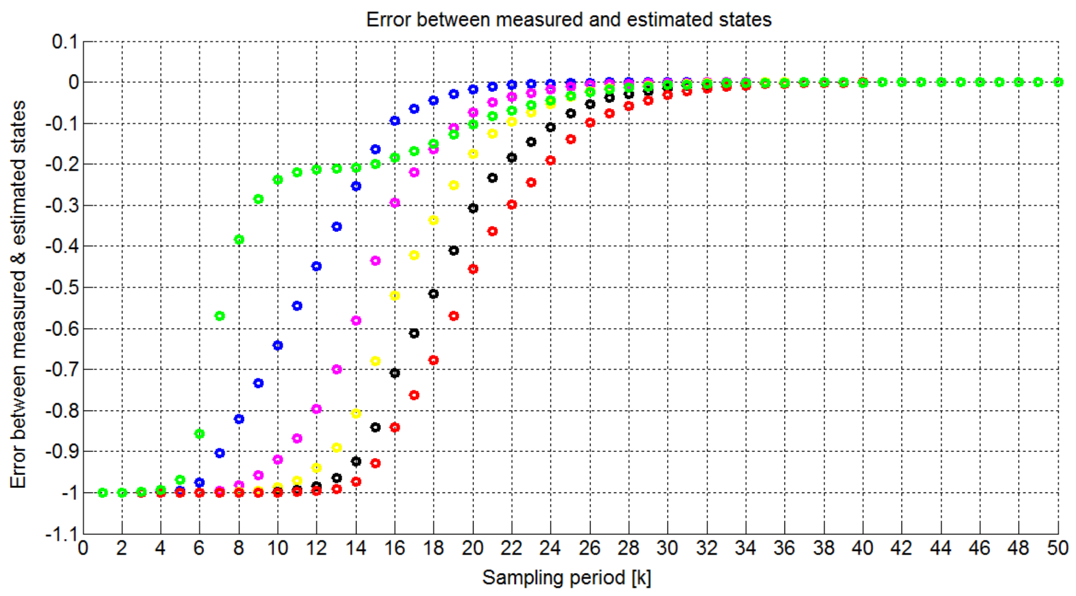


Figure 8.8. Error between process states and estimated states for observer initial conditions of 0.0 and model initial conditions of 1.0 using a constant control input of 1.0.

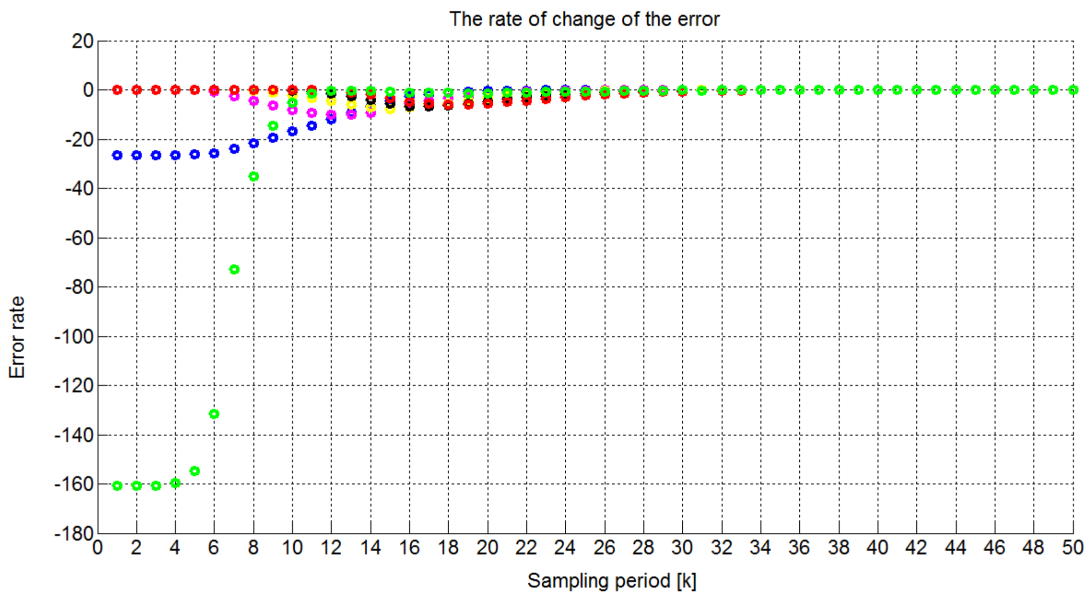


Figure 8.9. Rate of change of the error for observer initial conditions of 0.0 and model initial conditions of 1.0 using a constant control input of 1.0.

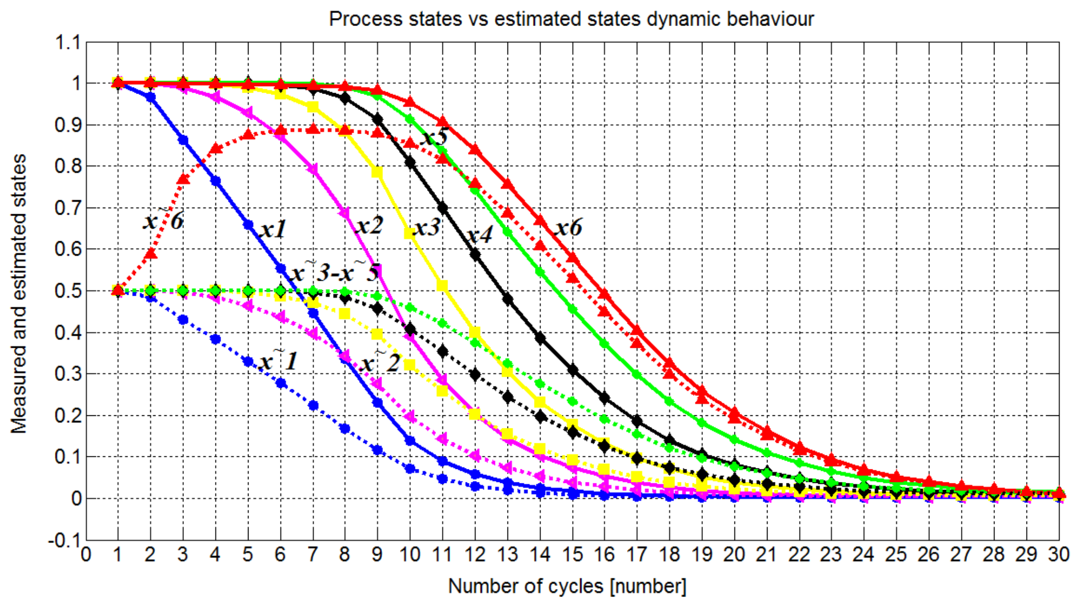


Figure 8.10. Model states and estimated states for observer initial conditions of ½ and model initial conditions of 1.0 using a constant control input of 1.0.

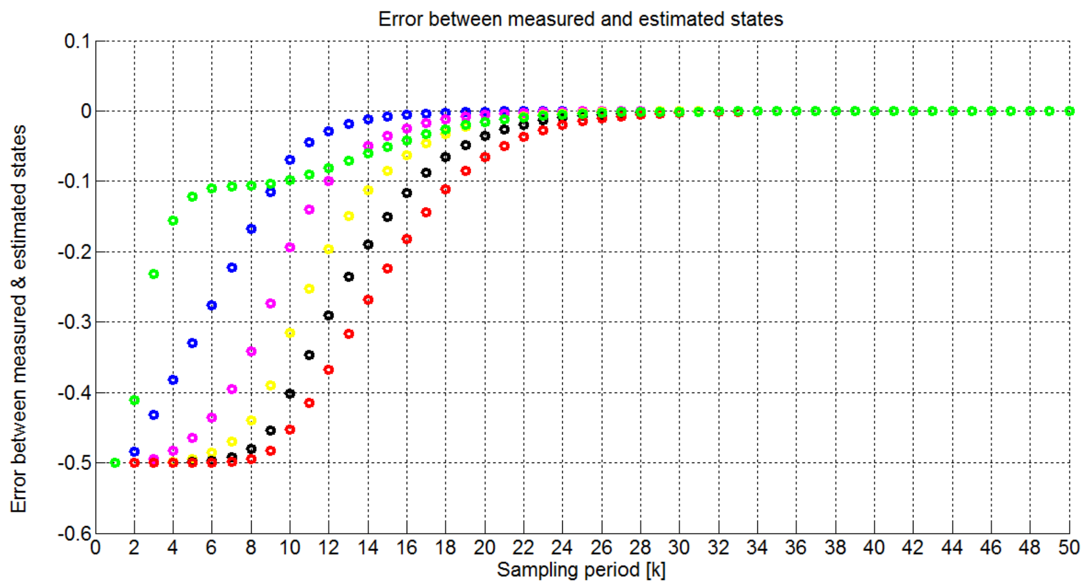


Figure 8.11. Error between process states and estimated states for observer initial conditions of ½ and model initial conditions of 1.0 using a constant control input of 1.0.

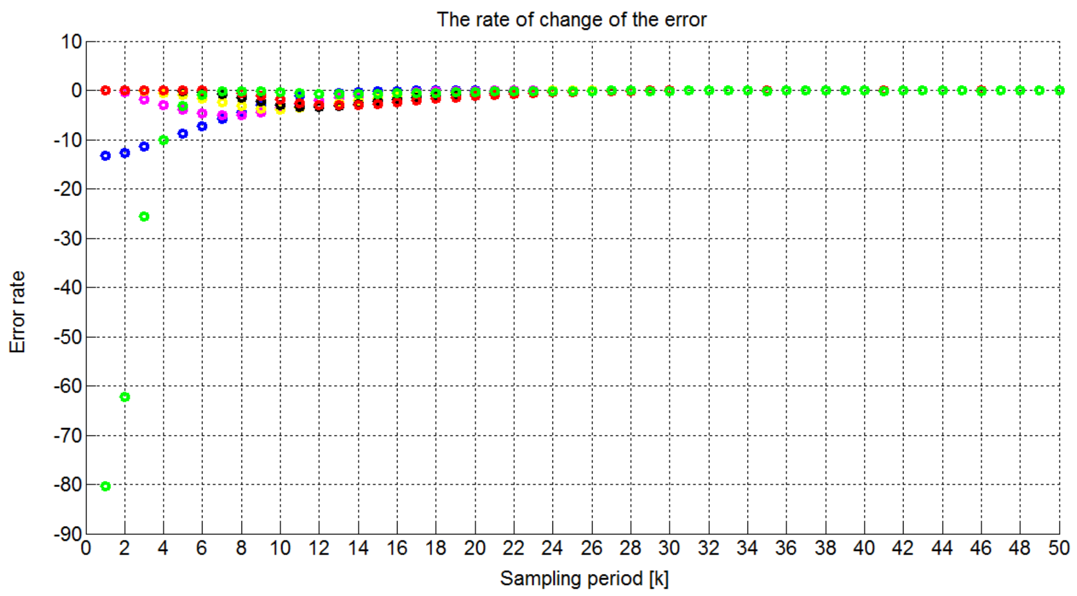


Figure 8.12. Rate of change of the error for observer initial conditions of ½ and model initial conditions of 1.0 using a constant control input of 1.0.

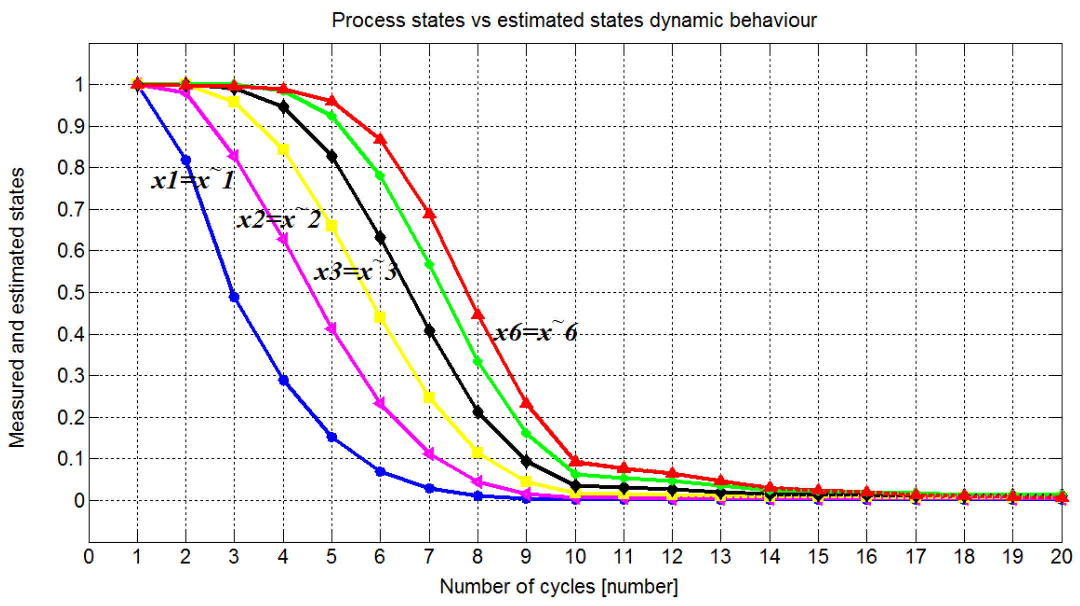


Figure 8.13. Model states and estimated states for initial conditions of 1.0 for both the observer and model using a constant input signal of 1.0.

In Figure 13, in the same fashion for $x1 = \tilde{x}1$, $x2 = \tilde{x}2$, etc., $x4 = \tilde{x}4$ and $x5 = \tilde{x}5$.

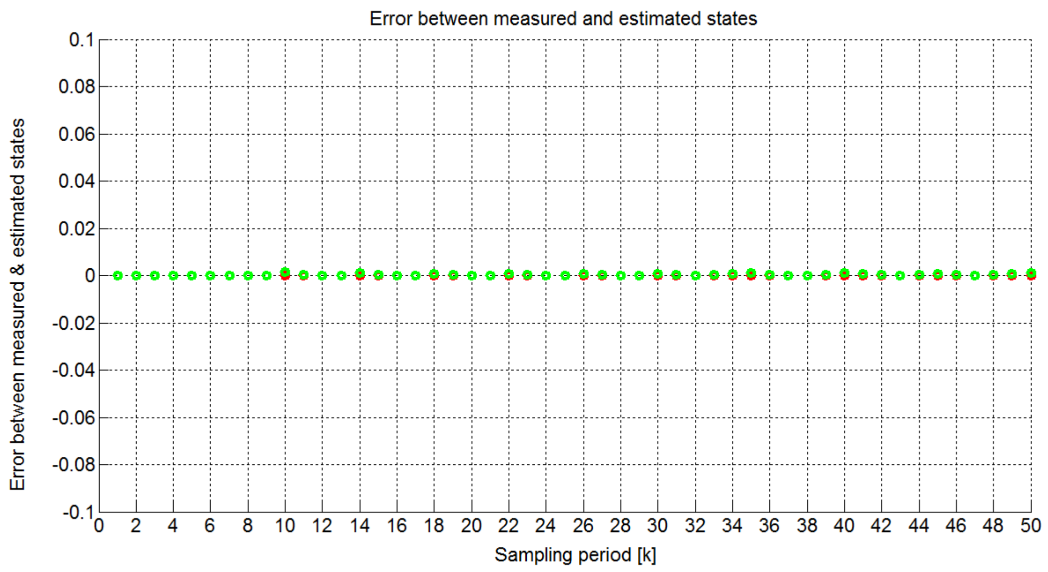


Figure 8.14. Error between process states and estimated states for observer initial conditions of 1.0 and model initial conditions of 1.0 using a constant control input of 1.0.

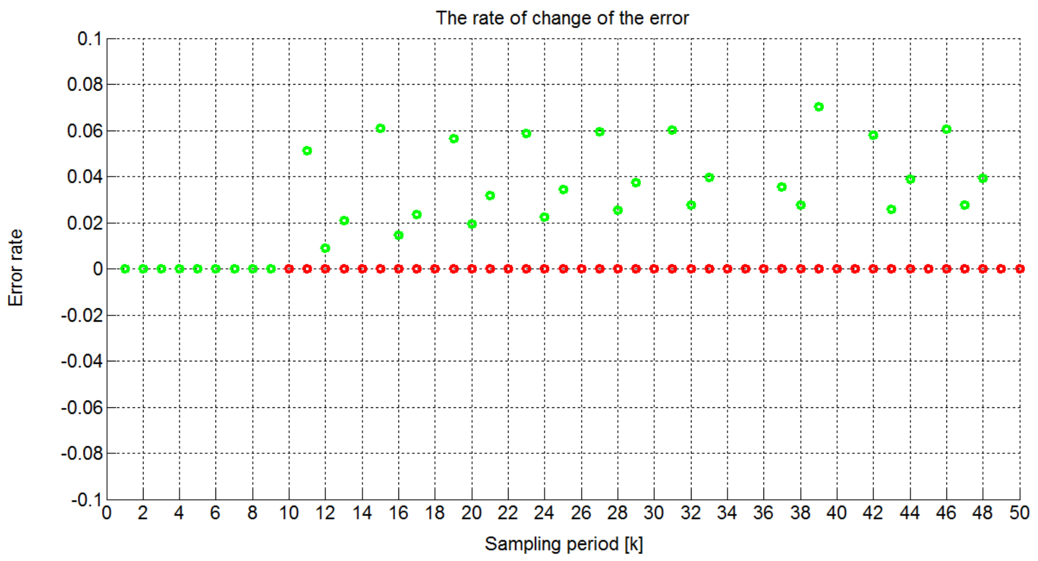


Figure 8.15. Rate of change of the error for observer initial conditions of 1.0 and model initial conditions of 1.0 using a constant control input of 1.0.

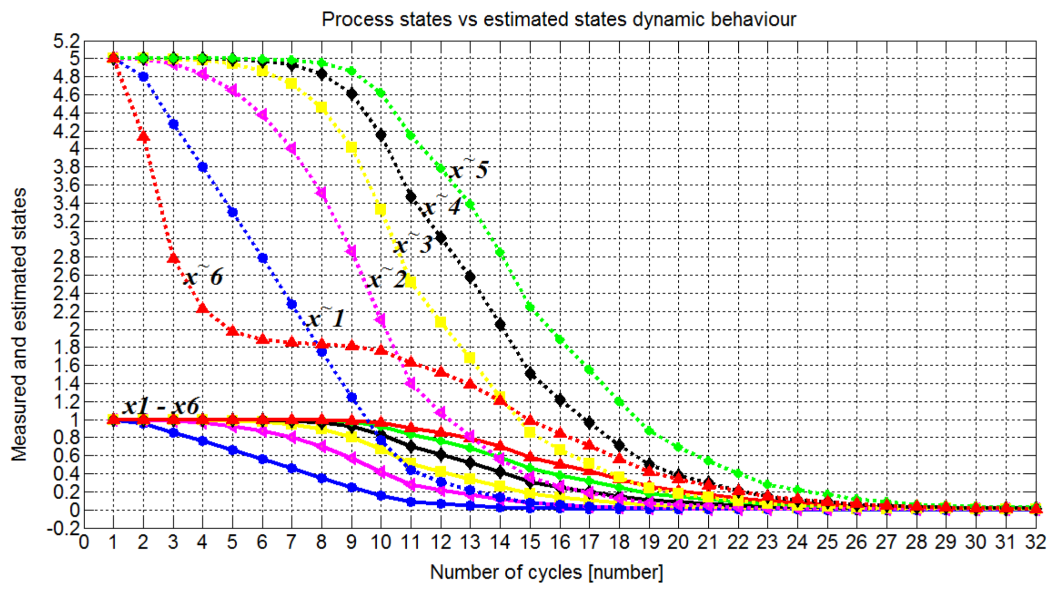


Figure 8.16. Model states and estimated states for observer initial conditions of 5.0 and model initial conditions of 1.0 using a constant input signal of 1.0.

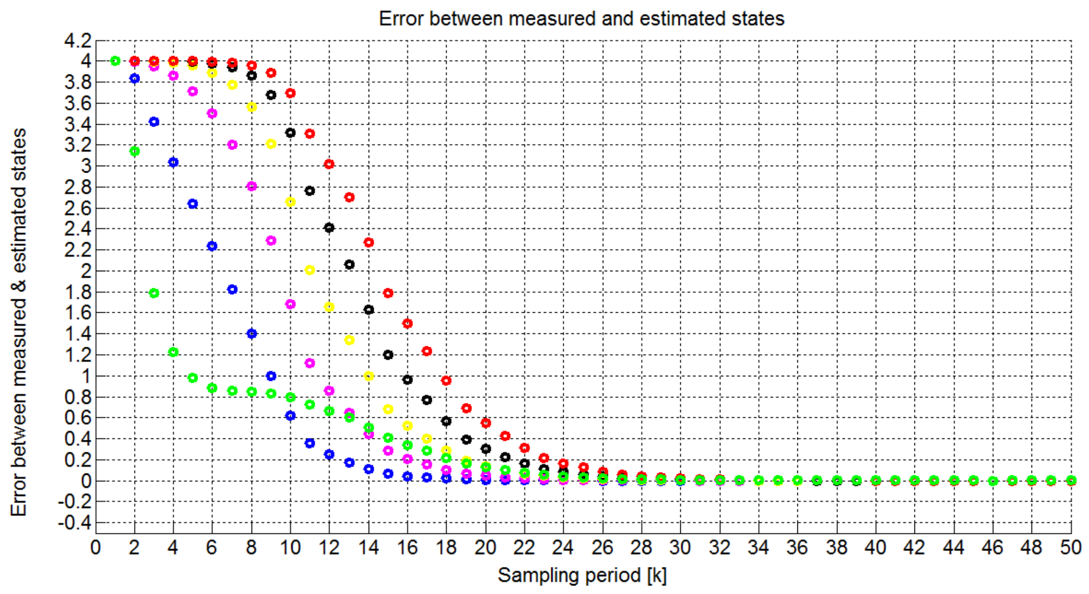


Figure 8.17. Error between process states and estimated states for observer initial conditions of 5.0 and model initial conditions of 1.0 using a constant control input of 1.0.

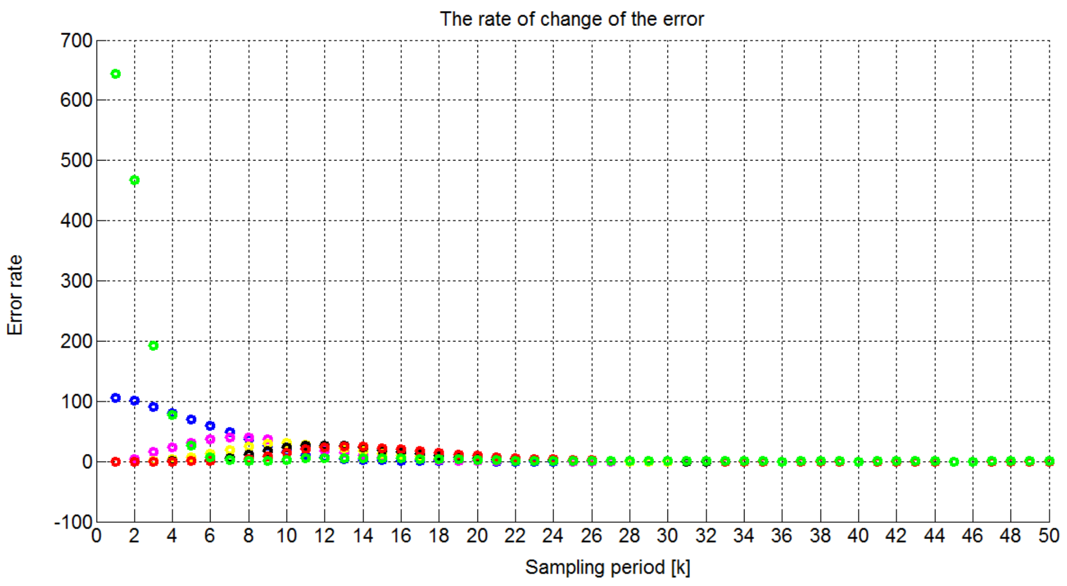


Figure 8.18. Rate of change of the error for observer initial conditions of 5.0 and model initial conditions of 1.0 using a constant control input of 1.0.

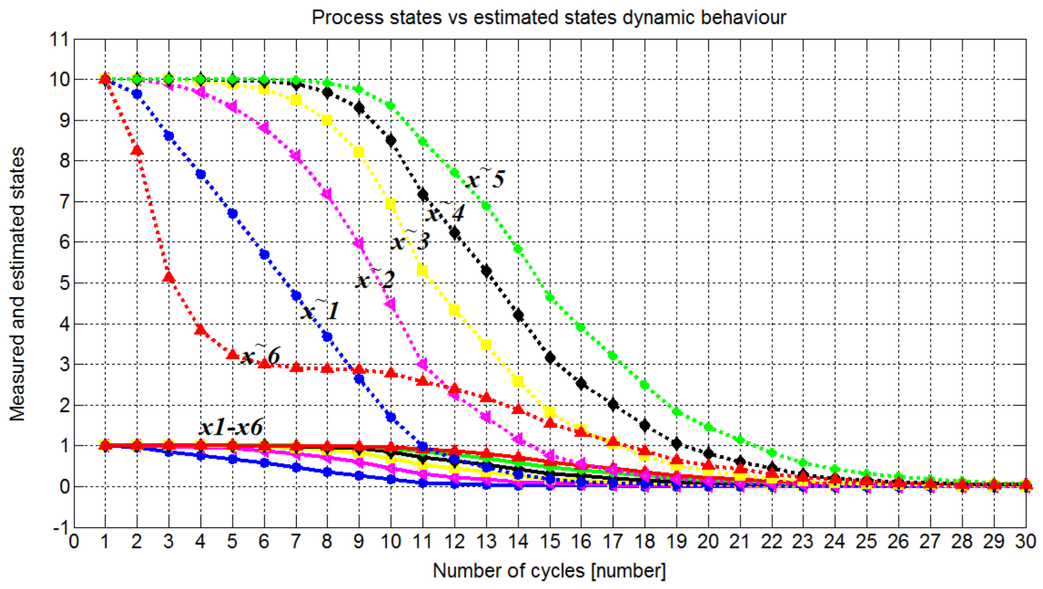


Figure 8.19. Model states and estimated states for observer initial conditions of 10.0 and model initial conditions of 1.0 using a constant input signal of 1.0.

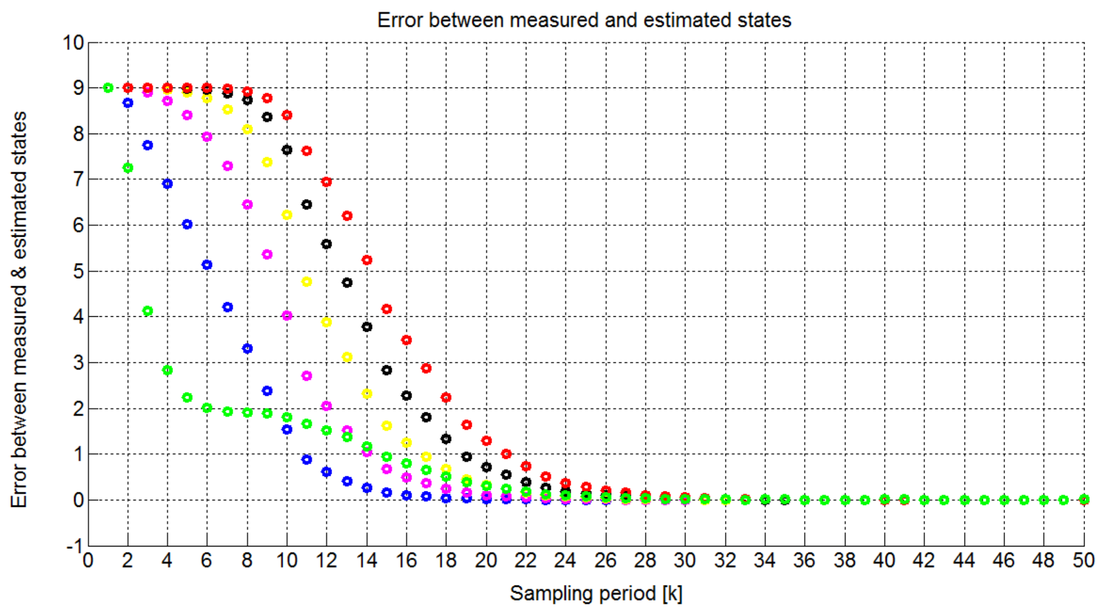


Figure 8.20. Error between process states and estimated states for observer initial conditions of 10.0 and model initial conditions of 1.0 using a constant control input of 1.0.

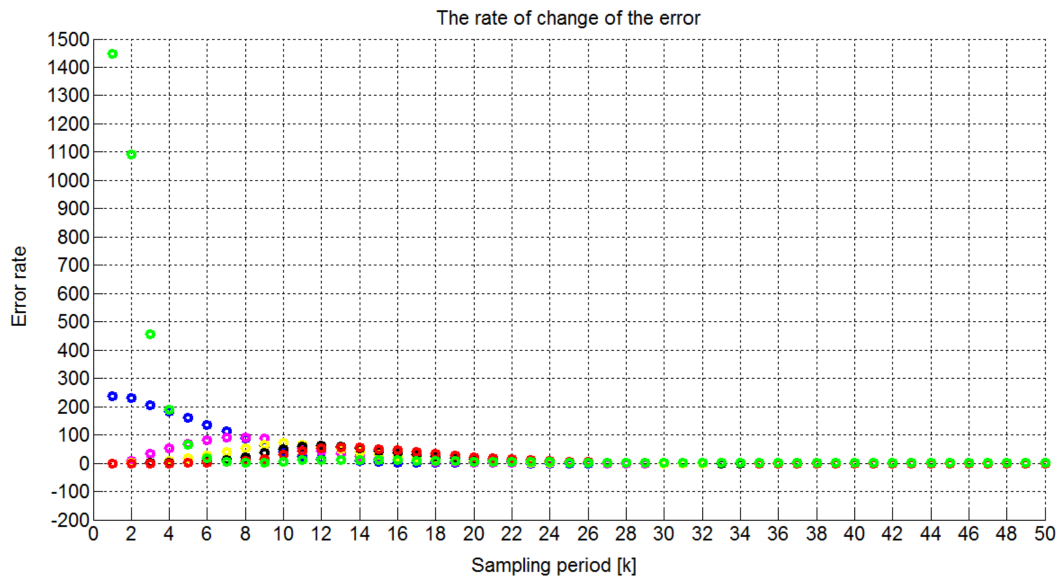


Figure 8.21. Rate of change of the error for observer initial conditions of 10.0 and model initial conditions of 1.0 using a constant control input of 1.0.

These results are based on a system with constant observer gain matrix L. It can be clearly seen from Figures 8.4–8.21 that the observer fully tracks the system and converges at a very short time period around 30 cycles; the ion exchange process is a very slow process. In the case where the observer and the system have the same initial conditions, the observer converges at a shorter period even, just below 20 cycles. This case also shows that the observer is stable. These simulation results clearly show that the observer is performing very well. At higher observer initial conditions, compared to that of the system (Figure

8.4); the observer seems to be unstable in its tracking but does converge in time compared to other cases where the initial conditions are not far from that of the system (Dube & Tzoneva, 2016).

The system response times have also been considered over the changing input signals versus different initial conditions, and these cases are presented by the Tables 8.3–8.5. The system responses of the rise time (Tr), the delay time (Td) and the settle time (Ts) for the first, the third and the sixth stages of the considered CCIX plant are presented. Both the model and the observer (estimated) system responses are presented.

The rise time for almost all data sets per changing input signal seem to be the same time irrespective of the observer initial conditions, and this indicates the stability of the observer. In cases where the values are shown as #NA means that the rise time could not be determined.

Table 8.3: **Rise time (Tr) vs. observer changing initial condition**

Process rise time vs. observer changing initial conditions							
Input Signal	Observer Initial Condition	State x1		State x3		State x6	
		x_1	$\hat{x}_1$	x_3	$\hat{x}_3$	x_6	$\hat{x}_6$
0	-1	9.319	9.319	9.170	9.170	11.42	1.622
0	0	8.846	0	8.940	0	11.42	#NA
0	0.5	8.079	8.079	8.993	8.993	11.42	1.622
0	1	4.061	4.061	4.625	4.625	4.237	4.237
1	-1	9.351	9.290	9.331	9.015	11.56	1.624
1	0	8.870	8.814	9.100	10.22	11.57	#NA
1	0.5	8.111	7.138	9.130	9.264	11.53	8.831
1	1	4.078	4.078	4.693	4.693	4.294	4.294
5	-1	9.483	9.173	10.07	8.638	12.13	1.633
5	0	8.964	8.823	10.02	10.85	12.21	#NA
5	0.5	8.238	8.380	9.847	11.43	12.08	9.550
5	1	4.145	4.145	4.964	6.309	6.300	6.309
10	-1	9.651	9.026	11.61	8.249	12.91	1.643
10	0	9.236	8.841	11.58	10.69	13.13	#NA
10	0.5	8.413	8.850	11.47	#NA	12.96	10.75
10	1	4.231	4.231	5.400	5.400	6.728	6.724
20	-1	10.04	8.863	#NA	7.731	18.56	1.667
20	0	9.783	8.947	#NA	10.89	18.21	#NA
20	0.5	8.723	10.30	13.26	12.86	17.95	17.10
20	1	4.407	4.407	20.96	20.96	20.96	20.96

The settle time for each run varies per initial conditions set; what is noticeable is that for the same observer initial conditions, the settle time increases gradually with the increase of the input signal. The rise time also increases through stages of the column respectively.

The delay time is the same for all the column stages irrespective of the increasing input signal value or changing observer initial conditions.

Table 8.4: **Settle time (Ts) vs. observer changing initial conditions**

Process settle time vs. observer changing initial conditions							
Input Signal	Observer Initial Condition	State x1		State x3		State x6	
		x_1	$\hat{x}_1$	x_3	$\hat{x}_3$	x_6	$\hat{x}_6$
0	-1	16.00	16.00	23.00	22.90	26.50	28.50
0	0	28.20	16.40	40.00	21.00	45.00	42.00
0	0.5	14.30	14.40	19.20	21.20	27.70	36.00
0	1	11.00	16.00	22.00	22.00	0.000	0.000
1	-1	16.20	15.90	24.00	22.50	29.80	28.80
1	0	31.50	21.30	39.70	42.80	46.50	59.00
1	0.5	14.50	14.50	22.30	20.30	27.20	28.00
1	1	11.50	11.50	14.60	14.60	25.50	29.10
5	-1	17.00	15.20	32.00	26.00	36.00	32.00
5	0	39.00	37.50	47.50	38.50	54.00	48.00
5	0.5	15.55	19.30	21.10	21.50	30.40	29.70
5	1	12.00	12.00	21.00	21.00	28.00	28.00
10	-1	16.00	13.40	23.50	18.20	29.80	28.30
10	0	37.50	37.50	41.50	36.00	59.00	59.00
10	0.5	17.74	27.50	21.88	21.60	28.40	35.00
10	1	15.50	15.50	25.60	25.60	26.00	26.00
20	-1	16.00	16.10	23.00	17.80	29.30	28.00
20	0	31.50	25.00	41.50	35.00	58.00	51.00
20	0.5	22.00	14.50	21.70	22.00	35.00	29.80
20	1	11.75	11.75	13.30	13.30	30.00	25.30

Table 8.5: **Delay time (Td) vs. observer changing initial condition**

Process delay time vs. observer changing initial conditions							
Input Signal	Observer Initial Condition	State x1		State x3		State x6	
		x_1	$\hat{x}_1$	x_3	$\hat{x}_3$	x_6	$\hat{x}_6$
0	-1	1.000	1.000	1.000	1.000	1.000	1.000
0	0	1.000	1.000	1.000	1.000	1.000	1.000
0	0.5	1.000	1.000	1.000	1.000	1.000	1.000
0	1	1.000	1.000	1.000	1.000	1.000	1.000
1	-1	1.000	1.000	1.000	1.000	1.000	1.000
1	0	1.000	1.000	1.000	1.000	1.000	1.000
1	0.5	1.000	1.000	1.000	1.000	1.000	1.000
1	1	1.000	1.000	1.000	1.000	1.000	1.000
5	-1	1.000	1.000	1.000	1.000	1.000	1.000
5	0	1.000	1.000	1.000	1.000	1.000	1.000
5	0.5	1.000	1.000	1.000	1.000	1.000	1.000
5	1	1.000	1.000	1.000	1.000	1.000	1.000
10	-1	1.000	1.000	1.000	1.000	1.000	1.000
10	0	1.000	1.000	1.000	1.000	1.000	1.000
10	0.5	1.000	1.000	1.000	1.000	1.000	1.000
10	1	1.000	1.000	1.000	1.000	1.000	1.000
20	-1	1.000	1.000	N/A	1.000	1.000	1.000

20	0	1.000	1.000	1.000	1.000	1.000	1.000
20	0.5	1.000	1.000	1.000	1.000	1.000	1.000
20	1	1.000	1.000	1.000	1.000	1.000	1.000

8.9. Conclusion

The observer design for a bilinear model of the CCIX process has been presented. The observer design has been developed based on real data of the ion exchange process obtained from experiments conducted previously in an ion exchange process of the same type.
Data have been normalized for this exercise and simulation results from SIMULINK and MATLAB showed the design to be competently conclusive. The observer converges even if different initial conditions are applied (Figures 8.4–8.21). The results show that the bilinear type observer is applicable in this type of a bilinear model.

The next chapter discusses state estimation for the CCIX bilinear process model using Kalman filtering. The design of the filter is developed using two methods, 1) the *mean square approach* and 2) the *direct optimization technique*.

CHAPTER NINE

METHODS FOR DESIGN OF A KALMAN FILTER

CHAPTER NINE

9. METHODS FOR DESIGN OF A KALMAN FILTER

9.1. Introduction

This chapter deals with the second method of solution for the state estimation problem developed for the continuous countercurrent ion exchange (CCIX) process. The major challenge in the CCIX control is that the states in the plant were not measurable except for the last one at the output of the plant. The state values are needed for monitoring and control of the process more especially for real-time control strategy. It is therefore necessary to estimate the rest of these states. This chapter is based on two derivations of the filter; the one that uses the common technique of the *mean square approach* and the other that uses *direct optimization* method based on the optimization criterion.

The process model used in implementing the Kalman filter is that of a bilinear system as generated in Chapter 5. A MATLAB software program is developed to simulate the Kalman filter based state estimate solution. Data used to confirm the validity of the developed solution were obtained from the same process (continuous countercurrent ion exchange process) implemented at the University of Cape Town, Cape Town, Western Cape, South Africa in 1982. Results from the software program are presented in the last section of the chapter.

The rest of the chapter is structured in the following manner: the design of the Kalman filter for state estimation; derivation of the Kalman filter, the theory; formulation of the problem of Kalman filter; application of the Kalman filter in the CCIX; the concept of covariance matrices and the filter gain for prediction and correction; two filter design methods are presented, the general formulation and direct optimization; algorithms for the two methods are developed; software program is developed based on the second method; and finally the results from the developed software and discussion of results. The programs listing are in APPENDIX E

9.2. Introduction to the design of the Kalman filter for the state estimation problem

Kalman filter design as one of many state estimation methods that was developed by optimization technique became an important part of estimation theory. One can therefore consider the Kalman filter as one of many optimal state estimation methods (algorithms). According to Siouris (1996:92) one of many advantages of the Kalman filter is that it is equally applicable to both continuous time and discrete time linear systems.

The Kalman filter has since been extended to different applications in many different forms to improve its solution. In this thesis the basic Kalman filter technique is considered, Figure 9.1 (Anderson & Moore, 2005; Bozic, 1979; Sorenson, 1985; Grewal & Angus, 1993; Siouris, 1996). The concept of a basic discrete linear Kalman filter design is extended to the bilinear model that represents the continuous countercurrent ion exchange process (CCIX). The discrete filter model allows optimization techniques to be implemented in estimating the unknown states of the CCIX process.

The Kalman algorithm uses the same equations for *filtering* and for *prediction* solutions. The most important part of the Kalman filter is that it uses *optimization techniques* (statistical approximation) to estimate unknown variables. This makes it the most favourable method in solving state estimation problems (Anderson & Moore, 1979; Meinhold & Singpurwalla, 1983; Mortensen, 1968; Siouris, 1996).

Siouris (1996:92) and Sorenson (1985) describe the Kalman filter as consisting a linearized model of a system dynamics and employing statistical estimates of the system error sources to compute the gains for processing of the external measurements data. The measurement data is then used to generate corrections and improve the system compensation for critical error sources (Bar-Shalom & Xiao-Rong,1993; Bozic, 1979; Dean, 1986; Grewal & Angus, 1993; Siouris 1996; Sorenson, 1985).

According to Romanenko, et al., (2004), Anderson and Moore (1979), and Siouris (1996), the Kalman-type filtering method consists of two major steps: *prediction* and *correction*. In the prediction step a new *a priori* estimate of the state vector is computed using the system model; and in the correction step, the predicted estimate is improved using available measurements (Maybeck, 1996; Kumar, Jerome & Ayyappan, 2013; Romanenko, et al., 2004; Siouris, 1996; Swerling, 1971; Rhodes, 1979).

This filtering method has become increasingly the most commonly used method due to its simplified procedure. The algorithm allows generating estimate of state variables by available measured variable (the output data). According to Siouris (1996:92) the method allows optimized error corrections if the system error dynamics and the associated statistics are accurately modelled in the filter (Straka, Dunik & Simandl, 2014; Shyamalagowri & Rajeswari, 2016).

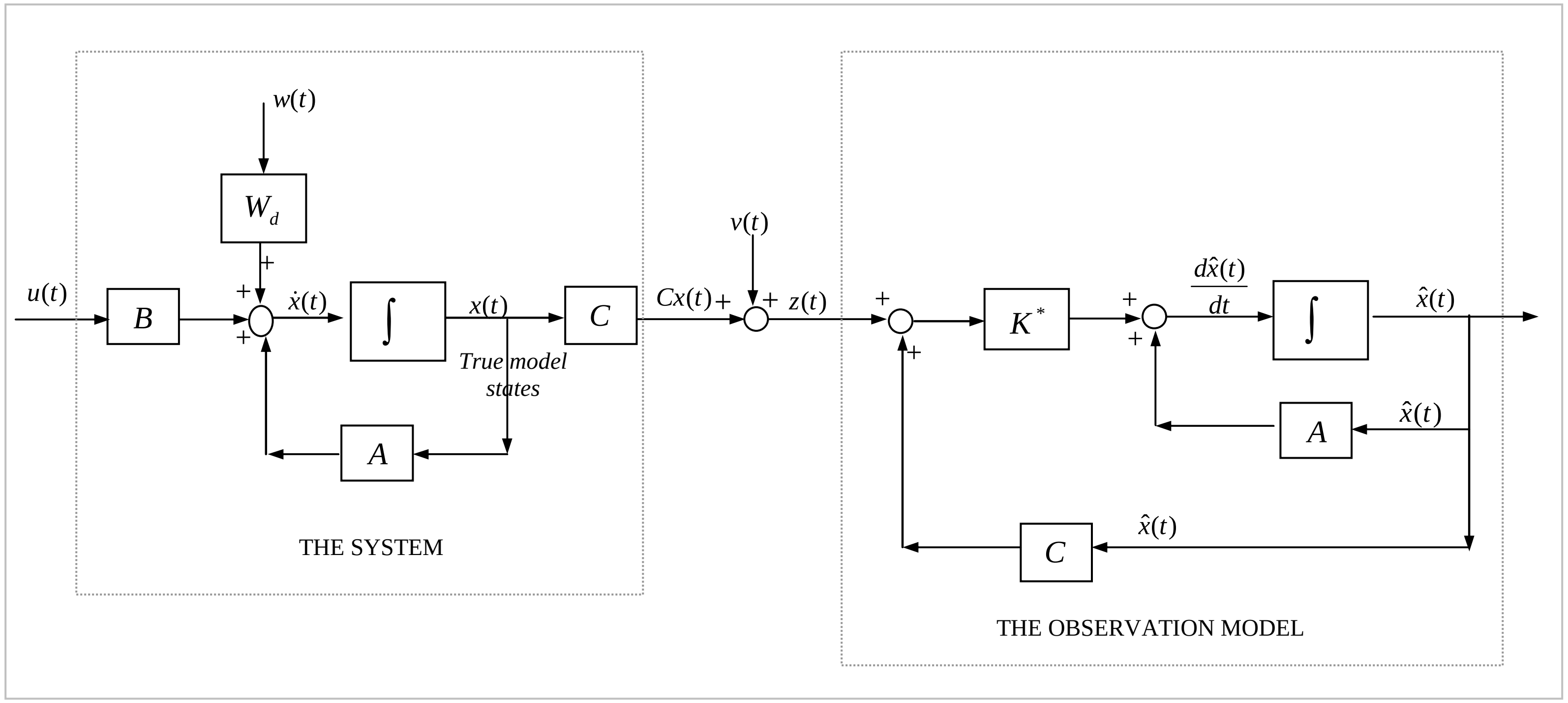


Figure 9.1. Continuous time Kalman filter design structure for a linear system

(Adapted from Siouris, 1996:98)

The filter also allows fine tuning of estimated variables to bring the estimates into agreement with the nominal performance of the system (Jazwinski, 1970; Siouris, 1996; Welch, 2001; Welch & Bishop, 2006).

Kalman filter is designed based on *stochastic processes* theory and *random variable* theory (Lewis, 1986; Maybeck, 1982; Maybeck, 1979). Ikonen and Najim (2002:37) describe this process using parameter estimation comparison. According to Ikonen and Najim (2002:37) in estimation procedures where a parameter of interest is considered a random variable and is determinable using other random variables that are somehow correlated to the parameter in question, this method is called a *Bayesian approach*. Kalman filter design for state estimation takes the same approach (Ikonen & Najim, 2002:37; Meinhold & Singpurwalla, 1983; Julier & Uhlmann, 2004). An unknown observable state vector is assumed to be correlated with the output of the system being observed and based on the output measurements, the values of the unknown state vector are possible to estimate (Ikonen & Najim, 2002:37).

According to Mortensen (1968) the Kalman filter takes different forms for state estimation, in some instances it takes the form of, 1) the *prediction observer* or 2) the *current observer*. The presentation in this text is that of the current observer format.

Let a discrete linear time invariant system be described by the form (Anderson & Moore, 1979; Ikonen & Najim, 2002:37; Mortensen, 1968; Siouris, 1996):

$$x(k+1) = Ax(k) + Bu(k) + w(k) \qquad (9.1)$$

$$z(k) = Cx(k) + v(k)$$

where

$A \in R^{n \times n}$ – is the matrix of system parameters,

$B \in R^{n \times m}$ – is the system input matrix,

$C \in R^{l \times n}$ – is the system output matrix,

$x(k) \in R^{n \times 1}$ – are the state variables,

$u(k) \in R^{m}$ – is the system input variable,

$z(k) \in R^{l}$ – is the system output variable,

$w(k) \in R^{l}$ – is the random process disturbance and,

$v(k) \in R^{l}$ – is the measurement random variables (noise).

According to Mortensen (1968) and Ikonen and Najim (2002:37) the aim of the observer is to estimate the state vector of the system $x(k)$ with the vector $\hat{x}(k)$

for every moment k based on the measured output values $z(k)$ assumed to be containing some noise $v(k)$. The system parameters A and output coefficients C are assumed known; the system noise $w(k)$ and the output noise $v(k)$ are assumed to be *zero mean*, *independent Gaussian* processes with known mean values and covariances (Julier, Uhlmann, & Durrant-Whyte, 1995; Julier, Uhlmann, & Durrant-Whyte, 2000; Jover, & Kailath, 1986; Lefebvre, Bruyninckx, & De Schutter, 2004; Rhodes, 1979; Rowell, 2004; Sorenson, 1970):

$$E\{v(k)\}=0;\ and\ E\{v(k)v(j)^T\}=V^*(k)\delta_{kj} \tag{9.2}$$

$$E\{w(k)\}=0;\ and\ E\{w(k)w(j)^T\}=W^*(k)\delta_{kj} \tag{9.3}$$

$$E\{w(k)v(j)^T\}=0;\ \ and\ E\left\{v(k)w(j)^T\right\}=0 \tag{9.4}$$

where

δ_{kj} – is the Kronecker delta function (Ikonen & Najim, 2002:39; Julier, et al., 1995; Julier, et al., 2000),

$$\delta_{kj}=\begin{cases}1 & if \quad k=j\\ 0 & every\text{where else}\end{cases}$$

$v(k)$ and $w(k)$ – are noises have covariance matrices $V^*(k)$ and $W^*(k)$ respectively, which are non-negative and symmetric (Ikonen & Najim, 2002:39).

The assumption made is that $\{z(k)\}$ is available for measurements but $\{x(k)\}$ is not measurable and the intention is to predict a sequence $\{x(k)\}$ from the measured values of the measurement sequence $\{z(k)\}$. According to Ikonen & Najim (2002:42), "if the disturbances $\{w(k)\}$ and $\{v(k)\}$ as well as the initial state $x(0)$ are Gaussian (with mean values of 0, 0, and x_0 and have covariances $W^*(k)$, and $V^*(k)$ respectively, $P(0)$, the initial error covariance matrix, then the estimate $\hat{x}(k+1)$ is *the mean of conditional distribution* of $x(k+1)$". The solution is mostly based on the measurements and the measured output $z(k)$, the observation model given by (Ikonen & Najim, 2002; Kushner, 1967):

$$z(k)=Cx(k)+v(k) \tag{9.5}$$

The filter is generalized around the use of the observation model, Equation (9.5). The state equation, Equation (9.1), represents the real dynamic activities within a process, through values of the state vector at time k, $x(k)$. The observation model, Equation (9.5) represents the measurement process. It further relates the actual state at the measured output to the system state using the output matrix, $C \in R^{l \times n}$, Equations (9.1). The estimation error of the process is given by Equation (9.6) (Julier, et al., 1995; Lefebvre, et al., 2004; Siouris, 1996; Sorenson, 1970):

$$e(i|k) = x(i) - \hat{x}(i|k) \tag{9.6}$$

where $x(i)$ – is the state at the i^{th} time sample,

$\hat{x}(i|k)$ – is the estimate of the state at the i^{th} sample based on the k^{th} sample".

Using the argument in Equation (9.1), the estimation process can be defined as: *filtering*, *smoothing* or *extrapolation*:

1) If $i = k$, the process is known as *filtering*,
2) for $i < k$, the model represents *smoothing* and
3) if $i > k$, the process is known as *extrapolation*.

The estimated state is equal to the predicted estimate from the previous moment in k plus the error difference between what should have been observed compare to what was observed, multiplied by the filter gain. The Kalman filter gain determines, by how much should the current estimate be changed based on the new observation. The Kalman filter gain changes with time and should be kept at its optimal value. This is achievable by introducing optimality criterion in the filtering procedure design. It is also highly dependent on the characterization of the measured noise, $v(k)$.

9.3. Derivation of the Kalman filter for the state estimation problem (the necessary terms and theory for the procedure)

In designing the Kalman filter for the solution of the state estimation problem, there are few steps that are necessary to complete, so as to achieve an optimized filter (Anderson & Moore, 1979; Anderson, & Moore, 2005; Gelb, 1974; Rowell, 2004). The design procedure includes an optimization step which is an integrated part of the filter design, and this makes the Kalman filter and its multiples of variations the most attractive and commonly used method for state estimation in process control problems (Grewal & Angus, 1993; Rowell, 2004).

The Kalman filter has two important steps of the estimation solution, i.e., the *Time Update* step (which is the initial estimation of the state at a moment k) and the *Measurement Update* step, (used as a correction term of the previous estimate using measured output value, applied at a moment $(k+1)$. It is therefore important to distinctly specify calculations that are necessary for the first step and those for the second step because if these are not correctly applied one may end up with incorrect estimates that may still work as a solution but not optimal to the best of the filter's abilities (Cao et al., 2016; Chatzis, Chatzi & Triantafyllou, 2017; Rowell, 2004; Romanenko, et al., 2004).

The Kalman filter design method can be considered as calculating the Kalman filter gain matrix used as a correction term in the estimation process. The Kalman filter gain matrix is calculated using a difference between the variables used for estimation; i.e., in this case, the real state $x(k)$ and the estimated state $\hat{x}(k)$.

Firstly, it is very important to specify if a calculation is done at the *Time Update* or *Measurement Update* moment and with respect to which moment in time, k or $(k+1)$ and or $(k-1)$ or k. These terms are defined as either being *a priori* or *a posteriori* to the calculation of the estimate. The set of equations in the *Time Update* moment are for the *prediction* part of the estimation and those at the *Measurement Update* are for the *correction* part of the estimation, Figure 9.2 (Kumar, et al., 2013; Rowell, 2004; Welch & Bishop, 2006).

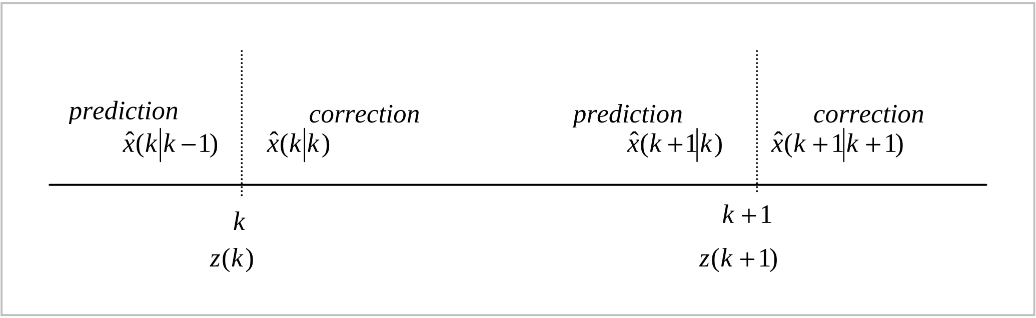


Figure 9.2. **Prediction and correction steps of the procedure for predicting unknown states of a system (Ikonen & Najim, 2002; Siouris, 1996)**

Ikonen and Najim (2002:39) suggest that there are a number of methods that can be used to derive the Kalman filter gain, but the *mean square approach* is the common one and thus, this method is followed here to indicate how the procedure and calculations of the mentioned terms are obtained. Later this method is extended specifically for use as one of the methods to solve the state estimation problem of the developed bilinear system.

There are three most important derivation for the Kalman filter design in a general sense, the *Kalman filter gain matrix*, $K^{*}(k)$, the *state estimate*, $\hat{x}(k+1)$, and the probability of the estimation error, known as the *covariance matrix of the prediction error* $P(k+1)$ (Julier, et al., 1995; Kumar, et al., 2013). The prediction error between the two state variables, Equation (9.7) forms the basis of the estimation procedure,

$$e(k+1) = \hat{x}(k+1) - x(k+1) \tag{9.7}$$

where

$x(k+1)$ – is the current state value at the next moment of $k+1$ based on the previous estimation,

$\hat{x}(k+1)$ – is the predicted state value based on measured output for the next moment of k.

The estimates are given by two different equations, that at the start of the procedure (the *prediction stage*) using the model, Equation (9.8) and then that at the *correction stage* given by the model, plus the Kalman gain multiplied by the error difference between the measured output $z(k) = Cx(k)$ and that calculated from the estimated state, Equation (9.10).

$$\begin{aligned} \hat{x}(k+1) &= A\hat{x}(k) + Bu(k) \\ e(k) &= Cx(k) - C\hat{x}(k) \end{aligned} \tag{9.8}$$

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K^*(k)e(k) \tag{9.9}$$

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K^*(k)\left[Cx(k) - C\hat{x}(k)\right] \tag{9.10}$$

where $K^*(k)$ is the Kalman filter gain at the moment k.

The equation at the start of the procedure uses an initial state estimation value which is then corrected at the correction stage by the second state estimate equation. Equation (9.10) is used for derivation of the Kalman gain based on *posteriori error* $e(k+1)$.

The requirements for the Kalman filter are, that:

- o the filter must be *stable*,
- o be *sensitive to the changes* in the measurements,
- o to have *fast convergence* towards the real state space variables and
- o to have *small amount of computations* for implementation of the algorithm.

9.3.1. Formulation of the discrete Kalman filter problem for a linear discrete time system

The aim is to determine the covariance matrix of the prediction error that will minimize the mean square criterion $J(k+1)$ determined by,

$$J(k+1) = E\{e(k+1)e(k+1)^T\} \tag{9.11}$$

The optimization problem can be formulated as follows: the aim is to *minimize the prediction error* $e(k+1)$ such that the system defined by the Kalman filter gain matrix, is optimized. This requires that the Kalman filter gain matrix be calculated before determining the optimization solution.

The gain matrix is determined as follows; using *the correction error equation*, Equation (9.10) and the *state prediction equation*, Equation (9.7); the prediction error $e(k+1)$ is expressed as:

$$\begin{aligned} e(k+1) &= \left[A\hat{x}(k) + K^*(k)[Cx(k) - C\hat{x}(k)] + Bu(k)\right] - x(k+1) = \\ &= [A - K^*(k)C]\hat{x}(k) + K^*(k)z(k) + Bu(k) - \left[Ax(k) + Bu(k) + w(k)\right] = \\ &= [A - K^*(k)C]\hat{x}(k) + K^*(k)z(k) - Ax(k) - w(k) \end{aligned} \quad (9.12)$$

Substituting for the output $z(k)$ and rearranging the Equation (9.12) results in,

$$\begin{aligned} e(k+1) &= [A - K^*(k)C]\hat{x}(k) + K^*(k)[Cx(k) + v(k)] - Ax(k) - w(k) = \\ &= [A - K^*(k)C]\hat{x}(k) - [A - K^*(k)C]x(k) + K^*(k)v(k) - w(k) \end{aligned} \quad (9.13)$$

Finally,

$$\begin{aligned} e(k+1) &= [A - K^*(k)C](\hat{x}(k) - x(k)) + K^*(k)v(k) - w(k) = \\ &= [A - K^*(k)C]e(k) + K^*(k)v(k) - w(k) \end{aligned} \quad (9.14)$$

Based on the prediction error equation, Equation (9.14), the cost function can now be expressed as follows:

$$\begin{aligned} J(k+1) &= E\left\{ \begin{matrix} \left[[A - K^*(k)C]e(k) + K^*(k)v(k) - w(k)\right] \times \\ \times \left[[A - K^*(k)C]e(k) + K^*(k)v(k) - w(k)\right]^T \end{matrix} \right\} = \\ &= E\left\{[[A - K^*(k)C]e(k)][[A - K^*(k)C]e(k)]^T\right\} + \\ &+ E\left\{K^*(k)v(k)[A - K^*(k)C]e(k)\right\} - E\left\{w(k)[A - K^*(k)C]e(k)\right\} + \\ &+ E\left\{[A - K^*(k)C]e(k)K^*(k)v(k)\right\} + E\left\{\left(K^*(k)v(k)\right)\left(K^*(k)v(k)\right)^T\right\} - \\ &- E\left\{w(k)K^*(k)v(k)\right\} - E\left\{[A - K^*(k)C]e(k)w(k)\right\} - \\ &- E\left\{K^*(k)v(k)w(k)\right\} + E\left\{\left(w(k)\right)\left(w(k)\right)^T\right\} \end{aligned} \quad (9.15)$$

From Equation (9.4) $E\{w(k)v(k)\} = 0$, the cost function can be expressed as

$$\begin{aligned} J(k+1) = E&\left\{\left[[A - K^*(k)C]e(k)\right]\left[[A - K^*(k)C]e(k)\right]^T\right\} + \\ &+ E\left\{K^*(k)v(k)[A - K^*(k)C]e(k)\right\} - E\left\{w(k)[A - K^*(k)C]e(k)\right\} + \\ &+ E\left\{[A - K^*(k)C]e(k)K^*(k)v(k)\right\} + E\left\{\left(K^*(k)v(k)\right)\left(K^*(k)v(k)\right)^T\right\} - \\ &- E\left\{[A - K^*(k)C]e(k)w(k)\right\} + E\left\{\left(w(k)\right)\left(w(k)\right)^T\right\} \end{aligned} \quad (9.16)$$

Given that $w(k)$, $v(k)$ and $e(k)$ are statistically independent; and $K^*(k)$ and $[A - K^*(k)C]$ are known, then

$$\begin{aligned} J(k+1) &= [A - K^*(k)C]E\left\{e(k)e(k)^T\right\}[A - K^*(k)C]^T + \\ &+ E\left\{K^*(k)v(k)v(k)^T K^*(k)\right\} + E\left\{w(k)w(k)^T\right\} \end{aligned} \quad (9.17)$$

From Equations (9.2) and (9.3), $E\left\{w(k)w^T(k)\right\}=W^*(k)$ and $E\left\{v(k)v(k)^T\right\}=V^*(k)$ given that $\delta_{kj}=\begin{cases}1 & if \quad k=j\\ 0 & \text{everywhere} \quad else\end{cases}$ Equation (9.17) simplifies to

$$J(k+1)=[A-K^*(k)C]E\left\{e(k)e(k)^T\right\}[A-K^*(k)C]^T+ \\ +W^*(k)+K^*(k)V^*(k)K^*(k)^T \tag{9.18}$$

The following assumptions are made in connection with Equation (9.18) $w(k)$ and $v(k)$ are statistically independent; $e(k)$ is given by $e(k)=\hat{x}(k)-x(k)$; the state prediction $\hat{x}(k)$ depends on the past output measurement $z(k-1)$ and thus on $v(k-1)$; the current calculated state $x(k)$ depends on the system noise $w(k-1)$ disturbing the state, not dependent on the current disturbances and noise of $w(k)$ and $v(k)$ respectively; and thus $e(k)$, $v(k)$ and $w(k)$ are statistically independent.

$$P(k)=E\left\{e(k)e(k)^T\right\}=E\left\{[x(k)-\hat{x}(k)][x-\hat{x}(k)]^T\right\} \tag{9.19}$$

where $P(k)$ is the covariance matrix of the estimation error $e(k)$

Therefore, the criterion is expressed by the covariance matrices of the error, and the process and measurement noises.

$$J(k+1)=[A-K^*(k)C]P(k)[A-K^*(k)C]^T+W^*(k)+K^*(k)V^*(k)K^*(k) \tag{9.20}$$

9.3.2. Derivation of the Kalman filter gain matrix

In Equation (9.20) the Kalman gain is unknown. It is determined on the basis of the necessary conditions for optimization of Equation (9.20). It is considered in terms of the *estimation error covariance matrix*, $P(k)$ and is rewritten as follows (Ikonen & Najim, 2002:41):

$$J(k+1)=[A-K^*(k)C]P(k)[A-K^*(k)C]+W^*(k)+K^*(k)V^*(k)K^*(k)^T \tag{9.21}$$

$$J(k+1)=AP(k)A^T-K^*(k)CP(k)A^T-AP(k)K^*(k)^TC^T+ \\ +K^*(k)CP(k)K^*(k)^TC^T+W^*(k)+K^*(k)V^*(k)K^{*T}(k)= \\ =AP(k)A^T-\left[K^*(k)CP(k)A^T+AP(k)K^*(k)^TC^T\right]+ \\ +K^*(k)\left[CP(k)C^T+V^*(k)\right]K^*(k)^T+W^*(k) \tag{9.22}$$

The following notation can be introduced,

$$R(k)=W^*(k)+CP(k)C^T \tag{9.23}$$

$$\begin{aligned}J(k+1) = AP(k)A^T + W^*(k) + K^*(k)R(k)K^*(k)^T - \\ -\left[K^*(k)CP(k)A^T + AP(k)K^*(k)^T C^T\right]\end{aligned} \tag{9.24}$$

The last two terms of the Equation (9.24) that contain $K^*(k)$ can be grouped as $J_K(k+1)$ and be solved by completing a square of a quadratic equation; let this new term be defined by,

$$\begin{aligned}J_K(k+1) = K^*(k)R(k)K^*(k)^T - 2\left[K^*(k)CP(k)A^T\right] = \\ = \left[K^*(k)K^*(k)^T - 2\left[K^*(k)CP(k)A^T\right]R^{-1}(k)\right] R(k) = \\ = \left[K^*(k) - \frac{2CP(k)A^T R^{-1}(k)}{2}\right]^2 R(k) - \left[K^*(k) - \frac{2CP(k)A^T R^{-1}(k)}{2}\right]^2 R(k)\end{aligned} \tag{9.25}$$

And this simplifies to

$$\begin{aligned}J_K(k+1) = \left[K^*(k) - CP(k)A^T R^{-1}(k)\right]Q(k)\left[K^*(k) - CP(k)A^T R^{-1}(k)\right]^T + \\ -\left[CP(k)A^T\right]R(k)\left[CP(k)A^T\right]^T\end{aligned} \tag{9.26}$$

Finally,

$$\begin{aligned}J(k+1) = AP(k)A^T + W^*(k) - \left[CP(k)A^T R(k)C^T P(k)^T A\right] + \\ + \left[K^*(k) - AP(k)C^T R^{-1}(k)\right]R(k)\left[K^*(k) - AP(k)C^T R^{-1}(k)\right]^T\end{aligned} \tag{9.27}$$

From Equation (9.27) it can be seen that only the last term of the equation depends on $K^*(k)$, the optimization criterion requires that the cost function be minimized such that all the values' expression of terms that contain $K^*(k)$ go to zero, $\left[K^*(k) - AP(k)C^T R^{-1}(k)\right]R(k)\left[K^*(k) - AP(k)C^T R^{-1}(k)\right]^T \to 0$; this implies that

$$K^*(k) - AP(k)C^T R^{-1}(k) = 0 \tag{9.28}$$

From this necessary condition of optimality, the gain matrix equation can be derived as follows:

$$K^*(k) = AP(k)C^T R^{-1}(k) = AP(k)C^T \left[W^*(k) + CP(k)C^T\right]^{-1} \tag{9.29}$$

Substituting the gain matrix Equation (9.29) back to the criterion, Equation (9.27) gives

$$J(k+1) = AP(k)A^T + W^*(k) - [K^*(k)CP(k)A^T] \tag{9.30}$$

According to Ikonen and Najim (2002:42), for Kalman filter algorithm to receive an *optimal estimate* in terms of the *mean square error*, the next state $x(k+1)$ estimated on the available output measurements up to k moment is based on the following equations:

1) *the filter gain matrix*

$$K^*(k) = AP(k)C^T\left[W^*(k) + CP(k)C^T\right]^{-1} \tag{9.31}$$

2) *the next state estimate*

$$\hat{x}(k+1) = A\hat{x}(k) - K^*(k)[C\hat{x}(k) - z(k)] \tag{9.32}$$

3) *the cost function* based on covariance of the estimation error,

$$J(k+1) = AP(k)A^T + W^*(k) - K^*(k)P(k)A^T \tag{9.33}$$

Though, the Kalman filter general design for a linear discrete time system has been presented in the simplest form and derivation of terms necessary for the implementation of the filter covered; in this form the filter is not implementable. It is necessary that the predictor part equations be developed separately and the correction part equations also be derived based on the predictor equations, for the full optimal filter presentation.

In the next two sections of this text, two different methods for deriving the filter design procedure are shown for the considered bilinear process of water desalination, and then followed by the algorithm for the implementation of the filter.

9.4. The Kalman filter gain design methods for the bilinear model of the continuous countercurrent ion exchange (CCIX) process

9.4.1. Ion exchange model with process and measurement noises

A bilinear model for the continuous countercurrent ion exchange process has been developed and it is considered to have two types of disturbances: 1) *deterministic* and 2) *stochastic* noises. The deterministic noise in the system is the noise that comes from the liquid inflow concentration $x_f(k)$ and the stochastic noises are the process noise in the model $w(k)$, and the output measurement noise, $v(k)$. The system to be estimated is represented by the bilinear model of the form

$$x(k+1) = Ax(k) + B_1x(k)u(k) + Bu(k) + Wx_f(k) + w(k) \tag{9.34}$$

$$z(k) = Cx(k) + v(k)$$

$$x(0) = x_0 \tag{9.35}$$

where

$x(k) \in R^{n\times l}$ – is the state vector of the system,

$u(k) \in R^{l\times m}$ – is the vector of system inputs,

$z(k) \in R^{l}$ – is the output of the system,

$x_f(k)$ – is the deterministic disturbance of the system, the inflow concentration, (concentration entering the first stage of the process column,)

$w(k) \in R^l$ – is the system disturbance with Gaussian distribution,

$v(k) \in R^l$ – is the output measurement noise with Gaussian distribution,

$A \in R^{n \times n}$ – the state transition matrix,

$B_1 \in R^{n \times n}$ – the state transition matrix with influence from inputs (bilinear transition matrix of the system states and inputs)

$B \in R^{n \times l}$ – the control input transition matrix

$C \in R^{l \times n}$ – is the system output matrix

$W \in R^{n \times l}$ – is the disturbance transition matrix.

The model is different from that considered in the other chapters of this document in that the internal noise and the output measurement noise were assumed to be of no great influence in those solutions, thus not considered; but for the state estimation solution using the Kalman filter it is important that all the noise components be identified.

The system contains two types of noise processes; the deterministic noise process and the stochastic noise process. The stochastic noise component is from two noise sources; the system noise $w(k)$ and the measurement noise $v(k)$. The stochastic noise variables $w(k)$ and $v(k)$ are considered independent of each other and are represented by white noise processes. These variables have *probability distributions* $P(w)$ and $P(v)$. These two distributions are given by:

$$P(w(k)) = N(0, W^*) \text{ , or} \tag{9.36}$$

$$E\{w(k)w(j)^T\} = \begin{cases} 0, & k \neq 0 \\ W^*(k), & k = j \end{cases} \tag{9.37}$$

$$P(v(k)) = N(0, V^*) \text{ , or} \tag{9.38}$$

$$E\{v(k)v(j)^T\} = \begin{cases} 0, & k \neq j \\ V^*(k), & k = j \end{cases} \tag{9.39}$$

where W^* is the process noise covariance and V^* is the measurement noise covariance matrices and are assumed constants.

The requirement for the noise characteristics is to preserve the diagonal structure of the covariance matrices. Since the noise variables are considered not correlated, the following expression holds true;

$$E\{v(k)w(k)^T\} = 0, \ f\text{or all } k \tag{9.40}$$

The initial state is also considered to be a normal (Gaussian) distribution and has a mean value of $x(0) = x_0$. Also, the system stochastic noises, $w(k)$ and $v(k)$ have normal (Gaussian) distribution determined by, $\overline{w}(k) = 0$ and $\overline{v}(k) = 0$ respectively.

At the start of the calculation procedure, the following variables must be known for the entire transition period, $k = \overline{1:K-1}$, i.e., $k = k_0, k_0+1, k_0+2, ..., j \le k$, values of transition matrices, A, B, C, and W, and the covariances:

$$E\{e(k)w(k)^T\} = 0 \tag{9.41}$$

$$E\{e(k)v(k)^T\} = 0\,, \tag{9.42}$$

$$x(k_0) = x_{k_0} \text{ and } \operatorname{cov}\{x(k_0)\} = P_{k_0} \tag{9.43}$$

For the design of the full filter procedure the following terms must be identified:

1) the initial estimate at k th moment, $\hat{x}(k)$,
2) the covariance matrix of the estimation error between the estimate and the real value based on measured output at k th moment, $P(k)$,
3) the filter gain matrix at the $(k+1)$ moment, $K^*(k+1)$, and the covariance matrix of estimation error at the $(k+1)$ moment, $P(k+1)$.

The Kalman gain matrix takes the centre stage in the design of the filter because it links the two steps in the procedure of the solution of the estimation problem.

9.4.2. Kalman filter gain design methods developed in the thesis

There are a number of methods for deriving a Kalman filter gain to solve a state estimation problem of a process. In this text, two methods are considered:

1) general formulation of *mean square approach* and
2) direct *optimization approach*.

Each of these methods follows the same implementation procedure. The Kalman filter design is based on two sets of equations: the *Time Update equations* (for prediction) and the *Measurement Update equations* (for correction). The idea is to start with *Time Update* (which is used as a prediction of the state at a moment k) and then the *Measurement Update* (which is used as a correction of the estimates using measured output value). The *Measurement Update* equation is responsible for the feedback of the noisy measurements through the covariance matrix of the prediction error (Jover & Kailath, 1986; Kumar, et al., 2013; Maybeck, 1979; Swerling, 1971; Rhodes, 1979; Xiong, Wei & Liu, 2012).

The two steps connect two consecutive points in the time domain as shown in the diagram (Figure 9.3) (at k th moment, with k the current step and $(k-1)$ the previous step. At $(k+1)$ moment, $(k+1)$ is the current moment and k defines the previous moment.

The method of solution requires the knowledge of the probability matrix of the estimation error at the k th moment, known as *a priori covariance matrix* $P(k+1|k)$ in the *Time Update* moment; and then, the probability of the estimation error at the k th moment for error correction is done during the *Measurement Update* moment, $k+1$. This probability matrix is known as *a posteriori covariance matrix* of the error, and it is given by $P(k+1|k+1)$. It is important to connect these probability matrices to obtain optimal state estimates, and this is achieved by using an optimal gain matrix, $K^*(k+1)$ (Kumar, et al., 2013; Lefebvre, et al., 2004; Ikonen & Najim, 2002; Siouris, 1996).

Both the considered methods incorporate optimal solutions in their procedures. The first method uses *completion of a square* to the $K^*(k+1)$, gain matrix of the criterion function to determine the optimal value of the filter gain; and the second method determines the optimal filter gain matrix by *differentiating the criterion functional* with respect to the $K^*(k+1)$ gain matrix, and then solving the obtained equation of the first derivative for the filter gain.

One of the most important assumptions in developing the Kalman filter is that, it is assumed that the behaviour of the process prior to the $(k+1)$ moment is available (Figure 9.3). The estimate $\hat{x}(k+1|k) \in R^n$ is defined as *a priori state estimate* and it is determinable from the available information about the process at the moment k.

Two possible solutions are developed in the thesis, for the derivation of the Kalman filter gain, 1) the *general formulation*, and 2) the *direct optimization method*. The first method is dependent on the output measurement at the current k^{th} moment $z(k) = Cx(k) + v(k)$, and the second method is based on the output measurement at the $(k+1)$ moment, $z(k+1) = Cx(k+1) + v(k+1)$.

Both solutions are based on the *prediction* and *correction* using error covariance matrix formulations, Figure 9.3 (Gelb, 1974; Lefebvre, et al., 2004; Rhodes, 1979; Sorenson, 1970).

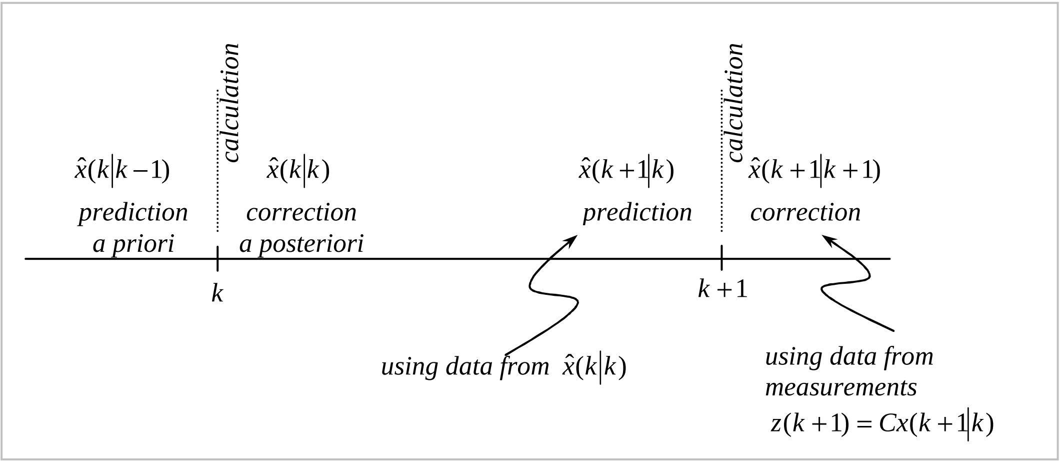


Figure 9.3. Prediction and correction steps of the procedure for predicting unknown states in a system used for the developed methods

9.5. Derivation of the discrete Kalman filter gain for the bilinear model of a continuous countercurrent ion exchange process (CCIX) – general formulation

At the beginning of the procedure, it is assumed that the behaviour of the process prior to the moment $k+1$ is known. The estimate $\hat{x}(k+1|k) \in R^n$ is defined as a *priori state estimate* as it is determined on the basis of the process available information. The *a priori* estimate at the moment $(k+1|k)$ assumes that the *a posteriori estimate* defined by $\hat{x}(k|k)$ is known, for it to be calculated. At the start of the procedure, this *a posteriori* estimate is given by initial conditions, where $\hat{x}(k|k)=\hat{x}(k_0|k_0)$. The *a priori estimate* at a moment $(k+1|k)$ is defined by:

$$\hat{x}(k+1|k) = A\hat{x}(k|k) + B_1\hat{x}(k|k)u(k) + Bu(k) + Wx_f(k) \tag{9.44}$$

The most important part of the Kalman filter for use in state estimation is the connection between the *a priori* and the *a posteriori* estimations, which is provided by the available output measurement at the next sampling moment, $(k+1)$, Figure 9.3. This connection is provided by supposition that the *a posteriori* estimate at the next moment of time, $\hat{x}(k+1|k+1)$ is calculated on the basis of the *a priori* estimate and added to it the value of the Kalman filter gain multiplied by the difference between the actual measured output and the output estimated from the state estimate (Ikonen & Najim, 2002:40–42)

$$z(k) = Cx(k|k) + v(k) \tag{9.45}$$

The *a priori error of estimation* is determined by the difference between the measured states and the estimated states at the moment $(k+1|k)$,

$$e(k+1|k) = \hat{x}(k+1|k) - x(k|k) \tag{9.46}$$

The *a posteriori estimate* is given by:

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K^{*}(k)[z(k) - C\hat{x}(k|k)] \tag{9.47}$$

This difference between the output terms $[z(k) - C\hat{x}(k|k)]$ is called *measurement innovation* or a *residual*. The measurement innovation is used to improve the estimate. Together with the Kalman filter gain, this difference is used to minimize the criterion such that optimal estimates are obtained (Ikonen & Najim, 2002).
The covariance matrix of the error at the *Time Update* is described using the prediction estimate $\hat{x}(k+1|k)$; it is known as the *a priori estimate error covariance matrix*, defined by $P(k+1|k)$. It is expressed by the following two equations (Lefebvre, et al., 2004; Sorenson, 1970; Welch & Bishop, 2006):

$$P(k+1|k) = E\left\{e(k+1|k)e(k+1|k)^{T}\right\}, \text{ for the moment } (k+1|k) \tag{9.48}$$

$$P(k|k-1) = E\left\{e(k|k-1)e(k|k-1)^{T}\right\}, \text{ for the moment } (k|k-1) \tag{9.49}$$

For optimality it is necessary that the Kalman filter gain matrix be determined such that the *a posteriori* covariance of the estimation error at moment $(k+1)$ is minimized,

$$P(k+1|k+1) \to \min \qquad (9.50).$$

The covariance matrix of the prediction error $P(k+1|k)$ is used to update the time (*prediction*) using $\hat{x}(k+1|k)$ and then as a *correction* at measurement update based on optimal values of *a posteriori* estimate $\hat{x}(k+1|k+1)$ at the moment $(k+1|k+1)$.

The error at the moment $(k+1|k+1)$ is given by Equation (9.51). This error is for the *a posteriori* covariance matrix and criterion calculation, Equation (9.52)

$$e(k+1|k+1) = \hat{x}(k+1|k+1) - x(k+1) \tag{9.51}$$

The error is calculated based on the availability of the output measurement at the moment $(k|k)$. Since the output measurement is done at the moment $(k|k)$, it is possible to express the error $e(k+1|k+1)$ directly as follows:

$$
\begin{aligned}
e(k+1|k+1) &= \hat{x}(k+1|k+1) - x(k+1) = \\
&= \hat{x}(k+1|k) + K^*(k)\left[z(k) - C\hat{x}(k|k)\right] = \\
&= \begin{bmatrix} A\hat{x}(k|k) + B_1\hat{x}(k|k)u(k) + Wx_f(k) + Bu(k) + \\ + K^*(k)\left[z(k) - C\hat{x}(k|k)\right] \end{bmatrix} - \\
&\quad - \left[Ax(k) + B_1x(k)u(k) + Bu(k) + Wx_f(k|k) + w(k|k)\right] = \\
&= A\left[\hat{x}(k|k) - x(k)\right] + B_1\left[\hat{x}(k|k) - x(k)\right]u(k) + \\
&\quad + K^*(k)\left[z(k) - C\hat{x}(k|k)\right] - w(k|k) = \\
&= Ae(k|k) + B_1e(k|k)u(k) \\
&\quad + K^*(k)\left[z(k) - C\hat{x}(k|k)\right] - w(k|k)
\end{aligned}
\tag{9.52}
$$

Finally,

$$
\begin{aligned}
e(k+1|k+1) &= Ae(k|k) + B_1e(k|k)u(k) + \\
&\quad + K^*(k)\left[Cx(k|k) + v(k|k) - C\hat{x}(k|k)\right] - w(k|k) = \\
&= \left[Ae(k|k) - K^*(k)C\right]e(k|k) + B_1e(k|k) + K^*(k)v(k|k) - w(k|k)
\end{aligned}
\tag{9.53}
$$

The *a posteriori* covariance matrix is expressed by

$$
P(k+1|k+1) = E\left\{e(k+1|k+1)e(k+1|k+1)^T\right\} = J(k+1) \tag{9.54}
$$

The derivation of the *a posteriori* covariance matrix is achieved through substitution of the error $e(k+1|k+1)$ into the Equation (9.54).

The error equation must now be substituted back into the criterion,

$$
\begin{aligned}
J(k+1) &= E\left\{e(k+1|k+1)\times e(k+1|k+1)^T\right\} = \\
&= E\left\{\begin{aligned} &\left[\left[A - K^*(k)C\right]e(k|k) + B_1e(k|k)u(k) + K^*(k)v(k) - w(k|k)\right]\times \\ &\times\left[\left[A - K^*(k)C\right]e(k|k) + B_1e(k|k)u(k) + K^*(k)v(k|k) - w(k|k)\right]^T \end{aligned}\right\} =
\end{aligned}
$$

$$
\begin{aligned}
&= E\left\{\cdot[A-K^{*}(k)C]e(k)\left[[A-K^{*}(k)C]e(k|k)\right]^{T}\right\}+E\left\{\ [A-K^{*}(k)C]e(k|k)\left[B_{1}e(k|k)u(k)\right]^{T}\right\}+\\
&+E\left\{\cdot[A-K^{*}(k)C]e(k|k)\left[K^{*}(k)v(k|k)\right]^{T}\right\}-E\left\{\ [A-K^{*}(k)C]e(k|k)w(k|k)^{T}\right\}+E\left\{B_{1}e(k|k)u(k)\left[[A-K^{*}(k)C]e(k|k)\right]^{T}\right\}+\\
&+E\left\{B_{1}e(k|k)u(k)\left[B_{1}e(k|k)u(k)\right]^{T}\right\}+E\left\{B_{1}e(k|k)u(k)\left[K^{*}(k)v(k|k)\right]^{T}\right\}-E\left\{B_{1}e(k|k)u(k)w(k|k)^{T}\right\}+\\
&+E\left\{K^{*}(k)v(k|k)\left[[A-K^{*}(k)C]e(k|k)\right]^{T}\right\}+E\left\{K^{*}(k)v(k|k)\left[B_{1}e(k|k)u(k)\right]^{T}\right\}+E\left\{K^{*}(k)v(k|k)\left[K^{*}(k)v(k|k)\right]^{T}\right\}-\\
&-E\left\{K^{*}(k)v(k|k)w(k|k)^{T}\right\}-E\left\{w(k|k)\left[[A-K^{*}(k)C]e(k|k)\right]^{T}\right\}-E\left\{w(k|k)\left[B_{1}e(k|k)u(k)\right]^{T}\right\}-\\
&-E\left\{w(k|k)\left[K^{*}(k)v(k|k)\right]^{T}\right\}+E\left\{w(k|k)w(k|k)^{T}\right\}=\\
&=\left[A-K^{*}(k)C\right]E\left\{e(k|k)e(k|k)^{T}\right\}\left[A-K^{*}(k)C\right]^{T}+\left[A-K^{*}(k)C\right]E\left\{e(k|k)e(k|k)^{T}\right\}[B_{1}u(k)]^{T}+\\
&+[A-K^{*}(k)C]E\left\{e(k|k)v(k|k)^{T}\right\}K^{*}(k)^{T}-\left[A-K^{*}(k)C\right]E\left\{e(k|k)w(k|k)\right\}+B_{1}u(k)E\left\{e(k|k)e(k|k)^{T}\right\}\left[A-K^{*}(k)C\right]^{T}+\\
&+B_{1}u(k)E\left\{e(k|k)e(k|k)^{T}\right\}[B_{1}u(k)]^{T}+B_{1}u(k)E\left\{e(k|k)v(k|k)^{T}\right\}K^{*}(k)^{T}-B_{1}u(k)E\left\{e(k|k)w(k|k)^{T}\right\}+\\
&+K^{*}(k)E\left\{v(k|k)e(k|k)^{T}\right\}\left[A-K^{*}(k)C\right]^{T}+K^{*}(k)E\left\{v(k|k)e(k|k)^{T}\right\}[B_{1}u(k)]^{T}+\\
&+K^{*}(k)E\left\{v(k|k)v(k|k)^{T}\right\}K^{*}(k)^{T}-K^{*}(k)E\left\{v(k|k)w(k|k)^{T}\right\}-E\left\{w(k|k)e(k|k)^{T}\right\}\left[A-K^{*}(k)C\right]^{T}-\\
&-E\left\{w(k|k)e(k|k)^{T}\right\}[B_{1}u(k)]^{T}-E\left\{w(k|k)v(k|k^{T})\right\}K^{*}(k)^{T}+E\left\{w(k|k)w(k|k)^{T}\right\}
\end{aligned}
\tag{9.55}
$$

Using the condition that there is no correlation between the error $e(k)$ and the stochastic noise processes of the system $w(k)$ and $v(k)$, the criterion simplifies to

$$
\begin{aligned}
J(k+1) = [A-K^{*}(k)C]P(k|k)[A-K^{*}(k)C]^{T}+B_{1}u(k)P(k|k)[A-K^{*}(k)C]^{T}+[A-K^{*}(k)C]P(k|k)[B_{1}u(k)]^{T}+\\
+B_{1}u(k)P(k|k)[B_{1}u(k)]^{T}+K^{*}(k)V^{*}(k|k)K^{*}(k)^{T}+W^{*}(k|k)
\end{aligned}
\tag{9.56}
$$

where

$E\left\{e(k|k)e(k|k)^{T}\right\}=P(k|k)$ and $P(k|k)$ is the covariance matrix of the estimation error at the current moment k.

It can be seen that minimization of the criterion function is equivalent to the minimization of the covariance matrix of the prediction error at the same moment of time as in that of the criterion, thus,

$$P(k+1|k+1) \to \min \cong J(k+1) \to \min \tag{9.57}$$

Equation (9.59) shows that the *posteriori* covariance matrix $P(k+1|k+1)$ is expressed by the *a priori* matrix $P(k|k)$. The connection between these covariance matrices is through the Kalman gain $K^*(k)$, as determined previously. The derivation of the Kalman gain is presented as shown by Equations (9.58)–(9.66)

$$\begin{aligned}
&P(k+1|k+1) = \\
&= \left[A - K^*(k)C\right]P(k|k)\left[A - K^*(k)C\right]^T + B_1u(k)P(k|k)\left[A - K^*(k)C\right]^T + \\
&\quad + \left[A - K^*(k)CP(k|k)\right]\left[B_1u(k)\right]^T + B_1u(k)P(k|k)u(k)^T B_1^T + \\
&\quad + K^*(k)V^*(k|k)K^*(k)^T + W^*(k|k) = \\
&= AP(k|k)A^T - AP(k|k)C^T K^*(k)^T - K^*(k)CP(k|k)A^T + \\
&\quad + K^*(k)CP(k|k)C^T K^*(k)^T + B_1u(k)P(k|k)A^T - B_1u(k)P(k|k)C^T K^*(k)^T + \\
&\quad + AP(k|k)u(k)B_1^T - K^*(k)CP(k|k)u^T(k)B_1^T \\
&\quad + B_1u(k)P(k|k)u^T(k)B_1^T + K^*(k)V^*(k|k)K^*(k)^T + W^*(k|k) = \\
&= AP(k|k)A^T - \left[AP(k|k)C^T K^*(k)^T\right] - \left[K^*(k)CP(k|k)A^T\right] - \\
&\quad - \left[B_1u(k)P(k|k)C^T K^*(k)^T\right] - \left[K^*(k)CP(k|k)u^T(k)B_1^T\right]^T + \left[B_1u(k)P(k|k)A^T\right] + \\
&\quad + \left[AP(k|k)u(k)B_1^T\right]^T + \left[B_1u(k)P(k|k)u^T(k)B_1^T\right] + \\
&\quad + \left[K^*(k)CP(k|k)C^T K^*(k)^T\right] + K^*(k)V^*(k|k)K^*(k)^T + W^*(k|k) = \\
&= AP(k|k)A^T - \left[AP(k|k)C^T K^*(k)^T + AP(k|k)^T C^T K^*(k)\right] - \\
&\quad - \left[B_1u(k)P(k|k)C^T K^*(k)^T + B_1^T u(k)P(k|k)C^T K^*(k)^T\right] + \\
&\quad + \left[B_1u(k)P(k|k)A^T + B_1u(k)P(k|k)^T A^T\right] + \left[B_1u(k)P(k|k)u^T(k)B_1^T\right] + \\
&\quad + \left[K^*(k)CP(k|k)C^T K^*(k)^T + K^*(k)V^*(k|k)K^*(k)^T\right] + W^*(k|k)
\end{aligned}$$

(9.58)

The covariance matrix $P(k|k)$ is nonnegative and symmetrical and that implies $P^T(k|k) = P(k|k)$; and therefore,

$$\begin{aligned} &P(k+1|k+1) = \\ &\quad = AP(k|k)A^T + 2B_1u(k)P(k|k)A^T + B_1u(k)P(k|k)u^T(k)B_1{}^T + W^*(k) + \\ &\qquad + K^*(k)\left[CP(k|k)C^T + V^*(k|k)\right] K^*(k)^T - \\ &\qquad - 2\left[AP(k|k)C^T + B_1u(k)P(k|k)C^T\right] K^*(k)^T \end{aligned} \tag{9.59}$$

Let

$$CP(k|k)C^T + V^*(k|k) = Q(k|k) \tag{9.60}$$

Then, the final expression for the posteriori covariance becomes,

$$\begin{aligned} P(k+1|k+1) = {} & AP(k|k)A^T + 2B_1u(k)P(k|k)A^T + \\ & + B_1u(k)P(k|k)u^T(k)B_1{}^T + K^*(k)Q(k|k)K^*(k) - \\ & - 2[AP(k|k)C^T + B_1u(k)P(k|k)C^T]K^*(k) + W^*(k) \end{aligned} \tag{9.61}$$

To determine the Kalman gain matrix from Equation (9.61) the *mean square method* on the $K^*(k)$ terms of the equation is used. This means that at optimal values of the criterion function, the expressions containing the Kalman matrix gain terms will approach zero. This expression is derived as follows:

$$K^*(k)Q(k|k)K^*(k)^T - 2\left[AP(k|k)C^T + B_1u(k)P(k|k)C^T K^*(k)^T\right] = 0 \tag{9.62}$$

$$K^*(k)K^*(k)^T - 2\left[AP(k|k)C^T + B_1u(k)C^T\right] K^*(k)^T Q^{-1}(k|k) = 0 \tag{9.63}$$

where $Q^{-1}(k|k)$ is a block diagonal matrix.

The Equation (9.63) can be regrouped using completion of a square to arrive at Equation (9.64)

$$\begin{aligned} &\left[K^*(k) - \left[AP(k|k)C^T + B_1u(k)C^T\right]Q^{-1}(k|k)\right]^2 + \\ &\qquad + \left[\left[AP(k|k)C^T + B_1u(k)C^T\right] Q^{-1}(k|k)\right]^2 = 0 \end{aligned} \tag{9.64}$$

To simplify this equation further, both sides of the equation are multiplied by $Q(k|k)$,

$$\begin{aligned} &\left[K^*(k) - \left[AP(k|k)C^T + B_1u(k)C^T\right]Q^{-1}(k|k)\right]^2 Q(k|k) - \\ &\qquad - \left[\left[AP(k|k)C^T + B_1u(k)C^T\right]Q^{-1}(k|k)\right]^2 Q(k|k) = 0 \end{aligned} \tag{9.65}$$

The $K^*(k)$ filter gain terms in the Equation (9.65) must go to zero for optimal filter gain values, i.e.,

$$\left[K^*(k) - \left(AP(k|k)C^T + B_1u(k)C^T\right)Q^{-1}(k|k)\right]^2 Q(k|k) = 0 \tag{9.66}$$

Equation (9.66) can be expressed in the following manner:

$$\left[K^*(k)-\left(AP(k|k)C^T+B_1u(k)C^TQ^{-1}(k|k)\right)\right]Q(k|k)\times$$
$$\times\left[K*(k)-\left(AP(k|k)C^T+B_1u(k)C^TQ^{-1}(k|k)\right)\right] \qquad (9.67)$$

or

$$K^*(k)=\left(AP(k|k)C^T+B_1u(k)C^T\right)Q^{-1}(k|k)=$$
$$=\left[AP(k|k)C^T+B_1u(k)C^T\right]\left[CP(k|k)C^T+V^*(k|k)\right]^{-1}$$

From this point the criterion for minimization has to be implemented $K^*(k)-\left(AP(k|k)C^T+B_1u(k)C^T\right)Q^{-1}(k|k)=0$

Substituting the Kalman gain matrix $K^*(k)$ from Equation (9.67) back to the covariance matrix for the prediction error, Equation (9.61) gives,

$$P(k+1|k+1)=AP(k|k)A^T+2B_1u(k)P(k|k)A^T+B_1u(k)P(k|k)u(k)^TB_1{}^T+$$
$$+W^*(k)-\left[AP(k|k)C^T+B_1u(k)C^T\right]Q^{-1}(k|k)\left[AP(k|k)C^T+B_1u(k)C^T\right]^T=$$
$$=AP(k|k)A^T+2B_1u(k)P(k|k)A^T+B_1u(k)P(k|k)u(k)^TB_1{}^T-$$
$$-K^*(k)\left[AP(k|k)C^T+B_1u(k)C^T\right]^T+W^*(k|k) \qquad (9.68)$$

Equation (9.67) determines the Kalman gain at the moment $(k|k)$ in terms of the posteriori covariance matrix at the moment $(k|k)$, the control input $u(k)$ and the covariance matrix of the output measurements. The posteriori covariance matrix $P(k+1|k+1)$ is determined in terms of the covariance matrix $P(k|k)$, the control input, the Kalman gain in the moment $(k|k)$ and the covariance matrix of the process noise $W^*(k|k)$.

9.6. Derivation of the discrete Kalman filter gain for the continuous countercurrent ion exchange (CCIX) process (bilinear model) –method of direct optimization

The second method's problem is formulated as follows: find the values of the Kalman gain matrix at every moment of time k in such a way that the errors of the state estimates from their true values are minimized according to the criterion

$$J(k)=\frac{1}{2}E\left\{\left[\hat{x}(k+1|k+1)-x(k+1)\right]^2\right\}\rightarrow\min \qquad (9.69)$$

where $\hat{x}(k+1|k+1)$ is the estimate of the state vector at the moment $(k+1|k+1)$ obtained by the Kalman filter.

The following are the assumptions for computational foundation of the Kalman filter gain:

1) the behaviour of the process prior to the moment $(k+1|k+1)$ is available. The estimate $\hat{x}(k+1|k) \in R^n$ is defined as a *priori state estimate* as it is determined on the basis of the process available data.

2) *a priori* state estimate error is defined as:

$$e(k+1|k) = \hat{x}(k+1|k) - x(k+1) \tag{9.70}$$

for the $k+1$ under given knowledge for the process until the moment $(k+1)$ and

$$e(k|k-1) = \hat{x}(k|k-1) - x(k) \tag{9.71}$$

for the previous sampling moment.

3) the state estimate $\hat{x}(k+1|k+1) \in R^n$ is determined at the moment $(k+1|k+1)$ using the given measurement $y(k+1)$ and is called a *posteriori state estimate*.

4) *a posteriori state estimate error* is defined as,

$$e(k+1|k+1) = \hat{x}(k+1|k+1) - x(k+1) \tag{9.72}$$

and

$$e(k|k) = \hat{x}(k|k) - x(k) \tag{9.73}$$

for the previous moment.

5) *a priori estimate error covariance* is calculated as

$$P(k+1|k) = E\left\{e(k+1|k)e(k+1|k)^T\right\} \tag{9.74}$$

and

$$P(k|k-1) = E\left\{e(k|k-1)e(k|k-1)^T\right\} \tag{9.75}$$

for the previous moment.

6) *a posteriori estimate error covariance matrix* is calculated as

$$P(k+1|k+1) = E\left\{e(k+1|k+1)e(k+1|k+1)^T\right\} \tag{9.76}$$

and

$$P(k|k) = E\left\{e(k|k)e(k|k)^T\right\} \tag{9.77}$$

for the previous moment.

7) *a posteriori estimate* $\hat{x}(k+1|k)$ is obtained from the model. It is supposed that the *a posteriori estimate* $\hat{x}(k|k)$ is known and the *a priori estimate* at the moment $(k+1|k)$ is

$$\hat{x}(k+1|k) = A\hat{x}(k|k) + B_1\hat{x}(k|k)u(k) + Bu(k) + Wx_f(k) \tag{9.78}$$

8) the *a posteriori* and the *a priori* estimates are connected by the measurement of the output at the moment $k+1$

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K^*(k+1)\left[z(k+1) - C\hat{x}(k+1|k)\right] \tag{9.79}$$

It is supposed that *a posteriori estimate* is calculated on the basis of the *a priori* one with added to it value of the weighted difference between the actual measurement $z(k+1)$ and the predicted output $C\hat{x}(k+1|k)$.

The difference between these measurements $z(k+1) - C\hat{x}(k+1|k)$ is called the *measurement innovation* or a *residual*. This difference determines the error between the real measurement and the estimated measurement. The matrix $K^*(k)$ (the *state output*) is called the *Kalman gain*. It has to be determined in such a way that the value of the *a posteriori error covariance matrix* $P(k+1|k+1)$ is minimized.

The derivation of the Kalman filter is based on the assumptions as listed (Cao, et al., 2016; Julier, et al., 2000). It is necessary to calculate the Kalman gain based on the *a priori* and *a posteriori covariance matrices* for the estimation errors at the moments $(k+1|k)$ and $(k+1|k+1)$. At the same time the value of the criterion for optimization $J(k+1)$ is equal to that of the *a posteriori covariance matrix* at the moment $(k+1|k+1)$. This means that the minimization of the criterion function is equivalent to the minimization of the *a posteriori covariance matrix* $P(k+1|k+1)$. The derivation is as follows:

9.6.1. Derivation of the *a priori estimate error* covariance matrix

The derivation of the *a priori estimate error covariance matrix* is done following Equation (9.74). First, the estimation error is determined as follows:

$$\begin{aligned}
&e(k+1|k) = \hat{x}(k+1|k) - x(k+1) = \\
&= [A\hat{x}(k|k) + B_1\hat{x}(k|k)u(k) + Bu(k) + Wx_f(k)] - \\
&\qquad - [Ax(k) + B_1x(k)u(k) + Bu(k) + Wx_f(k) + w(k)] = \\
&= A[\hat{x}(k|k) - x(k)] + B_1u(k)[\hat{x}(k|k) - x(k)] - w(k) = \\
&= Ae(k|k) + B_1e(k|k)u(k) - w(k), \qquad\qquad with\, u(k) \in R^1
\end{aligned} \tag{9.80}$$

Equation (9.80) is then substituted into Equation (9.74)

$$
\begin{aligned}
P(k+1|k) &= E\left\{e(k+1|k)e(k+1|k)^T\right\} = \\
&= E\left\{[Ae(k|k)+B_1e(k|k)u(k)-w(k)][Ae(k|k)+B_1e(k|k)u(k)-w(k)]^T\right\} = \\
&= E\left\{Ae(k|k)e(k|k)^T A^T\right\} + E\left\{B_1e(k|k)u(k)e(k|k)^T A^T\right\} - \\
&- E\left\{w(k)e(k|k)^T A^T\right\} + E\left\{Ae(k|k)u(k)^T e(k|k)^T B_1{}^T\right\} + \\
&+ E\left\{B_1e(k|k)u(k)u(k)^T e(k|k)^T B_1{}^T\right\} - E\left\{w(k)u(k)^T e(k|k)^T B_1{}^T\right\} - \\
&- E\left\{Ae(k|k)w(k)^T\right\} - E\left\{B_1e(k|k)u(k)w(k)^T\right\} + E\left\{w(k)w(k)^T\right\}
\end{aligned}
\tag{9.81}
$$

In Equation (9.81) $u(k)^T = u(k)$ as the input $u(k)$ is a simple variable. On the basis that the variables $u(k)$ and $e(k|k)$, and the variables $e(k|k)$ and $w(k)$ are not correlated as $x(k)$ and not $e(k|k)$ directly depends on $u(k)$ and $w(k)$. This relationship can be expressed as:

$$E\left\{w(k)e(k|k)^T\right\} = 0\,, \tag{9.82}$$

$$E\left\{e(k|k)w(k)^T\right\} = 0 \tag{9.83}$$

In addition, from the definition of the noise $w(k)$

$$E\left\{w(k)w(k)^T\right\} = W^*(k) \tag{9.84}$$

and from

$$P(k|k) = E\left\{e(k|k)e(k|k)^T\right\} \tag{9.85}$$

Using the arguments in Equations (9.73), the *error covariance matrix* can be written as

$$
\begin{aligned}
P(k+1|k) &= E\left\{Ae(k|k)e(k|k)^T A^T\right\} + E\left\{Au^T(k)e(k|k)e(k|k)^T B_1{}^T\right\} + \\
&+ E\left\{B_1u(k)^2 e(k|k)e(k|k)^T B_1{}^T\right\} + E\left\{w(k)w(k)^T\right\}
\end{aligned}
\tag{9.86}
$$

And from the *error covariance matrix* at the current moment k, Equation (9.74), finally Equation (9.86) is simplified to

$$
\begin{aligned}
P(k+1|k) &= AP(k|k)A^T + B_1u(k)P(k|k)A^T + \\
&+ Au(k)P(k|k)B_1{}^T + B_1u(k)^2 P(k|k)B_1{}^T + W^*(k)
\end{aligned}
\tag{9.87}
$$

9.6.2. Derivation of the *a posteriori estimate error* covariance matrix

For the determination of $P(k+1|k+1)$; first, the model output is added to the estimate Equation (9.79). And then the obtained equation can be written as

$$
\begin{aligned}
\hat{x}(k+1|k+1) &= \hat{x}(k+1|k) + K^*(k+1)\left[Cx(k+1)+v(k)-C\hat{x}(k+1|k)\right] = \\
&= \left[I-K^*(k+1)C\right]\hat{x}(k+1|k) + K^*(k+1)Cx(k+1) + K^*(k+1)v(k)
\end{aligned}
\tag{9.88}
$$

Equation (9.88) is substituted into the equation for calculation of the *a posteriori error*, Equation (9.72)

$$\begin{aligned}e(k+1|k+1) &= \hat{x}(k+1|k+1) - x(k+1) = \\ &= [I - K^*(k+1)C]\hat{x}(k+1|k) + K^*(k+1)Cx(k+1) + \\ &\qquad + K^*(k+1)v(k+1) - x(k+1) = \\ &= [I - K^*(k+1)C]\hat{x}(k+1|k) + K^*(k+1)Cx(k+1) + K^*(k+1)v(k+1) - \\ &\qquad - K^*(k+1)Cx(k+1) + K^*(k+1)Cx(k+1) - Ix(k+1) = \\ &= \left[I - K^*(k+1)C\right]\left[\hat{x}(k+1|k) - x(k+1)\right] + K^*(k+1)v(k+1)\end{aligned} \tag{9.89}$$

Finally,

$$e(k+1|k+1) = [I - K^*(k+1)C]e(k+1|k) + K^*(k+1)v(k+1) \tag{9.90}$$

where $[\hat{x}(k+1|k) - x(k+1)] = e(k+1|k)$, Equation (9.70).

Second, the calculation of the covariance matrix $P(k+1|k+1)$ is done using Equations (9.74) and (9.90).

$$\begin{aligned}P(k+1|k+1) &= E\left\{e(k+1|k+1)e(k+1|k+1)^T\right\} = \\ &= E\left\{\begin{matrix}\left[\left[I - K^*(k+1)C\right]e(k+1|k) + K^*(k+1)v(k+1)\right]\times \\ \times\left[\left[I - K^*(k+1)C\right]e(k+1|k) + K^*(k+1)v(k+1)\right]^T\end{matrix}\right\} = \\ &= E\left\{\begin{matrix}\left[I - K^*(k+1)C\right]e(k+1|k)e(k+1|k)^T\left[I - K^*(k+1)C\right]^T + \\ + \left[I - K^*(k+1)C\right]e(k+1|k)v(k+1)^T K^*(k+1)^T + \\ + K^*(k+1)v(k+1)e(k+1|k)^T\left[I - K^*(k+1)C\right]^T + \\ + K^*(k+1)v(k+1)v(k+1)^T K^*(k+1)\end{matrix}\right\}\end{aligned} \tag{9.91}$$

In Equation (9.91), the following is observable; there is no correlation between $e(k+1|k)$ and $v(k+1)$ since $y(k+1)$ and not, $e(k+1|k)$ depends on $v(k)$. From this observation, the following statements can be generated:

$$E\left\{e(k+1|k)v(k+1)^T\right\} = 0 \tag{9.92}$$

$$E\left\{v(k+1)e(k+1|k)^T\right\} = 0 \tag{9.93}$$

Based on the same argument as that used in Equations (9.15)–(9.25), the following statements can be written:

$$E\left\{v(k+1)v(k+1)^T\right\} = V^*(k) \tag{9.94}$$

$$E\left\{e(k+1|k)e(k+1|k)^T\right\} = P(k+1|k) \tag{9.95}$$

This means that the expression in Equation (9.91) can be simplified as

$$\begin{aligned} P(k+1|k+1) &= [I-K^*(k+1)C]E\{e(k+1|k)e(k+1|k)^T\}[I-K^*(k+1)C]^T + \\ &+ K^*(k+1)E\{v(k+1)v(k+1)^T\}K^*(k+1)^T = \\ &= [I-K^*(k+1)C]P(k+1|k)[I-K^*(k+1)C]^T + \\ &+ K^*(k+1)V^*(k+1)K^*(k+1)^T \end{aligned} \qquad (9.96)$$

9.6.3. Determination of the optimal value for the Kalman filter gain

In order to optimize the Kalman filter, it is necessary to *minimize the error covariance matrix*,

$$P(k+1|k+1) = J(k+1) \to \min \qquad (9.97)$$

according to the Kalman gain matrix $K^*(k+1)$, or written differently,

$$\left\{\begin{matrix}\min\{J(k+1)\} \\ K^*(k+1)\end{matrix}\right. = \left\{\begin{matrix}\min\left\{\begin{matrix}[I-K^*(k+1)C]P(k+1|k)[I-K^*(k+1)C]^T + \\ + K^*(k+1)V^*(k+1)K^*(k+1)\end{matrix}\right\} \\ K^*(k+1)\end{matrix}\right. \qquad (9.98)$$

for $k = 1, 2, 3, \ldots$

where $$\begin{aligned} P(k+1|k+1) &= [I-K^*(k+1)C]P(k+1|k)[I-K^*(k+1)C]^T + \\ &+ K^*(k+1)V^*(k+1)K^*(k+1) \end{aligned} \qquad (9.99)$$

According to optimization theory, it is possible to obtain the optimal gain on the basis of the first derivative of the *criterion functional* according to the Kalman gain; this can be done as follows

$$\begin{aligned} \frac{\partial J(k+1)}{\partial K^*(k+1)} &= \frac{\partial}{\partial K^*(k+1)}\left\{\begin{matrix}[P(k+1|k)-K^*(k+1)CP(k+1|k)][I-K^*(k+1)C]^T + \\ + K^*(k+1)V^*(k+1)K^*(k+1)^T\end{matrix}\right\} = \\ &= \frac{\partial}{\partial K^*(k+1)}\left\{\begin{matrix}\left[\begin{matrix}P(k+1|k)-P(k+1|k)C^TK^*(k+1)^T - \\ -K^*(k+1)CP(k+1|k) + \\ + K^*(k+1)CP(k+1|k)C^TK^*(k+1)^T\end{matrix}\right] + \\ + [K^*(k+1)V^*(k+1)K^*(k+1)^T]\end{matrix}\right\} = 0 = \\ &= -P(k+1|k)C^T - CP(k+1|k) + \left\{\begin{matrix}CP(k+1|k)C^TK^*(k+1)^T + \\ + K^*(k+1)CP(k+1|k)C^T\end{matrix}\right\} + \\ &+ \{V^*(k+1)K^*(k+1)^T + K^*(k+1)V^*(k+1)\} = \end{aligned} \qquad (9.100)$$

$$\frac{\partial J(k+1)}{\partial K^*(k+1)} = -P(k+1|k)C^T - P^T(k+1|k)C^T + \left\{ \begin{matrix} K^*(k+1)CP^T(k+1|k)C^T + \\ + K^*(k+1)CP(k+1|k)C^T \end{matrix} \right\} + \\ + \left[\begin{matrix} K^*(k+1)V^*(k+1)^T + \\ + K^*(k+1)V^*(k+1) \end{matrix} \right] = 0$$

(9.101)

Since both the covariance matrices are symmetrical in the first derivatives, $P(k+1|k) = P(k+1|k)^T$ and $V(k+1) = V(k+1)^T$. Equation (9.101) then becomes

$$\frac{\partial J(k+1)}{\partial K^*(k+1)} = -P(k+1|k)C^T + K^*(k+1)\left[CP(k+1|k)C^T + V^*(k+1)\right] = 0 \quad (9.102)$$

where

$CP(k+1|k) = P^T(k+1|k)C^T$,

$CP(k+1|k)C^T K^*(k+1)^T = K^*(k+1)CP^T(k+1|k)C^T$, (9.103)

$V^*(k+1)K^*(k+1)^T = K^*(k+1)V^*(k+1)^T$, and

since $V^*(k+1)$ and $P(k+1|k)$ are symmetrical, this implies that,

$$CP(k+1|k) = P^T(k+1|k)C^T = P(k+1|k)C^T \quad (9.104)$$

and

$$CP(k+1|k)C^T K^*(k+1)^T = K^*(k+1)CP(k+1|k)C^T \quad (9.105)$$

$$V(k+1)K^*(k+1)^T = K^*(k+1)V^*(k+1)^T = K^*(k+1)V^*(k+1) \quad (9.106)$$

The optimal gain of the filter can then be expressed as,

$$K^*(k+1) = P(k+1|k)C^T\left[CP(k+1|k)C^T + V^*(k+1)\right]^{-1} \quad (9.107)$$

This is known as the *optimal Kalman filter gain*.

Equation (9.107) is different from Equation (9.67) where $K^*(k) = \left[AP(k|k)C^T + B_1u(k)C^T\right]\left[CP(k|k)C^T + V*(k|k)\right]^{-1}$, the expression $B_1u(k)C^T$ is not included but in the $P(k+1|k)$ expression.

9.6.4. Update of the error *a posteriori covariance matrix* on the basis of the optimal filter gain matrix

Optimal values of the gain $K^*(k+1)$ can be used to calculate the optimal value of the error covariance matrix $P(k+1|k+1)$ by substituting Equation (9.107) as $K^*(k+1)V(k+1)^T = P(k+1|k)C^T - K^*(k+1)CP(k+1|k)C^T$ to Equation (9.96).

$$\begin{aligned} P(k+1|k+1) = {} & \left[I - K^*(k+1)C\right]P(k+1|k)\left[I - K^*(k+1)C\right]^T + \\ & + K^*(k+1)V(k+1)K^*(k+1)^T = \\ = {} & \left[P(k+1|k) - K^*(k+1)CP(k+1|k)\right]\left[I - K^*(k+1)C\right]^T + \\ & + K^*(k+1)V^*(k+1)K^*(k+1)^T = \\ = {} & P(k+1|k) - P(k+1|k)C^T K^*(k+1)^T - K^*(k+1)CP(k+1|k) + \\ & + K^*(k+1)CP(k+1|k)C^T K^*(k+1)^T + \left\{K^*(k+1)V^*(k+1)\right\}K^*(k+1)^T = \\ = {} & P(k+1|k) - K^*(k+1)CP(k+1|k) - P(k+1|k)C^T K^*(k+1)^T + \\ & + K^*(k+1)CP(k+1|k)C^T K^*(k+1)^T + \\ & + \left\{P(k+1|k)C^T - K^*(k+1)CP(k+1|k)C^T\right\}K^*(k+1) = \\ = {} & P(k+1|k) - K^*(k+1)CP(k+1|k) \end{aligned} \tag{9.108}$$

From Equation (9.108) it can be seen that *a posteriori covariance matrix* is determined by the *a priori covariance matrix*.

$$P(k+1|k+1) = \left[I - K^*(k+1)C\right]P(k+1|k) \tag{9.109}$$

Equation (9.106) determines the *optimal Kalman filter gain* at the moment of $(k+1|k+1)$ as a function of the *a priori error covariance matrix* $P(k+1|k)$ at the moment $(k+1|k)$, and of the covariance matrix for the noise in the measurement at moment $(k+1|k)$, Figure 9.3. The value of the *a posteriori covariance matrix* at the moment $(k+1|k+1)$ is a function of the Kalman filter gain $K^*(k+1)$ and the *a priori error covariance matrix* $P(k+1|k)$.

9.7. Comparison of the two developed methods for derivation of the filter

The Kalman filter is a recursive algorithm as mentioned earlier, and calculations are done through two stages to produce optimal filter gains. The first stage is a *prediction stage* followed by the *correction stage*. Each stage is also at a different discrete moment of time, $(k+1)$ and k, Figure 9.3. The prediction stage is referred to as *Time Update*. It contains all necessary equations to determine the prediction of the state estimate. The correction stage is known as the *Measurement Update*, and it contains a set of equations necessary to correct the estimate if need be. At the *Measurement Update*, the key factor is the moment for the output measurement.

The nature of the output measurement used in the algorithm determines which method to follow. The first method assumes that the measurement output is coming from the previous measurement at the moment $(k|k)$, and that the current measurement is not available during the correction. The second method, the direct optimization considers a case where it is possible to use the current measurement at moment $(k+1|k+1)$.

There is also a need that the two moments of time be connected, this relates the state estimates at prediction moment to the correction update moment. This provides the recursive nature of the Kalman filter in that it predicts and corrects system states for every moment of time $(k+1|k+1)$ using measurements at the previous moment $(k+1|k)$, until such time that the estimates are optimal, Figure 9.4.

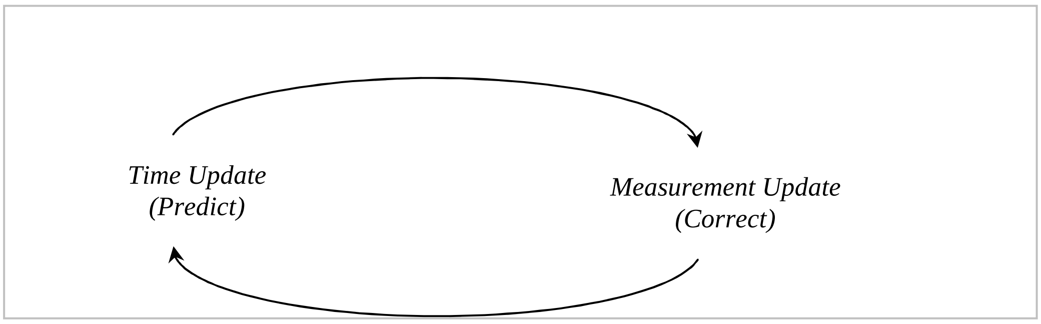


Figure 9.4. **Schematic representation of the two step procedure for estimating unknown states using measured data**

Based on the chosen moment of time, state estimates are defined as *a priori* or *a posteriori*, Figure 9.3. If the moment k is used, the *a priori* is the moment $(k|k-1)$ and the *a posteriori* is the moment $(k|k)$. If the moment $(k+1)$ is used, the *a priori* is the moment $(k+1|k)$ and the a posteriori is the moment $(k+1|k+1)$.

The filter algorithm revolves around the next moment $(k+1)$ because this is where the measurement (correction) is done. The two methods use naming convention based on the *output update moment* used, either at moment $(k+1)$ or moment k. The procedures for each method are discussed in the subsections below.

9.7.1. The first method (general formulation of the filter)

In the general formulation method, at the beginning of the estimation procedure, the process dynamic behaviour is assumed to be known prior to moment $(k+1)$, and thus the state estimate at this moment is defined by the *a posterori state* *estimate*, given by $\hat{x}(k|k)$. The *a priori estimate* assumes that the previous estimate is known (is available), the *a posteriori state estimate*, given by $x(k|k)$. In the *first method*:

- o the correction is done at the moment $(k+1|k+1)$ but the Kalman gain is calculated at moment $(k|k)$ and the measurement is done at $(k|k)$. This means that in the filter application, the procedure needs to store the previous state estimate; which is a negative in comparison to the direct optimization method.

The general method involves a very long derivation procedure before getting to the final gains calculation. Starting with the *a priori* state estimate at the current moment $(k+1|k)$, and then at the next sampling moment the filter provides a correction to the current estimate via the connection between *a priori* and *a posteriori* estimates. This connection is achieved through an assumption that at the next moment $(k+1|k+1)$, an *a posteriori estimate*, Equation(9.47), which depends on *a priori* estimate Equation (9.44) is determinable.

The error difference must then be determined, to be able to minimize any variance of the estimated state from the real state. This procedure is based on statistical methods using covariance of the estimation error. This means that an *a posteriori covariance matrix*, Equation (9.49) must be calculated. Based on the derivation method, in order to be able to determine the a *posteriori covariance matrix*, the Kalman filter gain must first be calculated.

At the moment $(k+1|k+1)$, the *estimation error* and the *a posteriori covariance matrix* are defined by $e(k+1|k+1)$ and $P(k+1|k+1)$ respectively. The *a posteriori covariance matrix* is used as an optimization criterion to produce optimal estimate. The final *a posteriori prediction covariance error matrix* is given in terms of: the system input $u(k)$, the *a priori prediction covariance error matrix* $P(k|k)$, the Kalman gain filter $K^{*}(k)$, and the system noise covariance $W^{*}(k|k)$. All these terms are calculated at the previous moment $(k|k)$. The full procedure for calculation of the optimal estimate is given in the algorithm, in the next section. A summary of equations involved is given.

9.7.2. The second method (direct optimization)

The second method utilizes the direct optimization technique. The method minimizes an optimization criterion based on the prediction error between the current estimate and that obtained from the output using the measurement update (correction) at the moment $(k+1|k+1)$. Even in this method, it is necessary to calculate the Kalman filter gains before obtaining the optimal states estimates. The filter gains require the following information to be available before they could be calculated:

- the process dynamic behavioiur prior to the moment $(k+1)$,
- this allows the determination of the state estimate $\hat{x}(k+1|k)$, the *a priori state estimate*,
- from all these details, the optimal state estimate is obtained via the *a posteriori estimate* calculated at moment $(k+1|k+1)$,
- the calculation of the *a posteriori* state estimate requires that the output at moment $(k+1)$ be known which is the major difference with the first method which uses the output from the measurement at the moment k.

In the same fashion as in the first method, it is necessary to calculate the state estimate at the next moment of measurement to produce optimal estimates. The state estimate $\hat{x}(k+1|k+1)$ for correction of the error in estimation is obtained using the output measurement $y(k+1)$, and all the calculations are done at moment $(k+1|k+1)$. The estimate at this moment is known as the *a posteriori state estimate* $\hat{x}(k+1|k+1)$. Once again to calculate this estimate, the previous state estimate $\hat{x}(k+1|k)$ is required.

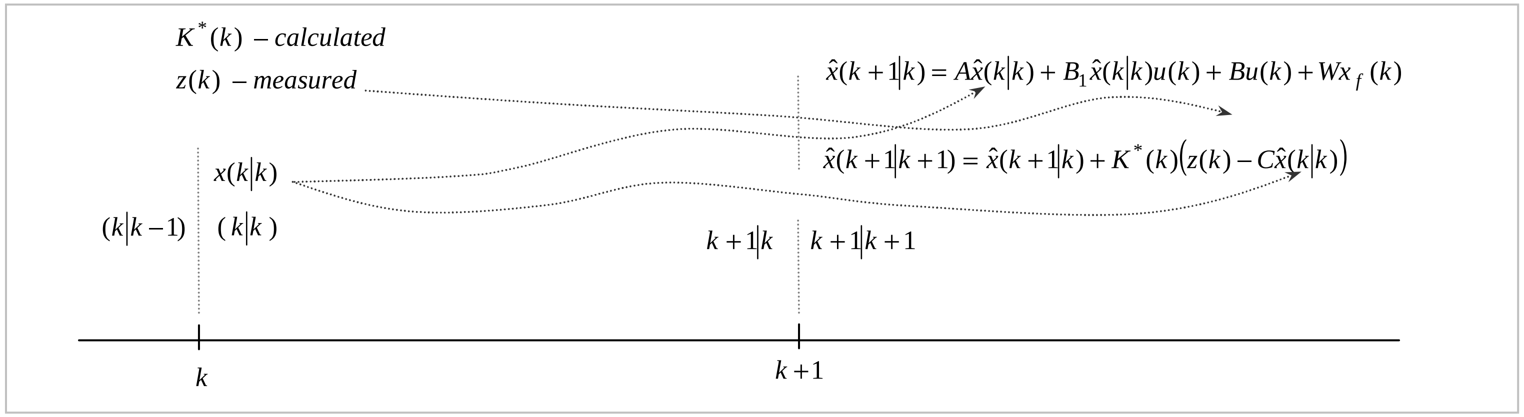
The statistical methods are again used to connect the two estimates at *prediction* and *correction* moments. The methods combine *a posteriori*, moment $(k+1|k+1)$ and the *a priori*, moment $(k+1|k)$ to produce the current estimate. The connection requires the use of covariance matrices, the *a posteriori state estimate error covariance* $P(k+1|k+1)$ at moment $(k+1|k+1)$ and the *a priori state estimate error covariance matrix* $P(k+1|k)$. From the derivation of the equations for use in optimal estimates, it has been shown that minimization of the estimation error through optimization criterion, $J(k+1)$ is achieved through minimization of the *a posteriori covariance matrix* $P(k+1|k+1)$, Equations

(9.97–9.99). This *a posteriori* covariance matrix is dependent on the *a priori* covariance matrix.

The final solution is dependent on two sets of equations, that, at the prediction stage and at the correction stage (Figures 9.5 and 9.6):

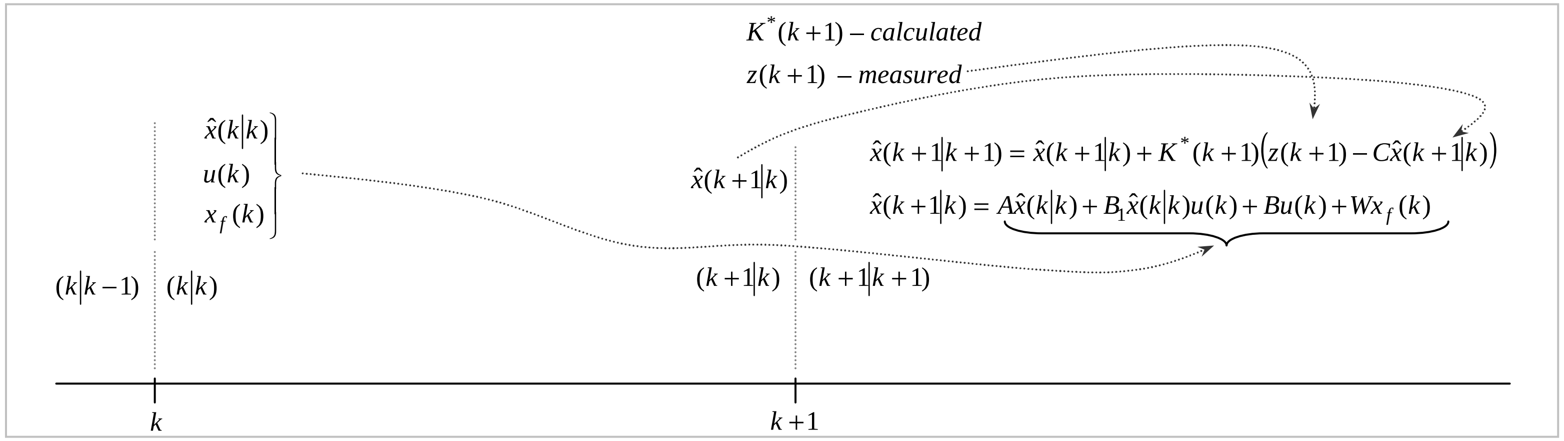
1) the first set of equations produces the *a priori estimate covariances*, and,
2) the second set is used to determine the *a posteriori estimate covariances*.

The connection between the two sets is via the output measurements $z(k+1)$, the filter gains $K^*(k+1)$, and the process noise covariances $W^*(k+1|k+1)$.



$K^*(k)$ – calculated
$z(k)$ – measured
$\hat{x}(k+1|k) = A\hat{x}(k|k) + B_1\hat{x}(k|k)u(k) + Bu(k) + Wx_f(k)$
$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K^*(k)\left(z(k) - C\hat{x}(k|k)\right)$
$x(k|k)$
$(k|k-1)$
$(k|k)$
$k+1|k$
$k+1|k+1$
k
$k+1$

Figure 9.5. First method for state estimation using general formulation of the Kalman filter



$K^*(k+1)$ – calculated
$z(k+1)$ – measured
$\hat{x}(k|k)$
$u(k)$
$x_f(k)$
$\hat{x}(k+1|k)$
$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K^*(k+1)\left(z(k+1) - C\hat{x}(k+1|k)\right)$
$\hat{x}(k+1|k) = A\hat{x}(k|k) + B_1\hat{x}(k|k)u(k) + Bu(k) + Wx_f(k)$
$(k|k-1)$
$(k|k)$
$(k+1|k)$
$(k+1|k+1)$
k
$k+1$

Figure 9.6. Second method for state estimation using direct optimization technique

9.7.3. The summary of equations for the two methods of solution

For both methods, the procedure is more or less the same:

- provide the initial estimate, at the beginning of the procedure,
- determine the current estimate based on the output measurement,
- determine the prediction error,
- calculate the filter gain using previous prediction error covariance,
- calculate the prediction error covariance,
- optimize the prediction error covariance (for optimal state estimates).

The differences are:

1) *Moment of measurement* – in the first method, the measurement is taken at moment $(k|k)$, and in the second method, it is done at the moment $(k+1|k+1)$,
2) *In the derivation of the a priori and a posteriori equations* – in the first method, the *a priori* and *a posteriori* estimates are considered to be expressed only by one equation that includes both the time update and the measurement update.
   In the second method two separate equations are considered, one for the time update and second for measurement update.
3) *The optimization of the criterion is done in two different ways* – in the first method by making the term containing the Kalman gain equal to zero; in the second method deriving the first derivation of the criterion according to the Kalman gain and equalizing the obtained expression to zero.

Table 9.1: **The summary of equations for the two methods of solution for determining Kalman filter gain**

Method of solution and description of the equation for optimal state estimates	Equation used in the procedure
First method	General formulation equations
The *a priori estimate* at moment k:	$\hat{x}(k+1\|k) = A\hat{x}(k\|k) + B_1\hat{x}(k\|k)u(k) + Bu(k) + Wx_f(k)$ (9.44)
The *a priori estimation error* at the moment $(k+1)$:	$e(k+1\|k) = \hat{x}(k+1\|k) - x(k\|k)$ (9.46)
The *a posteriori estimate* defined by $\hat{x}(k\|k)$ is known, obtainable from the output:	$\hat{x}(k+1\|k+1) = A\hat{x}(k+1\|k) + K^*(k)\left[z(k) - C\hat{x}(k\|k)\right]$ (9.47)
The *a posteriori* covariance of the estimation error at moment $(k+1)$:	$P(k+1\|k+1) \to \min$ (9.50)
The error at the moment $(k+1\|k+1)$ for the *a posteriori* covariance matrix and criterion calculation:	$e(k+1\|k+1) = \hat{x}(k+1\|k+1) - x(k+1)$ (9.51)
The final error equation used to determine the *a posteriori state estimate* at moment $(k+1\|k+1)$:	$e(k+1\|k+1) = \left[Ae(k\|k) - K^*(k)C\right]\ e(k\|k) + B_1e(k\|k) + K^*(k)v(k\|k) - w(k\|k)$ (9.53)
Equation for the *a posteriori* covariance matrix:	$P(k+1\|k+1) = E\left\{e(k+1\|k+1)e(k+1\|k+1)^T\right\} = J(k+1)$ (9.54)

The solution to the *a posteriori* covariance matrix after substituting the error equation:	$J(k+1)=[A-K^*(k)C]P(k\|k)[A-K^*(k)C]^T+$ $+B_1u(k)P(k\|k)[A-K^*(k)C]^T+[A-K^*(k+1)C]P(k\|k)[B_1u(k)]^T+$ $+B_1u(k)P(k\|k)[B_1u(k)]^T+K^*(k)V^*(k\|k)K^*(k)^T+W^*(k\|k)$ (9.56)
$P(k\|k)$ is the covariance matrix of the estimation error at the current moment k:	$P(k\|k)=E\{e(k\|k)e(k\|k)^T\}$ (9.56)
Minimization of the criterion function is the same as the minimization of the *covariance matrix of the prediction error* at the same moment of time:	$P(k+1\|k+1)\to\min\cong J(k+1)\to\min$ (9.57)
A block diagonal matrix used in covariances calculation $Q(k\|k)$:	$CP(k\|k)C^T+V^*(k\|k)=Q(k\|k)$ (9.60)
The final expression for the *a posteriori covariance* matrix:	$P(k+1\|k+1)=AP(k\|k)A^T+2B_1u(k)P(k\|k)A^T+B_1u(k)P(k\|k)u(k)^T B_1^T-$ $-K^*(k)\left[AP(k\|k)C^T+B_1u(k)C^T\right]^T+W^*(k\|k)$ (9.61)
The Kalman filter gain used in calculating the *a posteriori covariance* matrix:	$K^*(k)=[AP(k\|k)C^T+B_1u(k)C^T]\ [CP(k\|k)C^T+V^*(k\|k)]^{-1}$ (9.67)
Second method	Direct optimization of the estimation error
The criterion for minimizing the variances of state estimates from their true values:	$J(k)=\frac{1}{2}E\{[\hat{x}(k+1)-x(k+1)]^2\}\to\min$ (9.69)

The equation for *a priori estimate error* at moment $(k+1\|k)$:	$e(k+1\|k) = \hat{x}(k+1\|k) - x(k+1)$ (9.70)
The *a priori state estimate* at moment $(k+1\|k)$:	$\hat{x}(k+1\|k) = A\hat{x}(k\|k) + B_1\hat{x}(k\|k)u(k) + Wx_f(k)$ (9.78)
The final *a priori estimate error* at moment $(k+1\|k)$:	$e(k+1\|k) = Ae(k\|k) + B_1 e(k\|k)u(k) - w(k), \quad with\ u(k) \in R^1$ (9.80)
The *a priori error covariance matrix* at the current moment $(k+1\|k)$:	$P(k+1\|k) = AP(k\|k)A^T + B_1 u(k)P(k\|k)A^T + Au(k)P(k\|k)B_1^T + B_1 u(k)^2 P(k\|k)B_1^T + +W^*(k)$ (9.87)
The *a posteriori state estimate* at the next moment $(k+1\|k+1)$:	$\hat{x}(k+1\|k+1) = [I - K^*(k+1)C]\hat{x}(k+1\|k) + K^*(k+1)Cx(k+1) + K^*(k+1)v(k+1)$ (9.88)
The *a posteriori estimate error* at the moment $(k+1\|k+1)$:	$e(k+1\|k+1) = [I - K^*(k+1)C]e(k+1\|k) + K^*(k+1)v(k+1)$ (9.90)
The equation for calculation of the *a posteriori estimate covariance matrix*:	$P(k+1\|k+1) = E\left\{\begin{matrix} \left[I - K^*(k+1)C\right] e(k+1\|k)e(k+1\|k)^T \left[I - K^*(k+1)C\right]^T + \\ + \left[I - K^*(k+1)C\right] e(k+1\|k)v(k+1)^T K^*(k+1)^T + \\ + K^*(k+1)v(k+1)e(k+1\|k)^T \left[I - K^*(k+1)C\right]^T + \\ + K^*(k+1)v(k+1)v(k+1)^T K^*(k+1) \end{matrix}\right\}$ (9.91)

The solution to the *a posteriori estimate covariance matrix* at moment $(k+1\|k+1)$:	$P(k+1\|k+1)=[I-K^*(k+1)C]P(k+1\|k)[I-K^*(k+1)C]^T + {} + K^*(k+1)V^*(k+1)K^*(k+1)^T$ (9.96)
Criterion to *minimize the error covariance matrix* so as to optimize the Kalman filter gain:	$P(k+1\|k+1)=J(k+1)\rightarrow \min$ (9.97)
Equation (9.97) expressed differently:	$\begin{cases}\min\{J(k+1)\}\\ K(k+1)\end{cases} = \begin{cases}\min & \begin{Bmatrix}[I-K^*(k+1)C]P(k+1\|k)[I-K(k+1)C]^T + \\ + K^*(k+1)V^*(k+1)K^*(k+1)\end{Bmatrix}\\ K(k+1), & \textit{for } k=1,2,3\end{cases}$ (9.98)
Obtaining the optimal gain on the basis of the first derivative of the *criterion functional* according to the Kalman gain:	$\dfrac{\partial J(k+1)}{\partial K^*(k+1)}=-P(k+1\|k)C^T+K^*(k+1)\left[CP(k+1\|k)C^T+V^*(k+1)\right]=0$ (9.102)
Equations that are part of solution to (9.102):	$CP(k+1\|k)C^T K^*(k+1)^T = K^*(k+1)CP(k+1\|k)C^T$ (9.103) $V^*(k+1)K^*(k+1)^T = K^*(k+1)V^*(k+1)^T = K^*(k+1)V^*(k+1)$ (9.106)
The *optimal Kalman filter gain*:	$K^*(k+1)=P(k+1\|k)C^T\left[CP(k+1\|k)C^T+V^*(k+1)\right]^{-1}$ (9.107)
The *a posteriori covariance matrix* is determined from the *a priori covariance matrix*:	$P(k+1\|k+1)=\left[I-K^*(k+1)C\right]P(k+1\|k)$ (9.109)

9.8. Algorithms and MATLAB software program developed for the calculation of the Kalman filter gain for the bilinear model of the continuous countercurrent ion exchange (CCIX) process

Algorithms for both methods developed for the Kalman filter are presented. The Kalman filter, mainly, has two sets of equations:

- *Time update equations* (prediction) and
- *Measurement update equations* (correction).

The schematic representation of this operation for the bilinear process is shown in Figure 9.7 following the presentation in Welch & Bishop (2006:6).

9.8.1. Algorithm for the first general method

1) Set the initial values for the:
   - o Initial time $k = k_0$, initial state $x(k|k) = x(k_0|k_0) = x_{k_0}$, initial state estimate $\hat{x}(k|k)$, initial covariance matrix $P(k_0|k_0) = P_{k_0}$
2) Time update calculations (prediction)
   - o Set the trajectory period, $k = \overline{1, K}$
   - o Calculate the state estimate from Equation (9.44)

     $$\hat{x}(k+1|k) = A\hat{x}(k|k) + B_1\hat{x}(k|k)u(k) + Bu(k) + Wx_f(k)$$
   - o Calculate the error $e(k+1|k)$ at the moment $(k+1|k)$

     $$e(k+1|k) = \hat{x}(k+1|k) - x(k|k)$$
   - o Update the error covariance matrix $P(k|k)$ at the moment $(k+1|k)$
3) Measurement update calculations (error correction)
   - o Obtain the output measurement at the moment $(k|k)$

     $$z(k) = Cx(k|k) + v(k)$$
   - o Calculate the measurement noise covariance matrix $V^*(k|k)$ based on the measurement noise
   - o Calculate the filter gain matrix $K^*(k)$ using Equation (9.67)

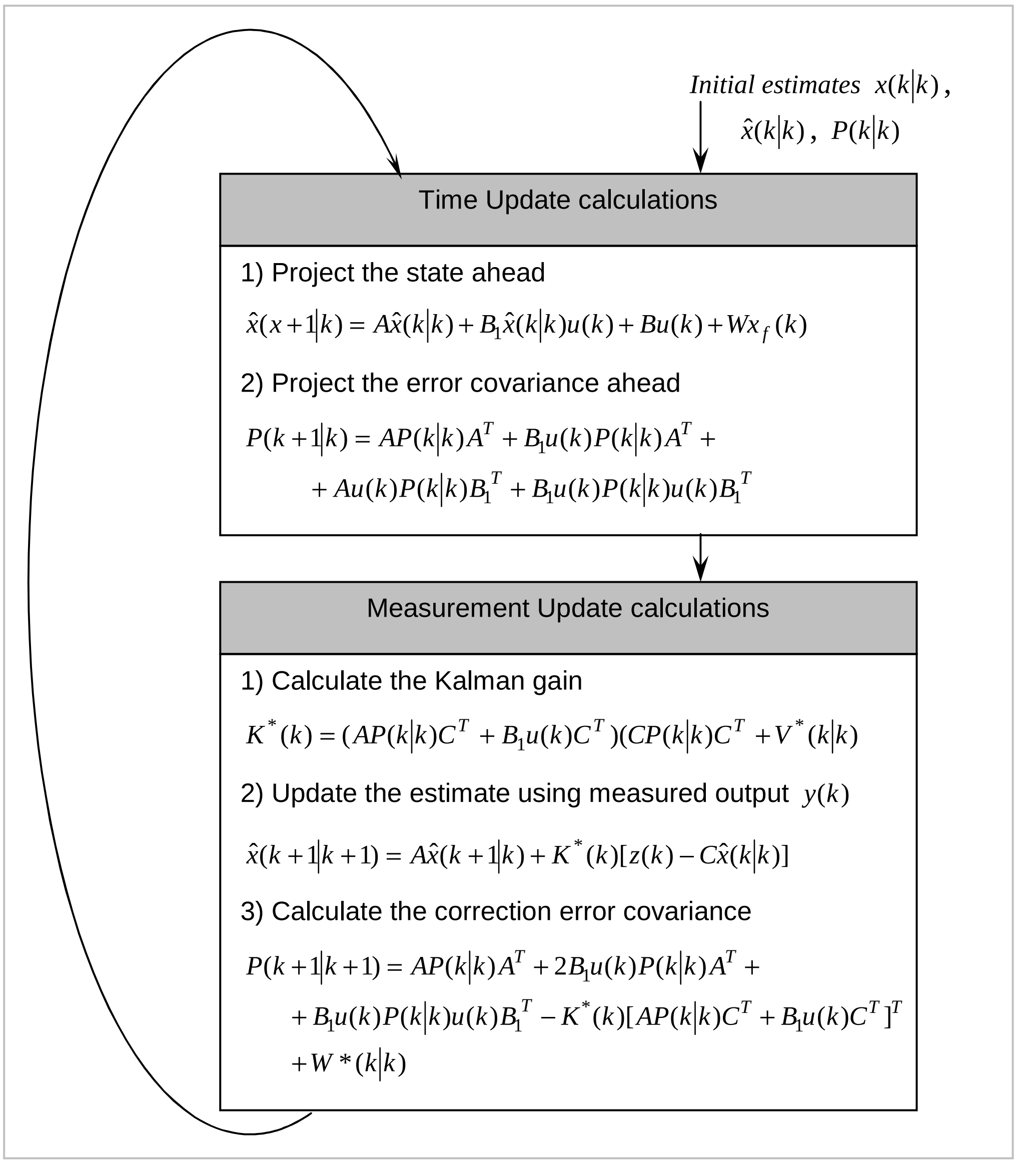
     $$K^*(k) = [AP(k|k)C^T + B_1u(k)C^T]\ [CP(k|k)C^T + V^*(k|k)]^{-1}$$
   - o Update the state estimate at the moment $(k+1|k)$

     $$\hat{x}(k+1|k+1) = A\hat{x}(k+1|k) + K^*(k)\left[z(k) - C\hat{x}(k|k)\right]$$
   - o Update the *a posteriori covariance matrix* to optimize the estimate

$$P(k+1|k+1) = J(k+1) \to \min$$

$$P(k+1|k+1) = AP(k|k)A^T + 2B_1u(k)P(k|k)A^T + B_1u(k)P(k|k)u(k)^T {B_1}^T -$$
$$- K^*(k)\left[AP(k|k)C^T + B_1u(k)C^T\right]^T + W^*(k|k)$$

- o The *a posteriori covariance matrix* is used in the next calculation of the next *a priori covariance matrix*.
- o The calculation goes back to 2), the *Time Update* and then followed by 3), *Measurement Update* calculations until the end of the projection time period k when K^{th} moment is reached.



Initial estimates $x(k|k)$, $\hat{x}(k|k)$, $P(k|k)$
Time Update calculations
1) Project the state ahead
$\hat{x}(x+1|k) = A\hat{x}(k|k) + B_1\hat{x}(k|k)u(k) + Bu(k) + Wx_f(k)$
2) Project the error covariance ahead
$P(k+1|k) = AP(k|k)A^T + B_1u(k)P(k|k)A^T + + Au(k)P(k|k){B_1}^T + B_1u(k)P(k|k)u(k){B_1}^T$
Measurement Update calculations
1) Calculate the Kalman gain
$K^*(k) = (AP(k|k)C^T + B_1u(k)C^T)(CP(k|k)C^T + V^*(k|k)$
2) Update the estimate using measured output $y(k)$
$\hat{x}(k+1|k+1) = A\hat{x}(k+1|k) + K^*(k)[z(k) - C\hat{x}(k|k)]$
3) Calculate the correction error covariance
$P(k+1|k+1) = AP(k|k)A^T + 2B_1u(k)P(k|k)A^T + + B_1u(k)P(k|k)u(k){B_1}^T - K^*(k)[AP(k|k)C^T + B_1u(k)C^T]^T + W*(k|k)$

Figure 9.7. **The Kalman filter presentation showing equations and the order of prediction and correction procedure for the general method**

9.8.2. Algorithm and flowchart for the direct optimization method

1) Set the initial values for the:

- Initial time $k = k_0$, initial state $x(k|k) = x(k_0|k_0) = x_{k_0}$, initial state estimate $\hat{x}(k|k)$, initial covariance matrix $P(k_0|k_0) = P_{k_0}$

2) Time update calculations (prediction)

- Set the trajectory period, $k = \overline{1, K}$
- Calculate the state estimate from Equation (9.44)

$$\hat{x}(k+1|k) = A\hat{x}(k|k) + B_1\hat{x}(k|k)u(k) + Bu(k) + Wx_f(k)$$

- Calculate the error $e(k+1|k)$ at the moment $(k+1|k)$

$$e(k+1|k) = Ae(k|k) - B_1 e(k|k)u(k) - w(k)$$

- Update the error covariance matrix $P(k|k)$ at the moment $(k+1|k)$

$$P(k+1|k) = AP(k|k)A^T + B_1 u(k)P(k|k)A^T + \\ + Au(k)P(k|k)B_1^T + B_1 u(k)^2 P(k|k)B_1^T + W^*(k)$$

3) Measurement update calculations (error correction)

- Obtain the output measurement at the moment $(k+1|k+1)$

$$z(k+1|k+1) = Cx(k+1) + v(k+1|k+1)$$

- Calculate the measurement noise covariance matrix $V^*(k|k)$ based on the measurement noise
- Calculate the filter gain matrix $K^*(k+1)$ using Equation (9.106)

$$K^*(k+1) = P(k+1|k)C^T[CP(k+1|k)C^T + V^*(k+1)]^{-1}$$

- Update the state estimate at the moment $(k+1|k+1)$, Equation (9.88)

$$\hat{x}(k+1|k+1) = [I - K^*(k+1)C]\hat{x}(k+1|k) + \\ + K^*(k+1)[Cx(k+1) + v(k+1)]$$

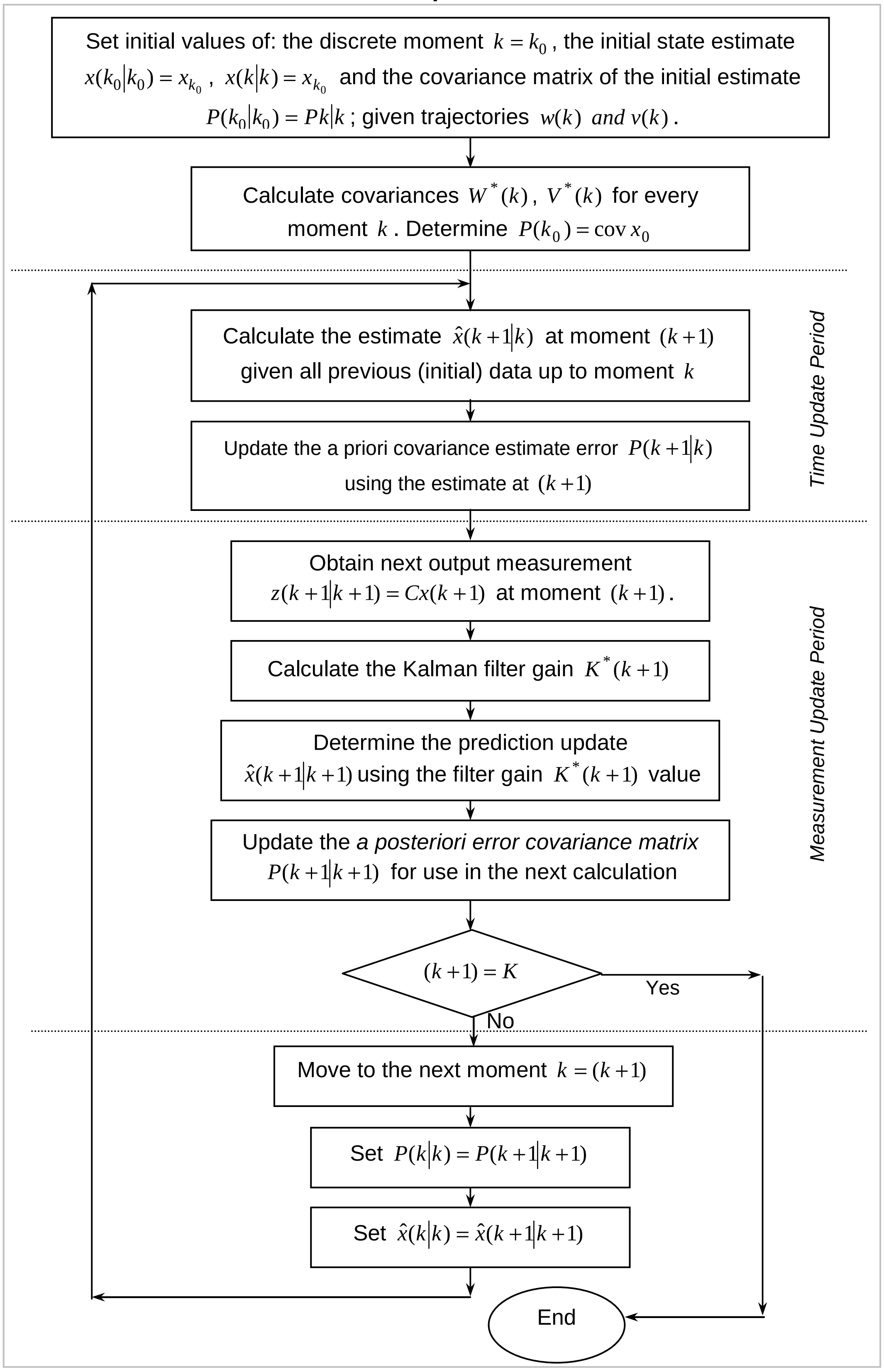
- Update the *a posteriori covariance matrix* to optimize the estimate

$$P(k+1|k+1) = J(k+1) \rightarrow \min$$

$$P(k+1|k+1) = [I - K^*(k+1)C]P(k+1|k)$$

- The *a posteriori covariance matrix* is used in the next calculation of the next *a priori covariance matrix*.
- The calculation goes back to 2), the *Time Update* and then followed by 3), *Measurement Update* calculations until the end of the projection time period when K is reached.

9.8.3. The Kalman filter software implementation flowchart



Set initial values of: the discrete moment $k = k_0$, the initial state estimate $x(k_0|k_0) = x_{k_0}$, $x(k|k) = x_{k_0}$ and the covariance matrix of the initial estimate $P(k_0|k_0) = Pk|k$; given trajectories $w(k)$ *and* $v(k)$.
Calculate covariances $W^*(k)$, $V^*(k)$ for every moment k. Determine $P(k_0) = \operatorname{cov} x_0$
Calculate the estimate $\hat{x}(k+1|k)$ at moment $(k+1)$ given all previous (initial) data up to moment k
Update the a priori covariance estimate error $P(k+1|k)$ using the estimate at $(k+1)$
Time Update Period
Obtain next output measurement $z(k+1|k+1) = Cx(k+1)$ at moment $(k+1)$.
Calculate the Kalman filter gain $K^*(k+1)$
Determine the prediction update $\hat{x}(k+1|k+1)$ using the filter gain $K^*(k+1)$ value
Update the *a posteriori error covariance matrix* $P(k+1|k+1)$ for use in the next calculation
Measurement Update Period
$(k+1) = K$
Yes
No
Move to the next moment $k = (k+1)$
Set $P(k|k) = P(k+1|k+1)$
Set $\hat{x}(k|k) = \hat{x}(k+1|k+1)$
End

Figure 9.8. Flowchart for implementing the Kalman filter algorithm used for state estimation of the bilinear model for the CCIX plant (based on method for direct optimization with measurements at $(k+1|k+1)$

The procedure for the direct optimization method is implemented in MATLAB software using the flowchart in Figure 9.8. The MATLAB code is listed in APPENDIX E.

9.9. Experiments and results

Experiments were conducted to observe the behaviour of the designed filter under the influence of both the *system and measurement noises*; the *combination of their variations of noises* and *initial values of the state vector*. The criterion for evaluating the success of the experiments is the *least square error* between the measured states and the estimated states, $J(e)=\sum_{k=1}^{K}\left\|e(k)\right\|^{2}$ and the *overall processing time*.

The following sets of experiments have been conducted:

- Experiment for changing system noise and a constant measurement noise,
- Experiment for changing measurement noise and a constant system noise,
- Experiment of both system and measurement noises changing at the same rate and direction and
- Experiment of the changing initial values for the estimated states and the constant initial system states.

Figure 9.9 represents smoothened real data obtained from the study of continuous countercurrent ion exchange (CCIX) of Hendry, (1982). This figure is consistent for all experiments as presented by all the sets of experiments from Table 9.2 to Table 9.5. The Kalman filter gains behaviour has been presented for each set of experiments. Its corrective nature is demonstrated for all cases, this clearly demonstrates the success of the developed method of the Kalman filter for state estimation of the bilinear system considered.

Experiment conditions:

1) The system noise covariance: $E\left\{w(k)w(j)^{T}\right\}=\begin{cases}0, & k\neq 0\\ W^{*}(k), & k\neq j\equiv E\left\{w(k)w(k)^{T}\right\}\end{cases}$

where $W^{*}(k)\in R^{N\times N}$ a diagonal matrix,

2) The measurement noise covariance: $E\left\{v(k)v(j)^{T}\right\}=\begin{cases}0, & k\neq j\\ V^{*}(k), & k\neq j\end{cases}$

where $V^{*}\in R^{l\times l}$,

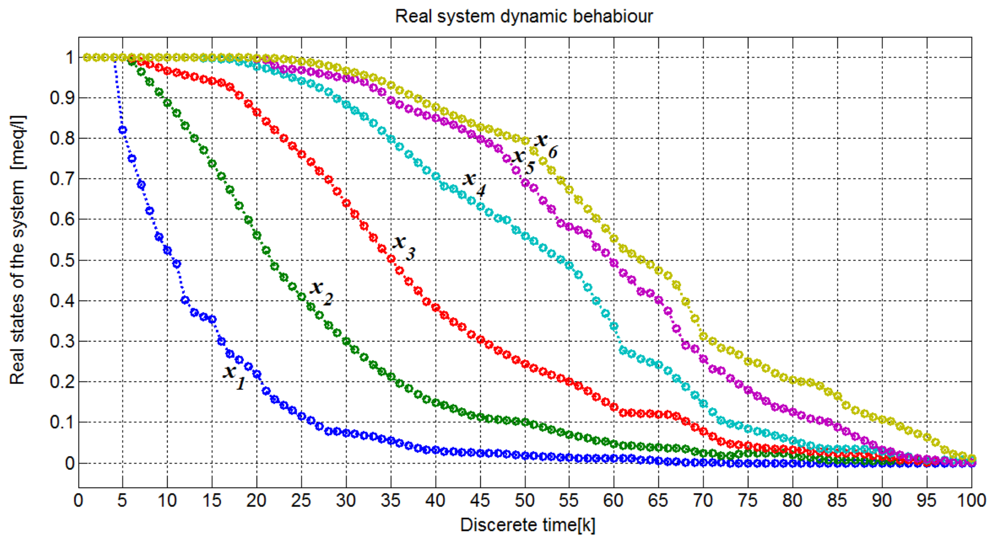


Figure 9.9. **CCIX process real system dynamic behaviour**

3) Requirements for the noise characteristics is to preserve the dialog structure of the covariance matrices.

4) The noise variables must not be corrected, then $E\{v(k)w(k)^T\}=0,$ for all k.

5) The two noise variables $w(k)$ and $v(k)$ must be stochastic noises, and

6) At $k=0$, $P(k_0)=E\{x(0)x(0)^T\}$, where $x(0)$ is the average of the stochastic process; and $P\in R^{N\times N}$ is not the function of time for the initial moment.

The first set of experiments (the changing amplitude of the system noise at constant measurement noise)

The first set of experiments in Table 9.2 is based on *the changing system noise* under the assumption that the measurement noise stays constant. The aim of the experiment is to observe how the system noise affects the filter at constant measurement noise. The filter gain, estimated state and estimation error are measured. The results are presented in Figures 9.10–9.13. This set of experiments considers a scenario where system noise maybe higher than and or increasing in comparison to the measurement noise.

Table 9.2: **Experiment for changing the amplitude of the system noise at a constant measurement noise**

Experiment set	System and measurement noises		Changing system noise			
1	$w(k)'$	$v(k)'$	Run 1	Run 2	Run 3	Run 4
	Details below	0.8138	$0.1\times w(k)$	$0.5\times w(k)$	$1.0\times w(k)$	$10\times w(k)$

$w(k)$, see APPENDIX E

The best case scenario:

$$W^*(k) = 1.0\times10^{-005}\begin{bmatrix} 0.0794 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0394 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1203 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0551 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0584 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0623 \end{bmatrix}$$

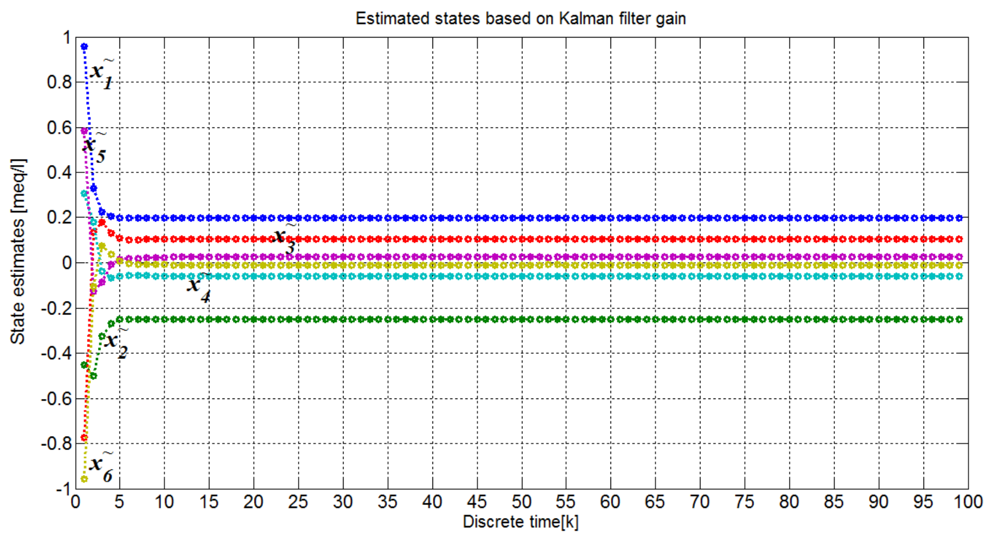


Figure 9.10. The estimated states for the CCIX process for the best case scenario at $w(k) = 0.1\times w(k)$ Run 1 of Table 9.2.

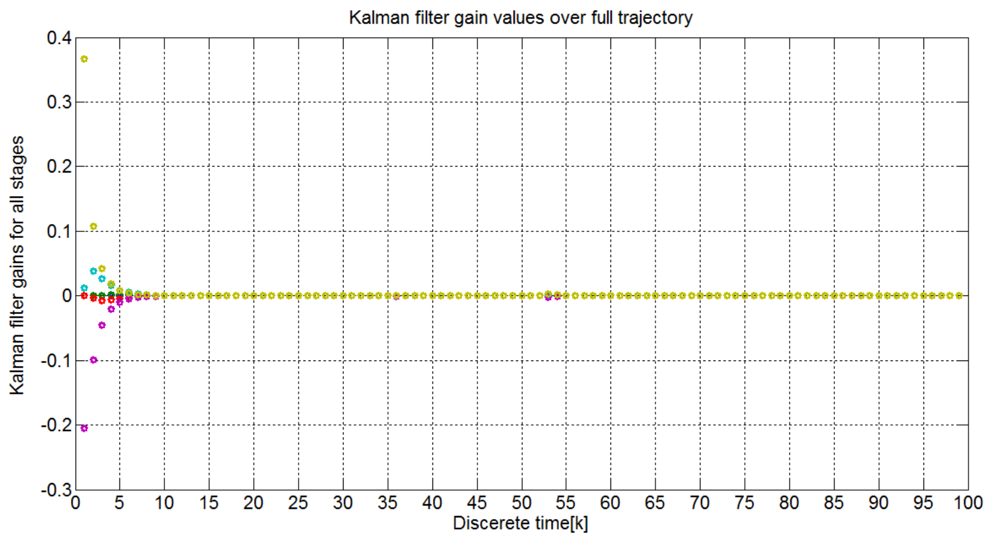


Figure 9.11. Kalman filter gain values for the best case scenario at $w(k) = 0.1\times w(k)$ Run 1 of Table 9.2.

The worst case scenario:

$$W^*(k) = \begin{bmatrix} 79.4027 & 0 & 0 & 0 & 0 & 0 \\ 0 & 39.3813 & 0 & 0 & 0 & 0 \\ 0 & 0 & 120.3064 & 0 & 0 & 0 \\ 0 & 0 & 0 & 55.0874 & 0 & 0 \\ 0 & 0 & 0 & 0 & 58.3892 & 0 \\ 0 & 0 & 0 & 0 & 0 & 62.2660 \end{bmatrix}$$

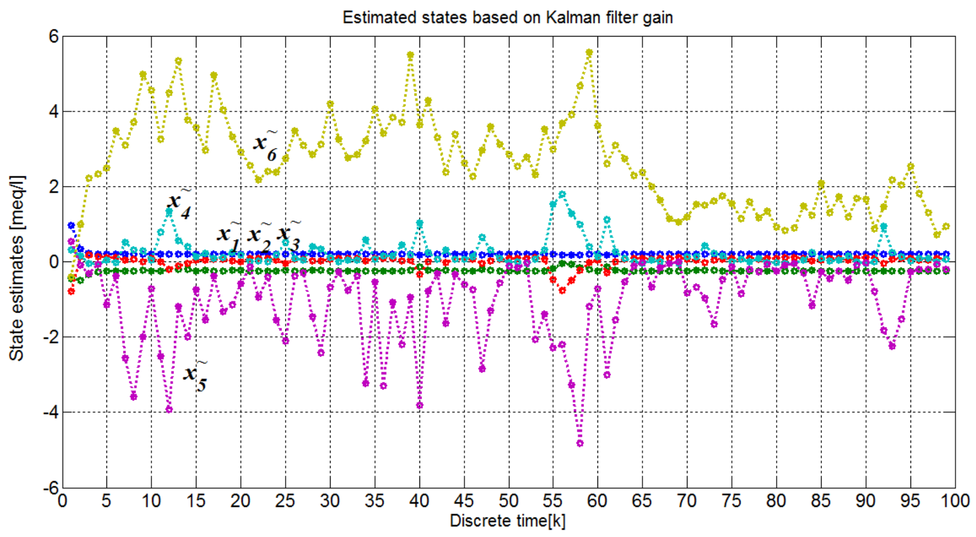


Figure 9.12. The estimated states for the CCIX process for the worst case at $w(k) = 10 \times w(k)$ Run 4 of Table 9.2.

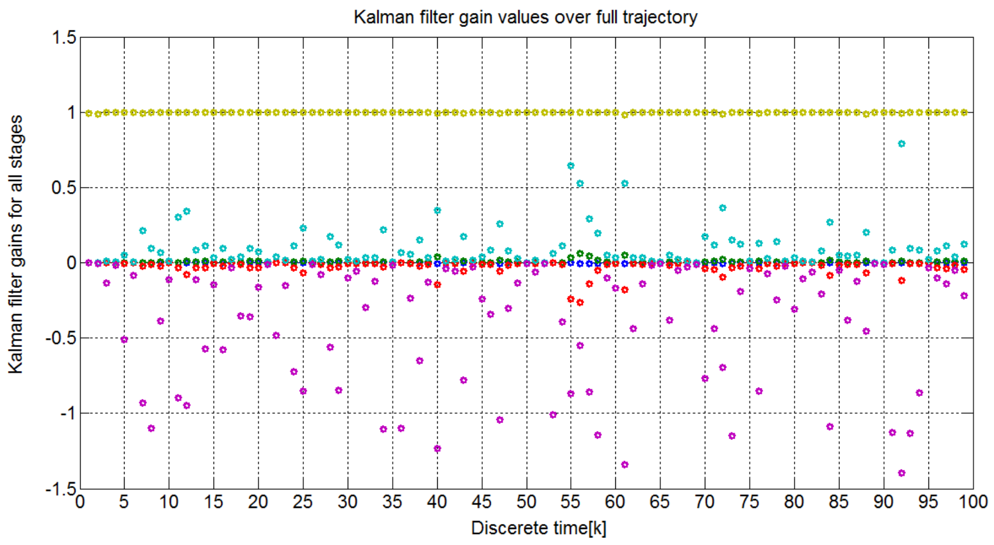


Figure 9.13. Kalman filter gain values for the worst case at $w(k) = 10 \times w(k)$ Run 4 of Table 9.2.

Table 9.3: Results of the first set of experiments for changing the system noise at constant measurement noise

Estimation criterion the least squares error and the estimation run time for the first set of experiments		
Case 1: (First set of experiments)	$J(e)$	Run time (t_{calc}) [sec]
Run 1	238.9205	0.0150
Run 2	238.0485	0.0160
Run 3	228.4817	0.0310
Run 4	1.0293e+003	0.0320

Results and discussion of the first set of experiments

The filter responds very well to noise changes; the effect of system noise is demonstrated in Figures 9.10–9.13. At lower system noise, the filter seems to produce best response according to the criteria, Table 9.3. At the lower system noise, the filter gains tend to be extremely small, close to zero, Figure 9.11.

The second set of experiments (the changing measurement noise at constant system noise)

The second set of experiments is that of *changing measurement noise* while keeping all other parameters constant, and using the best performance system noise $w(k)$ from the first set of experiments, Table 9.4. The results for this set of experiments are presented in Table 9.5 and Figures 9.14–9.17. The experiments consider the case where the measurement noise may be increasing while the system noise stays constant and or lower than the measurement noise.

Table 9.4: Experiment of the changing measurement noise and constant system noise

Experiment set	System and measurement noises		Changing system noise			
2	$w(k)'$	$v(k)'$	Run 1	Run 2	Run 3	Run 4
	$0.1\times w(k)$	0.8138	$0.1\times v(k)$	$0.5\times v(k)$	$1.0\times v(k)$	$10\times v(k)$

$w(k)$, see APPENDIX E

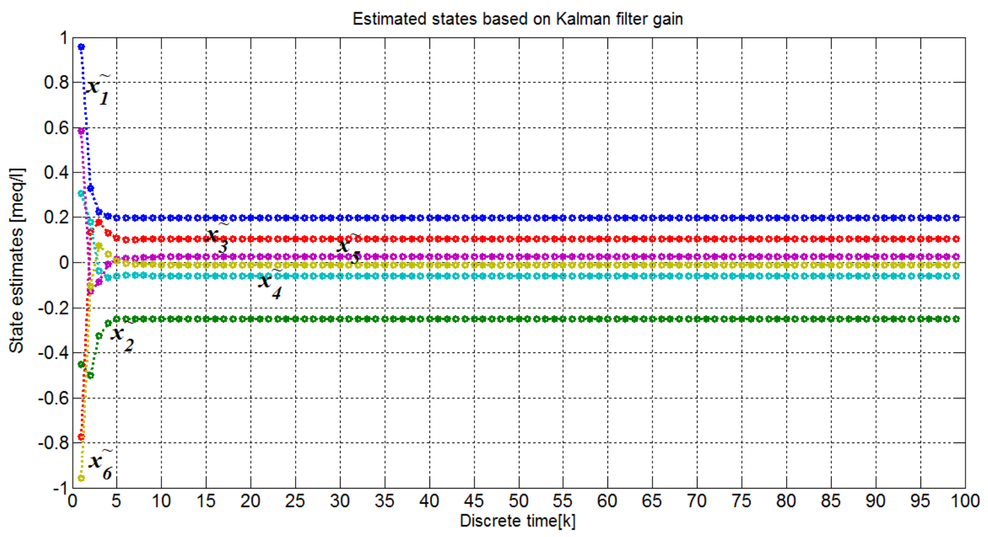


Figure 9.14. The estimated states for the CCIX process for the best case at $v(k) = 0.1 \times v(k)$ Run 1 of Table 9.4.

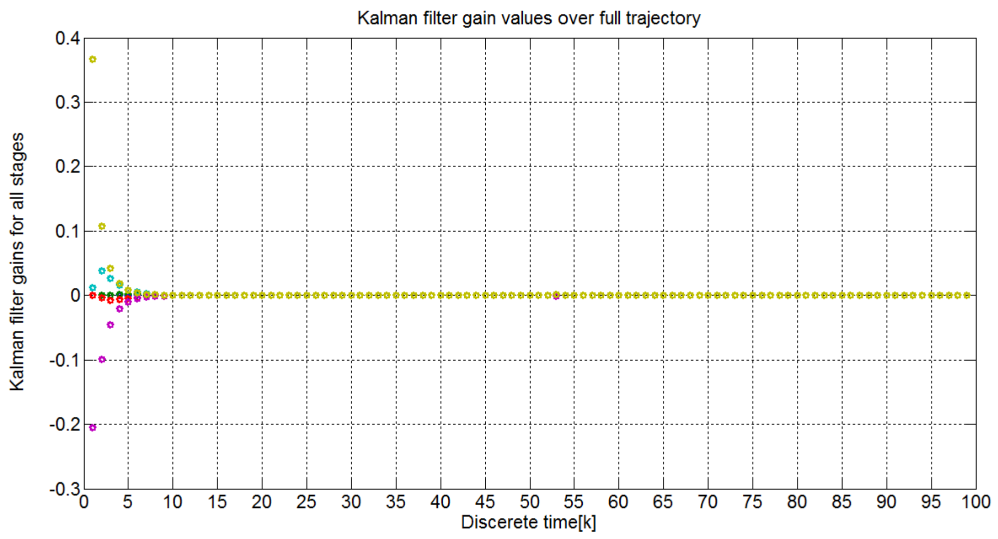


Figure 9.15. Kalman filter gain values for the best case at $v(k) = 0.1 \times v(k)$ Run 1 of Table 9.4.

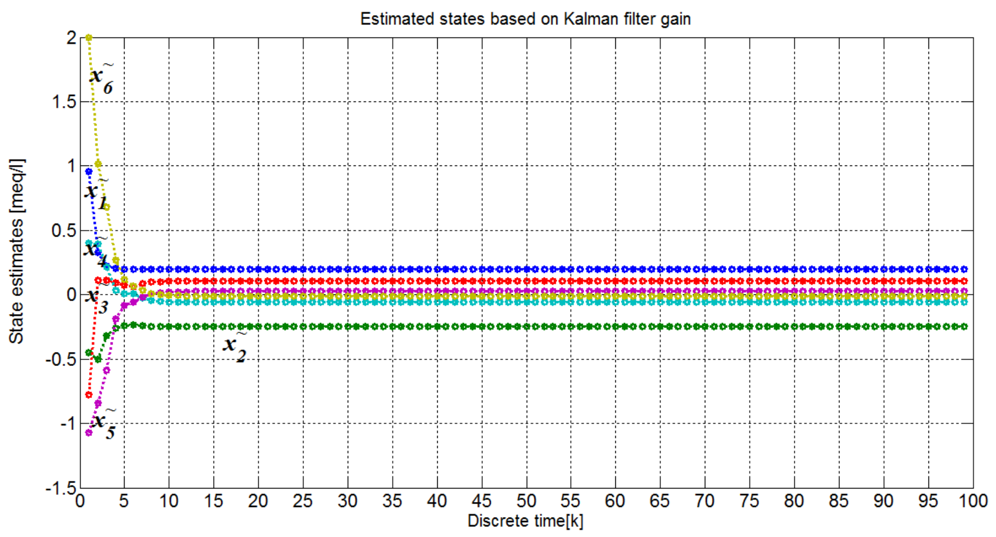


Figure 9.16. The estimated states for the CCIX process for the worst case scenario at $v(k) = 10 \times v(k)$ Run 4 of Table 9.4.

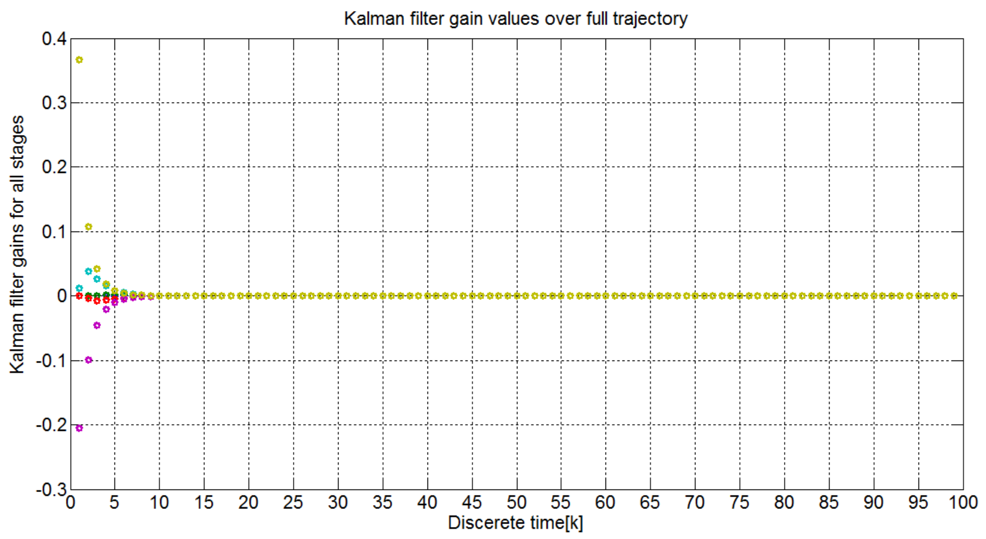


Figure 9.17. Kalman filter gain values for the worst case scenario at $v(k) = 10 \times v(k)$ Run 4 of Table 9.4.

Table 9.5: Results of the second set of experiments for changing the measurement noise at constant system noise

Estimation criterion the least squares error and the estimation run time for the second set of experiments		
Case 2: (Second set of experiments)	$J(e)$	Run time (t_{calc}) [sec]
Run 1	239.9425	0.0150
Run 2	239.4595	0.0310
Run 3	238.9205	0.0320
Run 4	241.5419	0.0150

Results and discussion of the second set of experiments

The filter response to the effects of measurement noise is demonstrated in Figures 9.14–9.17. The criteria for evaluation of the best results are presented in Table 9.5. The filter performs poorly at high measurement noise, Table 9.5. This is the same effect to that of system noise.

The third set of experiments (changing both the measurement and system noises – simultaneous increase or decrease)

The third set of experiments considers the both system and measurement noises increasing simultaneously. This behaviour is presented in Table 9.6 and the corresponding results are shown in Figures 9.18–9.21. For both noises increasing at the same rate and in the same direction, the filter performed better at lowest system and measurement noises.

Table 9.6: **Experiment of changing both system and measurement noises at the same rate and direction**

Experiment set	System and measurement noises		Simultaneously changing system and measurement noises			
3	$w(k)'$	$v(k)'$	Run 1	Run 2	Run 3	Run 4
	Details below	0.8138	$0.1\times v(k)$ $0.1\times w(k)$	$0.5\times v(k)$ $0.5\times w(k)$	$1.0\times v(k)$ $1.0\times w(k)$	$10\times v(k)$ $10\times w(k)$

$w(k)$, see APPENDIX E

For the best case results:

$$W^*(k) = 1.0\times10^{-005}\begin{bmatrix} 0.0794 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0394 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1203 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0551 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0584 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0623 \end{bmatrix}$$

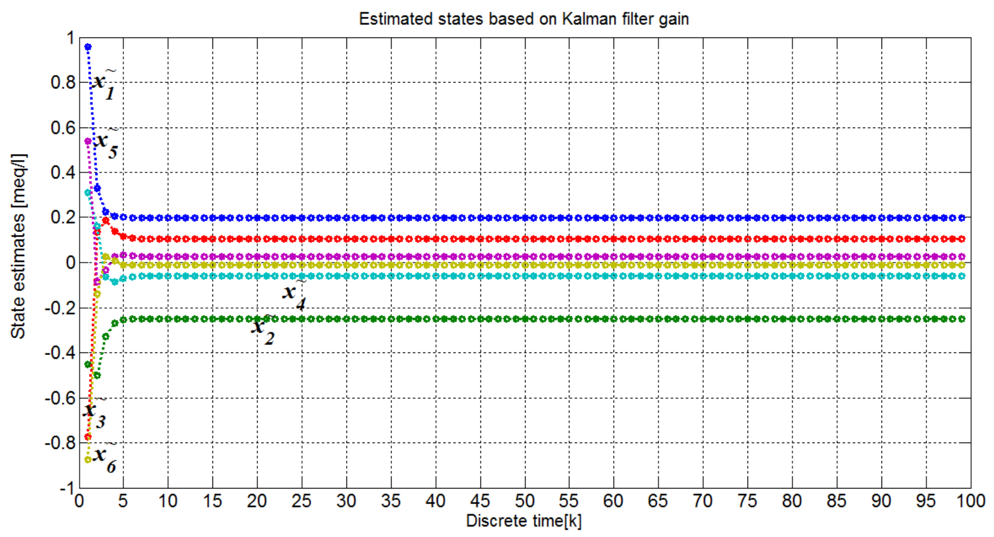


Figure 9.18. The estimated states for the CCIX process for the best case scenario at $w(k) = 0.1 \times w(k)$ and $v(k) = 0.1 \times v(k)$ Run 1 of Table 9.6.

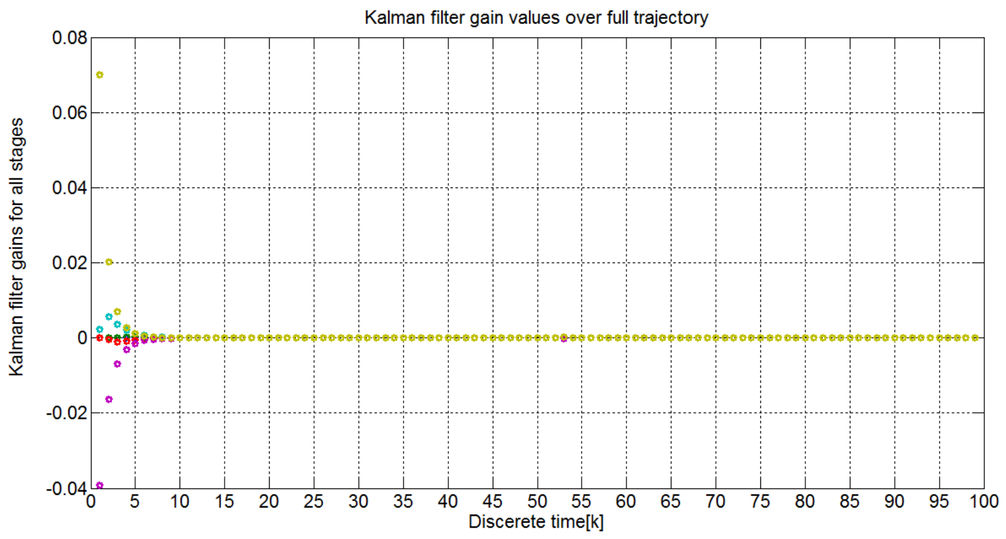


Figure 9.19. Kalman filter gain values for the best case scenario at $w(k) = 0.1 \times w(k)$ and $v(k) = 0.1 \times v(k)$ Run 1 of Table 9.6.

For the worst case scenario:

$$W^*(k) = \begin{bmatrix} 79.4027 & 0 & 0 & 0 & 0 & 0 \\ 0 & 39.3813 & 0 & 0 & 0 & 0 \\ 0 & 0 & 120.3064 & 0 & 0 & 0 \\ 0 & 0 & 0 & 55.0874 & 0 & 0 \\ 0 & 0 & 0 & 0 & 58.3892 & 0 \\ 0 & 0 & 0 & 0 & 0 & 62.2660 \end{bmatrix}$$

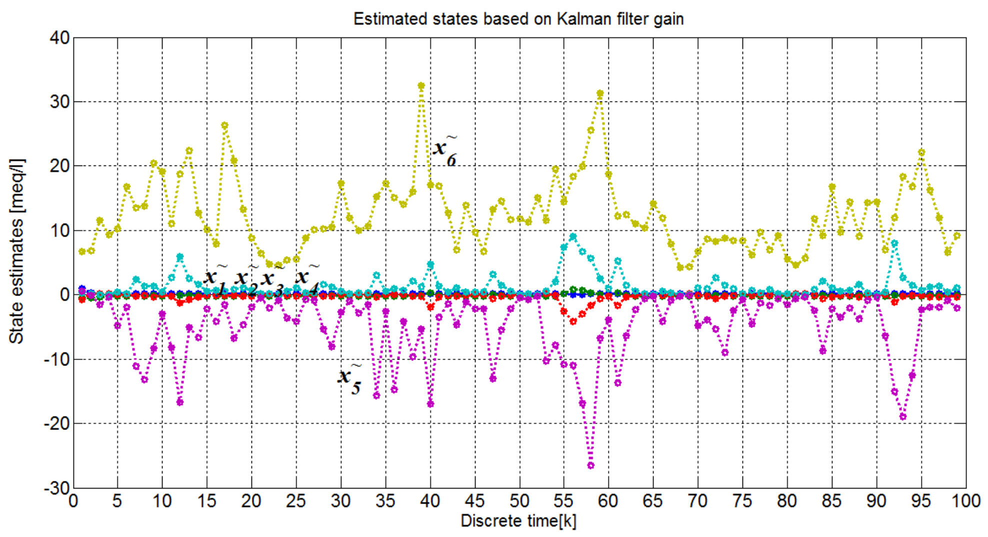


Figure 9.20. The estimated states for the CCIX process for the worst case at $w(k) = 10 \times w(k)$ and $v(k) = 10 \times v(k)$ Run 4 of Table 9.6.

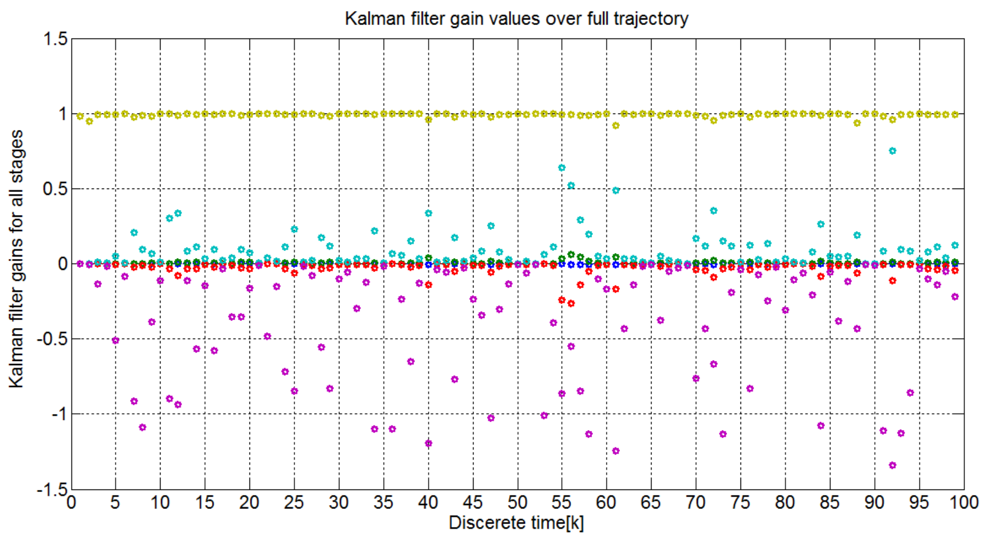


Figure 9.21. Kalman filter gain values for the worst case at $w(k) = 10 \times w(k)$ and $v(k) = 10 \times v(k)$ Run 4 of Table 9.6.

Table 9.7: Results of the third set of experiments for changing both the measurement and system noises simultaneously

Estimation criterion the least squares error and the estimation run time for the third set of experiments		
Case 3: (Third set of experiments)	$J(e)$	Run time (t_{calc}) [sec]
Run 1	239.8075	0.0150
Run 2	239.1048	0.0160
Run 3	233.9360	0.0320
Run 4	2.3083e+004	0.0160

Results and discussion of the third set of experiments

Results for both noises increasing simultaneously show that the system noise takes precedence over measurement noise, Table 9.7. At higher system noise the filter produced worse results at constant (or lower) measurement noise; but at both low noises the filter performed well, Figures 9.18–9.21.

The fourth set of experiments (changing the initial conditions of the estimated states)

The fourth set of experiments deals with *changing initial conditions of the estimated states* of the system at the constant system state initial conditions $x(k) = [1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0]^T$, and the reference initial estimated state of $\hat{x}(k) = [1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0]^T$, Table 9.8. This experiment uses the lowest values of $v(k)$ and $w(k)$ that produced best results in experiment one and two. The results for this experiment are presented in Figures 9.22–9.25 and in Table 9.9.

Table 9.8: **Experiment of the changing estimated state initial conditions**

Experiment set	System and measurement noises		Changing initial estimated state values			
4	$w(k)'$	$v(k)'$	Run 1	Run 2	Run 3	Run 4
	Details below	0.8138	$\hat{x}(k) = 0$	$1.0 \times \hat{x}(k)$	$5.0 \times \hat{x}(k)$	$10 \times \hat{x}(k)$

$w(k)$, see APPENDIX E

$$x(k) = [1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0]^T$$
$$\hat{x}(k) = [1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0]^T$$

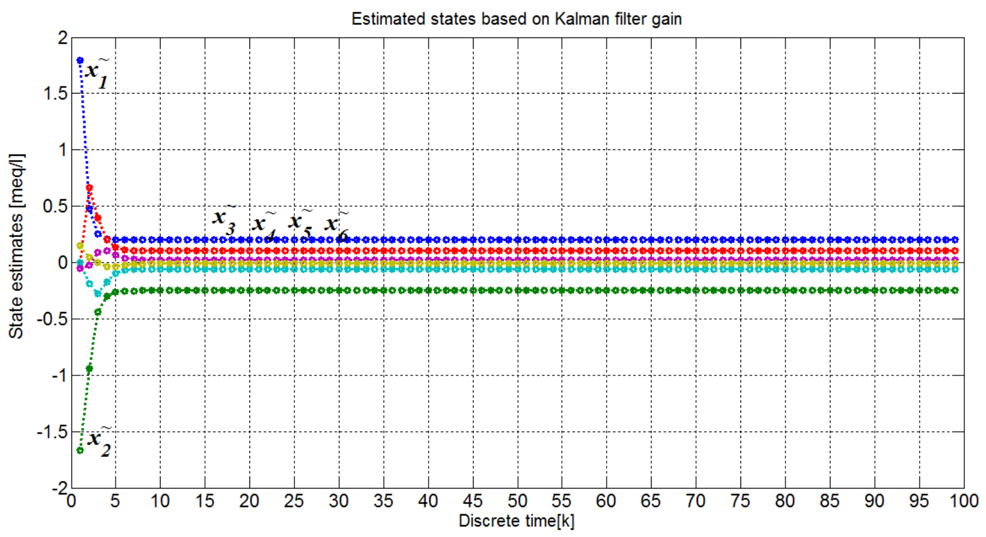


Figure 9.22. The estimated states for the CCIX process for the best case results at $\hat{x}(k) = 10 \times \hat{x}(k)$ Run 4 of Table 9.8.

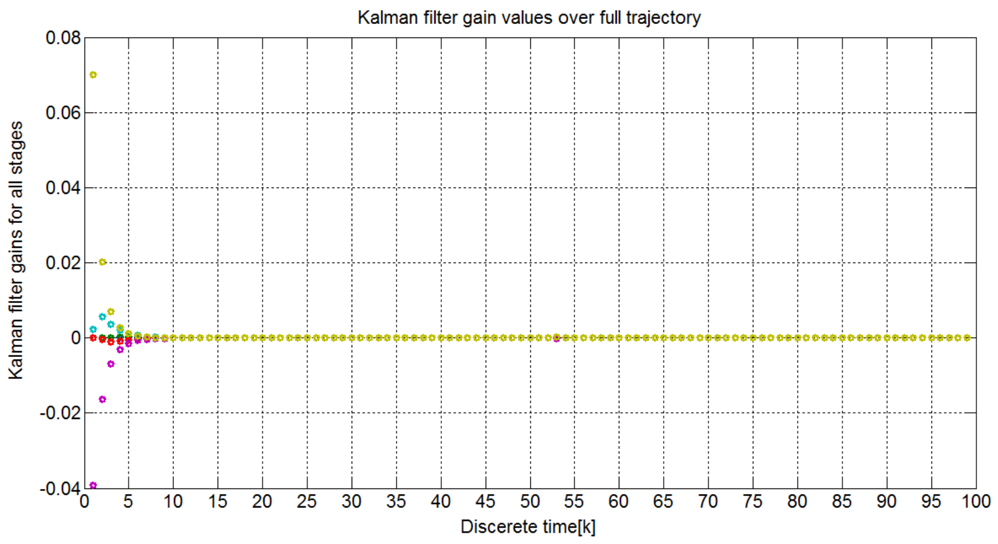


Figure 9.23. Kalman filter gain values for the best case results at $\hat{x}(k) = 10 \times \hat{x}(k)$ Run 4 of Table 9.8.

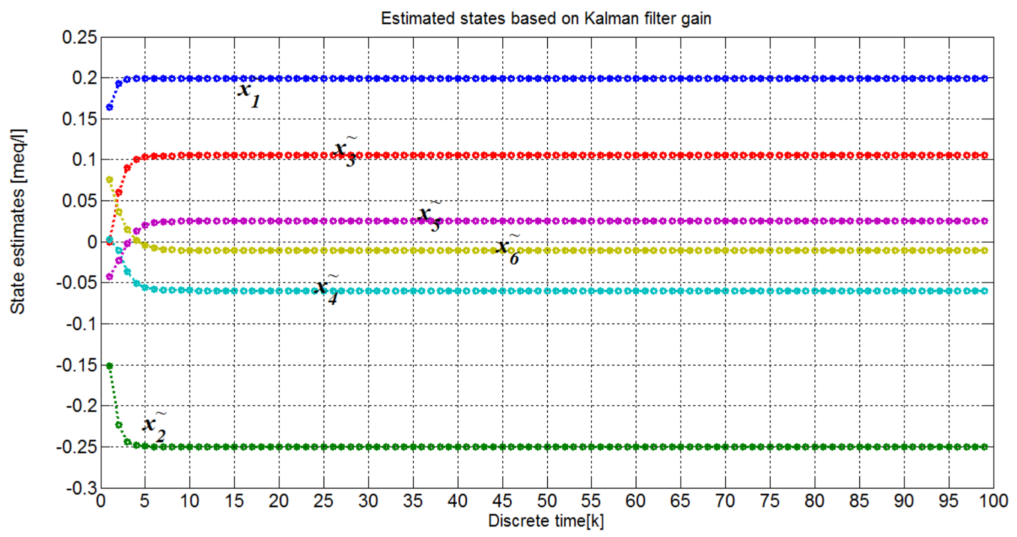


Figure 9.24. The estimated states for the CCIX process for the worst case scenario at $\hat{x}(k) = 0.0 \times \hat{x}(k)$ Run 1 of Table 9.8.

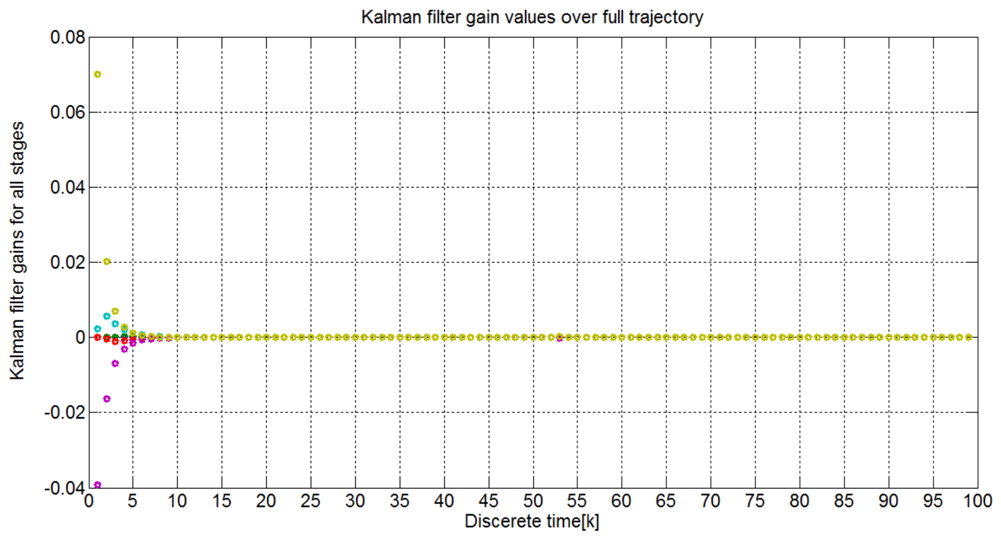


Figure 9.25. Kalman filter gain values for the worst case scenario at $\hat{x}(k) = 0.0 \times \hat{x}(k)$ Run 1 of Table 9.8.

Table 9.9: Results of the fourth set of experiments for the changing estimated state initial conditions

Estimation criterion the least squares error and the estimation run time for the fourth set of experiments		
Case 4: (Fourth set of experiments)	$J(e)$	Run time (t_{calc}) [sec]
Run 1	235.5467	0.0160
Run 2	235.7558	0.0160
Run 3	237.8325	0.0160
Run 4	243.2189	0.0150

Results and discussion of the fourth set of experiments

The best filter performance for the changing state estimate initial conditions is obtained at the initial values of $\hat{x}(k) = 10 \times \hat{x}(k)$, Table 9.9, and Figures 9.21–9.25. The processing time is lowest at the highest initial conditions $10 \times \hat{x}(k)$ and the least square error did not differ much throughout all the runs except for highest initial conditions. The state estimates also did not change in value much between the worst case and the best case scenarios.

9.10. Discussion of results from all experiments

Though the filter produced relevant results, these could be improved by investigating the developed software program further. The filter tended to converge too quickly in all experiments; this could be associated with effects of sampling on data used, Figures 9.10–9.25. This may need further investigation. All experiments produced more or less the same error condition; this indicates that the filter is performing accordingly except for data issues mentioned above.

Table 9.10: **Best results from all the filter experiments**

Method	System attribute used in the experiment	Criterion	
Experiment	Procedure of the experiment	$J(p)$	Run time (t_{calc}) [sec]
Case 1	Changing the system noise at a constant measurement noise	238.9205	0.0150
Case 2	Changing the measurement noise at constant system noise	239.9425	0.0150
Case 3	Changing measurement noise and system noise (simultaneous increase/decrease)	239.8075	0.0150
Case 4	Changing the initial conditions of the estimated states	243.2189	0.0150

9.11. Conclusion

This chapter presents two methods for design of Kalman filter used state estimation of the considered bilinear model representing a continuous countercurrent ion exchange (CCIX) process. General concepts of state estimation procedure have been discussed in the introduction of the chapter. This included the usage of statistical methods to solve estimation problems to produce optimal results. A Kalman filter for a discrete system has been considered in a general sense of linear systems to provide the basic understanding of the concept of how the Kalman filter is used to estimate an unknown state of a system. The Kalman filter design is then extended for the case of a bilinear discrete time system to estimate its unknown states using optimization techniques.

This Kalman filter for the bilinear model of the CCIX process has been derived using two methods: the first method is a general formulation of the Kalman filter design based on measurements available only up to $(k|k)$ moment, Figure 9.3; and the second method is based on direct optimization of the estimation error. The second method considers measurements that are available up to the measurement update moment $(k+1|k+1)$.

Derivation of the filter algorithms for each of the two methods has been presented. The algorithms and a flowchart for the methods are presented in detail, and also showing the methods' implementation in MATLAB software. The filter methods developed produced successful results, Figures 9.9–9.33. The experiments were conducted using MATLAB software, APPENDIX E.

MATLAB software program developed for the observer design has produced the best results in terms of estimating the process states. The MATLAB software program for the Kalman filter produced relevant results but needs some further investigation on issues of convergence.

The next chapter concludes all the work covered in the thesis: DEVELOPMENT OF METHODS FOR MODELLING, PARAMETER AND STATE ESTIMATION FOR NONLINEAR PROCESSES.

CHAPTER TEN

CONCLUSION

CHAPTER TEN
10. CONCLUSION

10.1. Introduction

Countries like South Africa have always had some water shortages one way or the other; and it seems as if the problem is getting worse by day. This has been made more complicated by high intensity of urbanization and population explosion. Recently, South Africa has experienced worse drought in many years; this means that any form of water recovery systems should be considered and be given priority, if the country is to grow economically for continued support of its population. Water is the basis of life irrespective of one's social standing. Processes like ion exchange should be considered as one of many strategies that can be implemented to prevent devastation that comes with extreme water shortages as currently experienced in the country and in the whole of sub-Saharan region.

An ion exchange process has many different formulations and thus many applications; it is a very convenient chemical process for recovery and reuse of water in its desalination capabilities. It is also used for recovery of industrial effluents, such as in:

- removal of organics and colloidal particulates,
- neutralisation of acidic mine drainage systems,
- desalination of hard brackish water,
- combined neutralisation and desalination of acidic effluents etc.

The ion exchange process can be used for a desalination function which is a huge benefit for water treatment. This means ion exchange process can be used as a treatment process of water coming out of water treatment plants, to recover domestic used water.

Work done in this thesis includes model development (reformulation), parameter estimation and state estimation methods for application in optimal control. The application is the continuous countercurrent ion exchange (CCIX) process used for desalination of water. A study for the chemical requirements and analysis of this process was done by the University of Cape Town Chemical Engineering Department in 1982. Though the focus of this earlier study was not optimal control, but on optimization of resin and regenerant chemicals based on the effluent being used (Hendry, 1982a; Hendry, 1982b). The data and results obtained from that study are equally applicable for parameter and state estimation procedures and serve as model validation tool for the current study;

and thus have been used for validating and estimating both parameters and the states of the process.

The developed model of the desalination process in the thesis is based on the reduction of sodium in the feed water but the available data was that of hydrogen content in the liquid. The model was therefore presented using both techniques but the final model is that presented by the sodium in liquid. This led to a number of proposed methods for solving the model equations; it also led to new ground in developing methods for the parameter estimation. This is as a result of introduction of nonlinear model of the process brought about by the model coefficients.

The chapter is presented as follows: 1) the deliverables of the study, process model development and reformulation, methods for parameter estimation, methods for state estimation, 2) applicability of deliverables, and 3) future developments

10.2. Deliverables

The continuous countercurrent ion exchange process model developed in Dube (2002) has been reformulated for solving the parameter and state estimation problems. The procedure for reformulation of the model with the aim that it is applied in optimal control design has been described. The developed model is then further translated into a discrete model to allow it to be used in optimal control process analysis (which includes parameter and state estimation procedures). In optimal control, if process analysis is considered with the intention of real-time implementation of the control strategy, parameter and state estimation are usually one of the requirements. This study did not consider optimal control design, but the model development, parameter and state estimation problems as applied to nonlinear processes.

The parameter methods developed can be categorized in two main types: 1) methods for estimation of parameters entering the model linearly, and 2) nonlinear methods. Under the linear methods, four solutions have been developed and are presented in the subsequent sections.

State estimation problem has also been solved based on two main techniques, the observer method and the Kalman filter. The observer method developed is based on the *pole placement technique* to solve for unknown states of the process. The Kalman filter method is a direct *Kalman technique* applied to the developed bilinear model. The solution is approached from two different possibilities; 1) the solution developed based on mean square difference of the error between estimated states and real states (referred to as the *direct solution*), and 2) the solution developed using optimization of the error difference between real states and estimated states.

10.2.1. Model development reformulation and discretization

Two models have been developed for continuous countercurrent ion exchange: 1) *a bilinear model with affine parameters* and 2) *a nonlinear model* based on the separation factor of the CCIX process. The first model was reformulated and discretized for use in parameter and state estimation. The data from the previous study (Hendry, 1982a; Hendry, 1982b) were rearranged to represent sodium (Na) ions in the liquid instead of hydrogen (H) ions as was in the previous study. The aim for reformulation was the possibility for implementation of the model in optimal control design of the process. These data, the original and the rearranged data are presented in MATLAB simulation runs performed in Chapter 5, and the results were successful.

The nonlinear model, nonlinear according to process parameters is considered under the parameter estimation problem and solved separately. The model is very complex and derivations of the solution have been presented in Chapter 7. Due to the complex nature of this model, it was not used in the state estimation problem and such a problem could be considered as a future study.

10.2.2. The parameter estimation deliverables

From the standard presentation of the continuous countercurrent ion exchange process model, developed in Chapter 4, the process model has been reformulated for parameter estimation problem (Chapter 6). Two models have been developed; 1) a bilinear model with affine parameters which allows application of linear techniques to solve the parameter estimation problem since the model is linear according to its parameters, and 2) the nonlinear model according to the parameters. The second model is firstly solved using MATLAB software and also solved using the Lagrange optimization technique.

10.2.2.1. Methods for linear parameter estimation

The parameter estimation problem for a bilinear model with affine parameters has been solved using four different methods: 1) optimization gradient-based method that uses the process output measurements, 2) optimization gradient-based method that uses the full state vector measurements, 3) direct solution using the state vector measurements, and 4) Lagrange's optimization technique based on state vector measurements. Derivation of each problem formulation of the above methods has been presented.

The results for all the four parameter estimation methods developed have been successful in variable levels, and these are presented and discussed at the end of Chapter 6. The direct method seems to have been the best of all four methods in terms of calculation speed and error evaluation criterion.

10.2.2.2. Methods for nonlinear parameter estimation

The nonlinear according to parameters model was developed based on introduction of the separation factor of the ion exchange process, Chapter 7. In this case, the model parameters enter the process equations in a nonlinear manner. Two methods for nonlinear parameter estimation were derived: 1) one method based on the direct solution of the model equations using MATLAB *software functions* and 2) a second one, developed based on the *Lagrange's function optimization*. The results from the direct method have been presented and the derivation of the solution for the Lagrangian method was developed.

10.2.3. The state estimation deliverables

The state estimation problem for the developed bilinear model of the CCIX process has been formulated and solved using two methods: 1) the observer method based on pole placement technique and the Kalman filter based method. The derivation of the solutions of these methods is presented in Chapter 8 and 9 respectively. The observer method produced very positive results. The results and discussion are presented in Chapter 8. The Kalman filter technique also produced satisfactory results. These results and the discussion are presented in Chapter 9.

10.2.3.1. Methods for design of a bilinear observer

A method for design of a bilinear observer based on *pole placement technique* has been proposed for the bilinear model of the CCIX process. The derivation of the observer matrices is based on determinants of the characteristic equation and is presented in Chapter 8. The results from MATLAB simulation are also presented in the same chapter.

10.2.3.2. Method for design of a Kalman filter

The method for design of the Kalman filter proposed is based on two approaches. The first approach is a *general method* based on minimization of least squares of the filter error. The second approach is based on *direct* *optimization* of the error between estimates and the real states. The final estimation solution simulation is presented in Chapter 9 and has shown the solution to be successful.

10.3. Developed software

Table 10.1: **A table showing developed software for all the project methods**

No.	Name of software	Functions of the software	Chapter number	Appendix reference
1.	MATLAB	MATLAB software program for H^+ fractional change for the 6 stage system of a LOW to HIGH concentration step change	Chapter 5	APPENDIX A.1
2.	MATLAB	MATLAB software program for H^+ fractional change for the 6 stage system of a HIGH to LOW concentration step change	Chapter 5	APPENDIX A.2
3.	MATLAB	MATLAB software program for the H^+ fractional change for the 12 stage system moving from a LOW to HIGH concentration step change	Chapter 5	APPENDIX A.3
4.	MATLAB	MATLAB software program for the H^+ fractional change for the 12 stage system moving from HIGH to LOW concentration step change	Chapter 5	APPENDIX A.4
5.	MATLAB	MATLAB software program for the Na^+ fractional change values of the 6 stage system for a LOW to HIGH concentration step change	Chapter 5	APPENDIX A.5
6.	MATLAB	MATLAB software program for the Na^+ fractional change values of the 6 stage system for a HIGH to LOW concentration step change	Chapter 5	APPENDIX A.6
7.	MATLAB	MATLAB software program for the Na^+ fractional change values of the 12 stage system for a LOW to HIGH concentration step change	Chapter 5	APPENDIX A.7
8.	MATLAB	MATLAB software program for the Na^+ fractional change values for the 12 stage system moving from high to low concentration step change	Chapter 5	APPENDIX A.8
9.	MATLAB	Program for parameter estimation using the measured data of the process output	Chapter 6	APPENDIX B.1
10.	MATLAB	Program for parameter estimation using the state vector measurements (the least squares method)	Chapter 6	APPENDIX B.2
11.	MATLAB	Program for parameter estimation using the state vector measurements (direct method)	Chapter 6	APPENDIX B.3
12.	MATLAB	Program for parameter estimation using the state vector measurements (Lagrange method)	Chapter 6	APPENDIX B.4

13.	MATLAB	Program for nonlinear parameter estimation – the direct method using MATLAB fsolve() function	Chapter 7	APPENDIX C.1
14.	MATLAB	Program for bilinear observer design	Chapter 8	APPENDIX D.1
15.	SIMULINK	Simulink program for the design of the bilinear observer	Chapter 8	APPENDIX D.2
16.	MATLAB	MATLAB program for extracting values of the design of the observer back to workspace	Chapter 8	APPENDIX D.3
17.	MATLAB	Program for state estimation using design of the Kalman filter	Chapter 9	APPENDIX E

10.4. Application of the thesis deliverables

The developed deliverables may be applied in the following settings:

- in classes for control systems,
- for research,
- for real-time process observation, modelling, and control, and
- relevant for design and development of new ion exchange plants.

The work covered could be used for educational purposes, for example, introduction of theory of modelling, parameter and state estimation in nonlinear processes at a graduate level. It could be used as a basis for practical implementation of different estimation methods and algorithms developed using software. The methods developed for parameter estimation and state estimation allow for further research in estimation of nonlinear processes.

The developed methods for modelling and estimation have been designed with the intent and purpose of being able to be applied in real-time observation and control of industrial processes, even if these processes constitute high nonlinearities within their components. The developed methods were designed based on the countercurrent ion exchange process, but should be applicable to any other process with the same characteristics. On the bases of the developed methods, it is further possible to advance a control system of an ion exchange process to a superior one using real-time optimal control. Such control strategies are designed to integrate commercial objectives with that of the control system.

10.5. Future developments of methods and applicability

Optimization and process automation have since become the forerunner for modernized economies. Process optimization and control, bridge the gap between the people’s needs, engineering (technology) and the economics. Online system engineering has also developed over a very short period of time with the improved Internet services, which allow easy access to information instantly irrespective of the physio-geographical location. Global Positioning Systems (GPS) have also opened online optimization to imaginable possibilities. Data storage capabilities and data transfer speeds have also increased sharply in the last few years at a much reduced cost. This makes process optimization attractive for process and or control engineers.

Highly optimized control possibilities have been made wide open by the mentioned developments, and these allow the developed solutions to be easily implemented in real–time environment. The developed solutions can be applied in the following possible future strategies:

- Application of the developed methods in a real plant for real-time control and monitoring,
- Extension of developed algorithms and programs for online estimation,
- Utilizing the proposed solutions in an optimal control problem in real–time, and
- Solving the state estimation problem of the developed nonlinear model.

10.6. Publications produced in connection with the thesis

N.M. Dube & R.T. Tzoneva, 2016. Observer design for a bilinear model of a continuous countercurrent ion exchange process, SAIEE Africa Research Journal, Vol. 107, (4), December 2016, SOUTH AFRICAN INSTITUTE OF ELECTRICAL ENGINEERS (SAIEE), pp. 193–243.

In progress: N.M. Dube & R.T. Tzoneva, 2017. Kalman filter design for a bilinear process model with affine parameters. To be submitted to SAIEE Africa Research Journal.

BIBLIOGRAPHY

BIBLIOGRAPHY

Abbaszadeh, M & Marquez, H.J. 2006. A robust observer design method for continuous-time Lipschitz nonlinear systems. *Proceedings of the* 45th *IEEE Conference on Decision* & *Control*, Manchester Grand Hyatt Hotel, San Diego, CA, USA, December 13–15, 2006, ThIP2.15, pp. 3795–3800

Abonyi, J., Babuska, R., Setnes, M., Verbruggen, H.B. & Szeifert, F. 1999. Constrained parameter estimation in fuzzy modeling. In proceedings of *IEEE*'99 *Fuzzy Systems Conference*, 2, pp. 951–956, 1999 IEEE International, 22 August 1999.

Aguilar, R. Martinez-Guerra, R. & Maya-Yescas, R. 2003. State estimation for partially unknown nonlinear systems: a class of integral high gain observers. *In IEE Proceedings on Control Theory* & *Applications*, 150, (3), 3 May 2003, pp. 240–244

Ahmed, M.S. 1994. An innovation representation for nonlinear–systems with application to parameter and state estimation. *Automatica*, 30, (12), pp. 1967-1974, Elsevier Science Ltd, 1994.

Ahmed–Ali, T., Giria, F., Krstic, M., Burlion, L. & Lamnabhi-Lagarrigue, F. 2017. Nonlinear robust observer design using an invariant manifold approach. *Automatica*, 82, August 2017, pp. 93–100.

Alaei, H.K., Salahshoor, K. & Alaei, H.K. 2010. Model predictive control of distillation column based recursive parameter estimation method using Hysys simulation. 2010 *International Conference on Intelligent Computing* & *Cognitive Informatics*, IEEE Computer Society, pp. 308–311

Alcaraz-Gonzalez, V., Salazar-Pena, R., Gonzalez-Alvarez, V., Gouze, J-L. & Steyer, J-P. 2005. A tunable multivariable nonlinear robust observer for biological systems. Biological modelling/Biomodélisation; C. R. *Biologies*, 328 (2005), pp. 317–325. http://france.elsevier.com/direct/CRASS3/

Alexandrov, A.G., Alexandrov, V.A. & Shatov, D.V. 2016. Parametric identification of nonlinear model for managed pressure drilling, *IFAC-Papers OnLine*, 49–12, (2016), pp. 1916–1921, IFAC OnLne Conference Paper Archive.

Alonso, C., Zhu, X., Suidan, M.T., Kim, B.R. & Kim, B.J. 2000. Parameter estimation in biofilter systems. *Environmental Science* & *Technology* (*American Chemical Society*), 34, 11, pp. 2318–2323.

Anderson, B.D.O. & Moore, J.B. 2005. *Optimal filtering*. Mineola, NY: Dover Publications Inc.

Anderson, B.D.O. & Moore, J.B. 1991. Kalman Filtering: Whence, What and Whither? Antoulas, A.C. (ed) *Mathematical System Theory*, pp. 41–54, Heidelberg, Berlin: Springer-Verlag, 1991.

Anderson, B.D.O. & Moore, J.B. 1979. *Optimal filtering*. INFORMATION AND SYSTEM SCIENCES SERIES Kailath T. (ed). Prentice-Hall Information and System Sciences Series. Englewood Cliffs, New Jersey: Prentice-Hall Inc.

Anderson, B.D.O. & Moore, J.B. 1971. The Kalman–Bucy filter as a true time-varying Wiener filter. *IEEE Transactions on Systems, Man, & Cybernatics*, SMC-1, (2), April, 1971, pp. 119–128. *Reprinted by permission from* IEEE Transactions On Systems, Man, & Cybernetics. Printed in the U.S.A. 1971, copyrighted by *Institute of Electrical & Electronics Engineers* Inc.

Andrieu, C., Doucet, A. & Tadic, V.B. 2005. On-line parameter estimation in general state-space models. *In Proceedings of the* 44th *Conference on Decision* & *Control*, pp. 332–337, Seville, Spain, December 2005.

Andrieu, C. & Doucet, A. 2003. On-line expectation-maximization type algorithms for parameter estimation in general state space models. In Proceedings of the *IEEE International Conference on Acoustics*, *Speech*, *& Signal Processing*, (6), 2003, pp. 69–72.

Ashayeri, L., Shafiee, M. & Menhaj, M.B. 2013. Kalman filter for fractional order singular systems. *Journal of American Science* 2013, 9, (1). http://www.jofamericanscience.org.

Atroune, D. 1988. Nonlinear observers for continuous fermentation processes, *Applied Mathematics Letters*, 1, (4), (1988), pp 321–325.

Austin, C.D., Ash, J.N. & Moses, R.L. 2013. Dynamic dictionary algorithms for model order and parameter estimation. *IEEE Transactions on Signal Processing*, 61, (20), October 15, 2013, pp. 5117–5130.

Awasthi, V & Raj, K. 2011. A comparison of Kalman filter and extended Kalman filter in state estimation. *International Journal of Electronics Engineering*, 3, (1), 2011, pp. 67–71.

Baang, D., Stoev, J., Choi, J.Y. & Park, J. 2005. Simplified adaptive nonlinear observer using B-spline based approximations. The 16th *IFAC World Congress*, Prague, July 3–8, 2005.

Badwe, A.S., Singh, A., Patwardhan, S.C., Ravindra D. & Gudi, R.D. 2010. A constrained recursive pseudo-linear regression scheme for on-line parameter estimation in adaptive control. *Journal of Process Control*, 20, (2010), pp. 559–572. www.elsevier.com/locate/jprocont.

Bai, E-W. & Liu, Y. 2006. Least squares solutions of bilinear equations. *Systems & Control Letters*, 55, (2006), pp. 466–472. www.elsevier.com/locate/sysconle, [available online at: www.sciencedirect.com].

Bai-Lan, Li. & Xiao-Lin T. 1988. On-line parameter identification for discontinuous nonlinear systems by pattern iteration recursive of partitioned data and expert system modification. Prepr. 8th *IFAC SYMPOSIUM on Identification and System Parameter Estimation*, (Beijing: CHINA) pp. 518–523.

Bak, D. Michalik, M. & Szafran, J. 2003. Application of Kalman filter technique to stationary and nonstationary state observer design. Accepted for presentation at 2003 *IEEE Bologna Power Technology Conference*, June 23–26, Bologna, Italy.

Balemi, S. 2008. Partial-order reduction of observers for linear systems. Proceedings of the 17th World Congress, The International Federation of Automatic Control (IFAC), Seoul, Korea, July 6 – 11, 2008, pp. 7723–7728.

Barbot, J-P., Boutat, D. & Floquet, T. 2009. An observation algorithm for nonlinear systems with unknown inputs. *Automatica*:Technical Communique, 45 (2009), pp. 1970–1974. www.elsevier.com/locate/automatica.

Bard, Y. 1974. *Nonlinear parameter estimation*. New York (NY): Academic Press, Inc.

Bard Y. & Lapidus L. 1970. Nonlinear system identification. *Industrial & Engineering Chemistry Fundamentals*, 9, (4), pp. 628–633.

Barham, R. H. & Drane, W. 1972. An algorithm for least squares estimation of nonlinear parameters when some of the parameters are linear. *Technometrics*, 14, (3), August, 1972.

Barret, J.F. 1963. The use of functionals in the analysis of nonlinear physical systems. *Journal of Electronics and Control*, 15, pp. 567–615.

Bar-Shalom, Y. & Xiao-Rong L. 1993. *Estimation and tracking: Principles, techniques and software*, Boston: Artech House.

Behal, A., Jain, A.K. & Joshi, K. 2006. Observers for a special class of bilinear systems: Design and application. *IEEE Transactions on Automatic Control*, 51, (11), November 2006, pp. 1854–1858.

Bengea, S., Adetola, V., Kang, K., Liba, M.J., Vrabie, D. Bitmead, R. & Narayanan, S. 2011. Parameter estimation of a building system model and impact of estimation error on closed-loop performance. 2011 50th *IEEE Conference on Decision* & *Control* and *European Control Conference* (CDC-ECC) Orlando, FL, (USA), December 12–15, 2011, pp. 5137–5142.

Bernstein, D.S. & Haddad, W.M. 1989. Optimal reduced-order state estimation for unstable plants. *International Journal of Control*, 1989, 50, (4), pp. 1259–1266.

Besancon, G. & Munteanu, I. 2015. Control strategy for state and input observer design. *Systems & Control Letters*, 85, November 2015, pp. 118–122.

Bezzaoucha, S., Marx, B., Maquin, D. & Ragot, J. 2013. State and parameter estimation for nonlinear systems: A Takagi-Sugeno approach, 2013 *American Control Conference* (ACC), Washington, DC, USA, June 17-19, 2013, pp. 1050–1055.

Bhattacharjee, U. & Das, P. 2013. Performance evaluation of Wiener filter and Kalman filter combined with spectral subtraction in speaker verification system. *International Journal of Innovative Technology* & *Exploring Engineering* (IJITEE), 2, (2), January 2013, pp. 108–112.

Biagiola, S.I., & Figueroa, J.L., 2002. State estimation in nonlinear processes. Application to pH process control. *Industrial & Engineering Chemistry Research*. Published on the Web 15 August 2002 by American Chemical Society, 41, pp. 4777–4785.

Biagiola, S.I., & Figueroa, J.L., 2002. State estimation in nonlinear processes. Application to pH process control. *Industrial & Engineering Chemistry Research*. Published on the Web 15 August 2002 by American Chemical Society, 41, pp. 4777–4785.

Biller, B. & Nelson. B.L. 2002. Fitting ARTA input processes for simulation. Working paper, *Department of Industrial Engineering and Management Sciences*, Northwestern University, Evanston, Illinois.

Billings, S.A. 1980. Identification of nonlinear systems–A survey. *Proceedings of IEE, Part D*, 127, (6), pp. 272–285.

Billings, S.A. & Fadzil M.B. 1985. The practical identification of systems with nonlinearities. IFAC/IFORS *Symposium on Identification and System Parameter Estimation*, York: UK, pp. 155–160.

Billings, S.A. & Fakhouri S.Y. 1982. Identification of systems containing linear dynamic and static nonlinear elements. *Automatica*, 18, (1), pp. 15–26.

Billings, S.A. & Fakhouri, S.Y. 1979. Identification of systems composed of linear dynamic and static nonlinear elements. Prepr. at 5th *IFAC SYMPOSIUM on Identification and Parameter Estimation*, (Darmstadt: FRG), pp. 493–500.

Billings, S.A. & Voon W.S.F. 1987. Piecewise linear identification of nonlinear systems. *International Journal of Control*, 46, (1), pp. 215–235.

Billings, S.A. & Voon W.S.F. 1984. Least squares parameter estimation algorithms for nonlinear systems. *International Journal of Systems Science*, 15, (6), pp. 601–615.

Billings, S.A. 1980. Identification of nonlinear systems–A survey. *Proc. IEE, Part D*, 127, (6), pp. 272–285

Bochenek, R., Sitarz, R. & Antos, D. 2011. Design of continuous ion exchange process for the wastewater treatment. *Chemical Engineering Science*, 66, (23), 1 December 2011, pp. 6209–6219. https://doi.org/10.1016/j.ces.2011.08.046.

Bohner, M. & Wintz, N. 2013. The Kalman filter for linear systems on time scales, *Journal of Mathematical Analysis & Applications*, 406, (2013), pp. 419–436.

Boker, A.M. & Khalil H.K. 2013. Nonlinear observers comprising high-gain observers and extended Kalman filters. *Automatica*, 49, (2013), pp. 3583–3590.

Bolognani, S., Tubiana, L. & Zigliotto, M. 2003. Extended Kalman filter tuning in sensorless PMSM drives. *IEEE Transactions on Industry Applications*, 39, (6), November/December 2003, pp. 1741–1747.

Bonilla, J., Diehl, M., Logist, F., De Moor, B., Diehl, M. & van Impe, J. 2009. A convex approximation for parameter estimation involving parameter-affine dynamic models. *Joint* 48th *IEEE Conference on Decision and Control* and 28th *Chinese Control Conference*, Shanghai, P.R. China, 16–18 December, (2009), ThB14.1, pp. 4670–4675.

Bornard, G. 1995. Physical Modeling. In Fossard, A.J. & Normand-Cyrot, D. (eds). *Nonlinear Systems: Modeling and estimation. Volume 1*. [English language translation revised by Mrs M.B. Groen-Garrer. Original French language edition: *Systèmes non linéaires*, coordonné par A.J. Fossard et D. Normand-Cyrot. (1993), France: Masson Éditeur]. London (UK): Chapman & Hall: pp. 5–32.

Bose, A.G. 1959. Nonlinear system characterization and optimization. *IRE Transactions on Circuit Theory*, 6. (Special Supplement), pp. 34–40.

Boutat, D., Zheng, G. & Hammouri. H. 2010. A nonlinear canonical form for reduced order observer design. 8th *IFAC Symposium on Nonlinear Control Systems* (NOLCOS), University of Bologna, Italy, September 1–3, 2010, IFAC Proceedings Volumes, 43, (14), September 2010, Pages 409–414.

Bozic, S M. 1979. *Digital and Kalman filtering*, London: Edward Arnold.

Braiek, E.B. & Rotella, F. 1994. State observer design for analytical nonlinear systems. *Proceedings of IEEE International Conference on Systems Man & Cybernetics*, 1994, 3, pp. 2045–2050.

Bregon, A., Biswas, G. & Pulido, B. 2012. A decomposition method for nonlinear parameter estimation in TRANSCEND. IEEE *Transactions on Systems, Man, & Cybernetics—Part A: Systems* & *Humans*, 42, (3), May 2012, pp. 751–763.

Britt, H.I. & Luecke, R.H. 1973. The estimation of parameters in nonlinear, implicit models. *Technometrics*, 15, (2), May 1973. *American Statistical Association* & *American Society for Quality*. http://www.jstor.org.

Bruni, C., DiPillo, G. & Koch, G. 1974. Bilinear systems: An appealing class of nearly linear systems in theory and application. *IEEE Transactions on Automatic Control*, AC-19, (1974), pp. 334–348.

Busawon, K. Farza, M. & Hammouri, H. 1998. A simple observer for a class of nonlinear systems. *Applied Mathematics Letters*, 11, (3), pp. 27–31, 1998, Elsevier Science Ltd.

Busawon, K.K. & Saif, M. 1999. A State observer for nonlinear systems. *IEEE Transactions on Automatic Control*, 44, (11), pp. 2098–2103, November 1999.

Califano, C., Monaco S. & Normand-Cyrot, D. 2003. On the observer design in discrete-time. *Systems & Control Letters*, 49, (2003), pp. 255–265.

Cantoni, A. 1971. Optimal curve fitting with piecewise linear functions. *IEEE Transactions on Computers*, C-20, 1, pp. 1307–1308.

Cao, Z., Lu, J., Zhanga, R. & Gao, F. 2016. Iterative learning Kalman filter for repetitive processes, *Journal of Process Control*, 46, October 2016, Pages 92–104.

Cao, L. & Schwartz, H.M. 2004. Analysis of the Kalman filter based estimation algorithm: An orthogonal decomposition approach. *Automatica*, 40 (2004), pp. 5–19. www.elsevier.com/locate/automatica.

Cao, L. & Schwartz, H.M. 2001. The Kalman filter based recursive algorithm: Windup and its avoidance. *Proceedings of the 2001 American Control Conference*, Arlington, Virginia (USA), pp. 3606–3611.

Cappelle, M.A. & Davis, T.A. 2016. Ion exchange membranes for water softening and high-recovery desalination. In *Emerging Membrane Technology for Sustainable Water Treatment*, 2016, pp. 163–179. https://doi.org/10.1016/B978-0-444-63312-5.00007-3.

Cari, E.P.T. & Alberto, L.F.C. 2011. Parameter estimation of synchronous generators from different types of disturbances. *IEEE* 2011, pp. 1–7.

Carvalho, E.P. Martinez, J. Martinez, J.M., & Pisnitchenko, F. 2015. On optimization strategies for parameter estimation in models governed by partial differential equations, *Mathematics & Computers in Simulation*, 114, (2015), pp. 14–24.

Chambers, J.M. 1973. Fitting nonlinear models: Numerical techniques. *Biometrika* (1973), 60, (1), p. 1–13.

Chatzis, M.N., Chatzi, E.N. & Triantafyllou, S.P. 2017. A discontinuous extended Kalman Filter for non-smooth dynamic problems. *Mechanical Systems & Signal Processing*, 92, August 2017, pp. 13–29.

Chen, M.-S. & Chen, C.-C. 2007. Robust nonlinear observer for Lipschitz nonlinear systems subject to disturbances. *IEEE Transactions on Automatic Control*, 52, (12), December 2007, pp. 2365–2369.

Chen, X. & Kano, H. 2002. A new state observer for perspective systems. *IEEE Transactions on Automatic Control*, 47, (4), April 2002, pp. 658–663.

Chianeh, H.A., Stigter, J.D. & Keesman, K.J. 2011. Optimal input design for parameter estimation in a single and double tank system through direct control of parametric output sensitivities. *Journal of Process Control*, 21, (2011), pp. 111–118.

Ching, J., Beck, J.L., Porter, K.A. & Shaikhutdinov, R. 2006. Bayesian state estimation method for nonlinear systems and its application to recorded seismic response. *Journal of Engineering Mechanics*. ASCE April 2006. DIO:10.1061 (ASCE) 0733-9399, (2006) 132, (4) pp. 396.

Ching, J., Beck, J.L. & Porter, K.A. 2006. Bayesian state and parameter estimation of uncertain dynamical systems. *Probabilistic Engineering Mechanics*, 21, (2006), pp. 81–96.

Ching, J., Beck; J.L., Porter, K.A. & Shaikhutdinov, R. 2006. Bayesian state estimation method for nonlinear systems and its application to recorded seismic response, *Journal of Engineering Mechanics*, © ASCE / APRIL 2006, pp. 396–410.

Chitralekha, S.B., Prakash, J., Raghavan, H., Gopaluni, R.B. & Shah, S.L. 2010. A comparison of simultaneous state and parameter estimation schemes for a continuous fermentor reactor. *Journal of Process Control*, 20 (2010), pp. 934–943. www.elsevier.com/locate/jprocont

Chon, K.H. & Cohen, R.J. 1997. Linear and nonlinear ARMA model parameter estimation using an artificial neural network. *IEEE Transactions on Biomedical Engineering*, 44, (3), March 1997, pp. 168–175.

Chong, M.S., Nesic, D., Postoyan, R. & Kuhlmann, L. 2014. State and parameter estimation of nonlinear systems: a multi-observer approach. 53rd *IEEE*

Conference on Decision & Control, December 15-17, 2014. Los Angeles, California, USA, pp. 1067–1072.

Chu, D., Chen, T. & Marquez, H.J. 2007. Robust moving horizon state observer. *International Journal of Control*, 80, (10), October 2007, pp. 1636–1650.

Ciccrella, G., Dalla-Mora, M. & Germani, A.A 1995. A robust observer for discrete time nonlinear systems, *Systems & Control Letters*, 24, (1995), pp. 291–300.

Ciccarella, G., Dolla-Mora, M.D. & Germani, A. 1993. Observers for discrete-time nonlinear systems. *Systems & Control Letters*, 20, (5), pp. 373-382, May 1993.

Cimpoesu, E.M., Ciubotaru, B.D. & Stefanoiu, D. 2013. Fault detection and diagnosis using parameter estimation with recursive least squares. *In the* 2013 19th *International Conference on Control Systems* & *Computer Science*, IEEE Computer Society, pp. 18–23.

Conticelli, F. & Bicchi, A. 2000. Observer design for locally observable analytic systems: Convergence and separation property. In Isidori, A., Lamnabhi-Lagarrigue, F. & Respondek, W. (eds). *Nonlinear control in the Year* 2000, pp 315–330, Part of the Lecture Notes in Control and Information Sciences book series (LNCIS, volume 258), Conference paper, first online: 28 September 2007.

Copp, D.A. & Hespanha, J.P. 2017. Simultaneous nonlinear model predictive control and state estimation, *Automatica*, 77, (2017), pp. 143–154.

Corlis, R.G. & Luus R. 1969. Use of residuals in the identification and control of two-input, single-output systems. *Industrial & Engineering Chemistry Fundamentals* (*I & EC Fund*), 8, (5), pp. 246–253, ©American Chemical Society Publication.

Coulson, J.M., Richardson, J.F., Rackhurst, J.R. & Harker, J.H. 1996. *Coulson and Richardson's Chemical Engineering Volume* 1. 5th ed. Oxford: Butterworth-Heinemann.

Dai, H., Sinha, N.K. & Puthenpura, S.C. 1989. Robust combined estimation of the state and parameters of bilinear systems. *Automatica*, 1989, 25, (4) pp. 613–616. UK Permagon Press PLC: 1989 International Federation of Automatic Control.

Dani, A.P., Chung, S.J. & Hutchinson, S. 2012. Observer design for stochastic nonlinear systems using contraction analysis. *In Proceeding of the IEEE Conference in Decision Control*, Maui, Hawaii, 2012.

Dean, G.C. 1986. An introduction to Kalman filter. *Measurement & Control*. 19. March 1986, pp. 69–73.

Delgado, A., Hou, M. & Kambhampati, C. 2005. Neural observer by coordinate transformation. *IEE Proceedings–Control Theory Applications*, 152, (6), November 2005, pp. 698–706. doi: 10.1049/ip-cta:20045069.

Demirbas, K. 2015. Online state estimation for discrete nonlinear dynamic systems with nonlinear noise and interference. *Journal of the Franklin Institute*, 352, (1), January 2015, pp. 216–235.

Derese, I. & Noldus, E.J. 1980. Nonlinear control of bilinear systems. In *IEE Proceedings* D–*Control Theory and Applications*, 127, (4), July 1980, pp. 169–175.

Derese, I. & Noldus, E. 1981. Existence of bilinear state observers for bilinear systems. *IEEE Transactions on Automatic Control*, 26, (2), April 1981, pp. 590–592.

Derese, I., Stevens, P. & Noldus, E. 1979. Observers for bilinear systems with bounded input. *International Journal of Systems Science*, 6, (1979), pp. 649–668. Published online: 27 Apr 2007, http://dx.doi.org/10.1080/00207727908941608.

Deza, F., Bossanne, D., Bussevelle, E., Gauthier, J.P. & Rakotopara, D. 1993. Exponential observers for nonlinear systems. *IEEE Transactions on Automatic Control*, 38, pp.482–484.

Deza, F., Busvelle, E., Gauthier, J.P. & Rakotopara, D. 1992. High gain estimation for nonlinear systems, *Systems & Control Letters*, 28, (1992), pp. 295–299.

Diaz H. & Desrochers A.A. 1988. Modeling of nonlinear systems of nonlinear discrete time systems from input-output data. *Automatica*, 24, (5), pp. 629–641.

Diekmann K. & Unbehauen H. 1985. Online parameter estimation in a class of nonlinear systems via modified least squares and instrumental variable algorithms. In 7th *IFAC SYMPOSIUM on Identification and System Parameter Estimation*, (York: UK), pp. 149–153.

Ding, F. Liu, X. & Liu, M. 2016. The recursive least squares identification algorithm for a class of Wiener nonlinear systems. *Journal of the Franklin Institute*, 353, (7), May 2016, pp. 1518–1526.

Ding, F., Wang, Y., Dai, J., Li, Q., & Chen, Q. 2017. A recursive least squares parameter estimation algorithm for output nonlinear autoregressive systems using the input–output data filtering. *Journal of the Franklin Institute*, Available online 12 August 2017, pp. 1–18. http://dx.doi.org/10.1016/j.jfranklin.2017.08.009

Diversi, R., Guidorzi, R. & Soverini, U. 2005. Kalman filtering in extended noise environments. *IEEE Transactions on Automatic Control*, 50, (9), September 2005, pp. 1396–1402.

Dochain, D. 2003. State and parameter estimation in chemical and biochemical processes: A tutorial. *Journal of Process Control*, 13, pp. 801–818.

Dochain, D. & Vanrolleghem, P.A. 2001. *Dynamical Modelling and Estimation in Wastewater Treatment Processes*. London, UK: IWA Publishing.

Dodds, R. Hudson, P.I. Kershenbaum, L. & Streat, M. 1973. The operation and modelling of a periodic countercurrent solid-liquid reactor. *Chemical Engineering Science*, 28, (6), 1973, pp. 1233–1248.

Doris, A., Juloski, A.L., Mihajlovic, N., Heemels-(Maurice), W.P.M.H., van de Wouw, N. & Nijmeijer, H. 2008. Observer designs for experimental non-smooth and discontinuous systems. *IEEE Transactions on Control Systems Technology*, 16, (6), November 2008, pp. 1323–1332.

Doyle III, F.J. & Henson, M.A. 1997. Nonlinear systems theory. In Henson, M.A. & Seborg, D.E. (eds). *Nonlinear process control*. Upper Saddle River, New Jersey: Prentice Hall, pp. 111–147.

Dreano, P., Fantini, J. & Laurent, R. 1997. Recursive identification of continuous bilinear systems. *IFAC Proceedings Volumes*, 30, (6), May 1997, pp. 497–501.

Dube, N.M. & Tzoneva, R.T. 2016. Observer design for a bilinear model of a continuous countercurrent ion exchange process. *SAIEE Africa Research Journal*, 107 (4), December 2016, pp. 192–214.

Dube, N.M. & Tzoneva, R. 2006a. Minimum startup time control of an Ion exchange process used for water desalination. Proceedings of the *14th Mediterranean Conference on Control and Automation, Università Politecnica delle Marche, Ancona*, 28–30 June 2006. CD-ROM Proceedings.

Dube, N.M. & Tzoneva, R. 2006b. MATLAB simulation program for ion exchange process for desalination of water. Proceedings of the *SACAC Conference*, UKNZ University, Durban, 6–7 July 2006.

Dube, N.M., Stokwe, B. & Tzoneva, R. 2005. Optimal control strategy for continuous countercurrent ion exchange (CCIX) process for water desalination. Paper presented at the SACC Post-Graduation Workshop, University of Pretoria (UP), 02 December 2005.

Dube, N.M. & Tzoneva, R. 2004. Parameter estimation of countercurrent ion exchange process for desalination of water. Technology Innovation, *7th Africon Conference in Africa, 2004 IEEE Africon*, *Gaborone, Botswana*. 15–17 September 2004, Vol.1, pp. 470–483.

Dube, N.M. & Tzoneva, R. 2003a. Development of a mathematical model for ion exchange process for desalination of water. *The Fourth International Conference on Control and Automation* (ICCA '03), June 2003, pp. 723–727.

Dube, N.M. & Tzoneva, R. 2003b. Automation of Ion exchange process used for desalination of water. *Proceedings of 1st African Control Conference, AFCON 2003*, *University of Cape Town, Cape Town, South Africa, 3-5 December 2003.*

Dube, N.M. 2002a. Modelling and optimal control of ion exchange process. Unpublished MTech dissertation, Peninsula Technikon, Bellville.

Dube, N.M. 2002a. Modelling and Optimal Control of Ion Exchange Process. Unpublished MTech dissertation, Peninsula Technikon, Bellville.

Dube, N.M. & Tzoneva, R. 2002b. Program for pH and conductivity measurements and determination of salt concentration in the ion exchange process for desalination of water. Proceedings of the *6th Africon Conference in Africa, 2004 IEEE Africon,* 2–4 October 2002, 1, pp. 109–114.

Dube, N.M. & Tzoneva, R. 2001. Method for optimal control of ion exchange process for desalination of water. *The Third International Conference on Control and Automation* (ICCA '01), December 2001, pp. 579–583.

Dyer, A. 2013. Ion exchange. Reference module in *Chemistry, Molecular Sciences & Chemical Engineering*, 2013. [Current as of 11 February 2017, Update of: A. Dyer, ION EXCHANGE, Encyclopedia of Separation Science, 2000, pp. 156–173]. https://doi.org/10.1016/B978-0-12-409547-2.04402-4.

Dzhaparidze, K., Kormos, J., van Der Meer, T. & Van Zuijlen, M.C.A. 1994. Parameter estimation for nearly nonstationary AR(1) processes. *Mathematical & Computer Modelling*, 19, (2), 1994, pp. 29–41, (c)Pergamon, 1994 Elsevier Science Ltd.

El-Sherief, H.E. 1984. State and parameter estimation of linear stochastic, multivariable sampled data systems. *IEEE Transactions on Systems, Man, & Cybernetics*, SMC-14, (6), November/ December 1984, pp. 911–919.

Engel, R. & Kreisselmeier, G. 2002. A continuous-time observer which converges in finite time. *IEEE Transactions on Automatic Control*, 47, (7), July 2002, pp.1202–1204.

Englezos, P. & Kalogerakis, N. 2001. *Applied Parameter Estimation for Chemical Engineers*. New York: Marcel Dekker.

Esposito, W.R. & Floudas, C.A. 2001. Rebuttal to Comments on "Global Optimization for the parameter estimation of differential-algebraic systems". *Industrial & Engineering Chemical Research.* 2001, 40*,* pp. 490–491.

Esposito, W. R. & Floudas, C. A. 2000. Global optimization for the parameter estimation of differential-algebraic systems. *Industrial & Engineering Chemistry Research*, 39, pp. 1291–1310.

Esposito, W.R. & Floudas, C.A. 1998. Global optimization in parameter estimation of nonlinear algebraic models via the error-in-variables approach. *Industrial & Engineering Chemistry Research*, 37, (5), pp. 1841–185.

Essabre, M., Soulami, J. & Elyaagoubi, E. 2013. Design of state observer for a class of non linear singular systems described by Takagi-Sugeno model. *Contemporary Engineering Sciences*, 6, (2013), (3), pp. 99–109. HIKARI Ltd, www.m-hikari.com.

Evensen, G. 2009. The ensemble Kalman filter for combined state and parameter estimation. *IEEE Control Systems Magazine*, 29, (3), (2009), pp. 83–104. DOI: 10.1109/MCS.2009.932223.

Eykhoff, P. 1974. *System identification*. *Parameter and state estimation*. Chichester: John Wiley & Sons.

Eykhoff, P. 1968. Process parameter and state estimation. *Automatica*, 4, pp. 205–233, Great Britain: Pergamon Press, 1968.

Faber, R., Arellano-Garcia, H., Li, P. & Wozny, G. 2007. An optimization framework for parameter estimation of large-scale systems. *Chemical Engineering* & *Processing*, 46, pp. 1085–1095.

Fang, H., de Callafon, R.A. & Cortes, J. 2013. Simultaneous input and state estimation for nonlinear systems with applications to flow field estimation. *Automatica*, 49, (2013) pp. 2805–2812.

Faragher, R. 2012. Understanding the basis of the Kalman filter via a simple and intuitive derivation. *Lecture Notes in IEEE Signal Processing Magazine*, *September* 2012, pp. 128–132. DOI: 10.1109/MSP.2012.2203621.

Farza, M. M'Saad, M. & Rossignol, L. 2004. Observer design for a class of MIMO nonlinear systems. *Automaica*, 4, pp. 135–143, 2004.

Farza, M., Hammouri, H., Jallut, C., Chouri, V & Liéto, J. 1998. Nonlinear estimation strategies for parameter estimation in chemical reactors. *Compurers & Chemical Engineering*, 22, *Suppliment*, 1998. Elsevier Science Ltd., pp. S687-S690.

Fattah, S.A., Zhu, W.-P. & Ahmad, M.O. 2008. An Algorithm for ARMA model parameter estimation from noisy observations, 2008 *IEEE International Symposium on Circuits and Systems*, (ISCAS), Seattle, WA, USA, 21–28 May, 2008, pp. 3202–3205.

Ferrero, A. 2006. Kalman filtering. CiteSeerx. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.306.5896. [Date Accessed: 20 February 2014].

Fitzgerald, R.J. 1971. Divergence of the Kalman filter. *IEEE Transactions on Automatic Control*, AC-16, (6), December 1971, pp. 736–747.

Fliess M. & Normand-Cyrot D. 1982. On the Approximation of nonlinear systems by some simple state space models. Prepr. 6th *IFAC Symposium on Identification and System Parameter Estimation*, (Washington D.C.: USA), pp. 511–514.

Fogler, H.S. 1991. Elements of chemical engineering. 2nd ed. Englewood Cliffs (NJ): Prentice Hall.

Foust, A.S., Wenzel, L.A., Clump, C.W., Maus, L. & Andersen, L.B. 1980. Principles of unit operations. 2nd ed. New York: John Wiley & Sons.

Franks, R. 1972. *Modeling and Simulation in Chemical Engineering*. New York: Wiley.

Frick, P.A. & Valavi, A.S. 1978. Estimation and identification of bilinear systems, 6, pp. 1–7.

Fronza G., Guardabassi G., Locatelli A. & Rinaldi S. 1969. A simple nonlinear model for binary distillation columns. Report, *Laboratorio Controlli Automatici, Instituto di Elettrotecnica ed Elettronica*, Polytecnico Di Milano.

Fujimoto, K. & Takaki, Y. 2016. On system identification for ARMAX models based on the variational Bayesian method, 2016 *IEEE* 55th *Conference on Decision and Control* (*CDC*), ARIA Resort & Casino, December 12–14, 2016, Las Vegas, USA, pp. 1217–1222.

Fujii, Y. & Kuroda, Y. 2011. Online motion model parameter estimation using augmented Kalman filter and discriminative training. *In Proceedings of the* 2011 *IEEE International Conference on Robotics and Biomimetics*, December 7-11, 2011, Phuket, Thailand, pp. 1035–1040.

Funahashi, Y. 1979. An observable canonical form of discrete-time bilinear systems. *IEEE Transactions on Automatic Control*, 24, (5), pp. 802–803.

Gabr, M.M. & Rao, T.S. 1984. On the identification of bilinear systems from operating records. *International Journal of Control*, 40, (1), pp. 121–128.

Gabrea, M. 2003. Kalman filter-based single microphone noise canceller. International *Workshop on Acoustic Echo and Noise Control*, (IWAENC2003), September 2003, Kyoto, Japan, pp. 111–114.

Ganguly, S., Sairam, V., & Saraf, D.N. 1993. Nonlinear parameter estimation for real-time analytical distillation models. *Industrial & Engineering Chemical Research*, 32, pp. 99–107.

Gardiner, P.G. 1973. Identification of processes containing single-valued nonlinearities. *International Journal of Control*, 18, (5), pp. 1029–1039.

Garimella, P. & Yao, B. 2003. Nonlinear adaptive robust observer design for a class of nonlinear systems. *Proceedings of the American Control Conference*, Denver, Colorado June 4–6, 2003, pp. 4391–4396.

Gau, C-Y. & Stadtherr, M.A. 2000. Reliable nonlinear parameter estimation using interval analysis: Error-in-variable approach. *Computers & Chemical Engineering*, 24 (2000), pp. 631–637. www.elsevier.com/locate/compchemeng.

Gauthier, J.P., Hammouri, H. & Othman S. 1992. A simple observer for nonlinear systems: Application to bioreactors. *IEEE Transactions on Automatic Control*, 37 (1992), pp. 875–880.

Gelb, A. 1974. *Applied optimal estimation*. Cambridge, Massachusetts: The M.I.T. Press

George, D.A. 1980. Continuous nonlinear systems. Technical Report 355, MIT Research Laboratory of Electronics, Boston; MA, (USA).

Germani, A., Manes, C. & Pepe, P. 2002. A new approach to state observation of nonlinear systems with delayed output. *IEEE Transactions on Automatic Control*, 47, pp. 96–101.

Germani, A., Manes, C., & Palumbo, P. 2005. Polynomial extended Kalman filter. *IEEE Transactions on Automatic Control*, 50, (12), December 2005, pp. 2059–2064. IEEE Control Systems Society (IETAA9).

Gilles, G. & Laggoune, N. 1985. Digital control of bilinear continuous processes. Prepr. 7th *IFAC Conference on Digital Computer Applications to Process Control*, (Vienna: AUSTRIA) pp. 119–124.

Gillijns, S., Mendoza, O. B., Chandrasekar, J., De Moor, B.L.R.D., Bernstein, S. & Ridley, A. 2006. What is the ensemble Kalman filter and how well does it work? *In the Proceedings of the* 2006 *American Control Conference*, *Minneapolis, Minnesota*, USA, June 14-16, 2006, Fr08.1, pp. 4448–4453.

Goldberg, S. & Durling, A.A. 1971. Computer algorithm for the identification of nonlinear systems. *Journal of The Franklin Institute*, 291, (6), pp. 427–447.

Goncharov, O.I. 2012. Observer design for bilinear systems of a special form. *Differential Equations*, 2012, 48, (12), pp. 1596–1606, Pleiades Publishing, Ltd., 2012.

Grafa, K.C., Cornwell, D.A. & Boyer, T.H. 2014. Removal of dissolved organic carbon from surface water by anion exchange and adsorption: Bench-scale testing to simulate a two-stage countercurrent process. *Separation & Purification Technology*, 122, 10 February 2014, pp. 523–532. https://doi.org/10.1016/j.seppur.2013.12.012.

Grancharova, A. & Johansen, T.A. 2009. Approaches to explicit nonlinear model predictive control with reduced partition complexity, *Proceedings of the European Control Conference 2009*, Budapest, Hungary, August 23–26, 2009, Paper TuB3.4, pp. 2414–2419.

Grasselli, O.M. & Isidori, A. 1981. An existence theorem for observers of bilinear systems. *IEEE Transactions on Automatic Control*, 1981, 27, (6), pp. 1299–1300.

Grewal, M.S. & Angus P.A. 1993. *Kalman filtering theory and practice*. Upper Saddle River, NJ (USA): Prentice Hall.

Guay, M. 2002. Observer linearization by output-dependent time-scale transformations. *IEEE Transactions on Automatic Control*, 47, (10), October 2002, pp. 1730–1735.

Haber, R. 1979. Identification of nonlinear dynamic systems having signal dependent parameters. Report*, Center for Control Sciences*, Minneapolis (USA): University of Minnesota.

Haber, R. & Kevikczky, L. 1974. Nonlinear structures for system identification. Periodica Polytechnica, *Electrical Engineering*, 18, (4), pp. 393–414

Haber, R. & Keviczky, L. 1999. *Nonlinear system identification–Input-output modeling approach. Volume 1: Nonlinear System Parameter Identification*. [MATHEMATICAL MODELLING: Theory and Applications. Volume 7/1, editor: Lowen, R. (Antwerp, Belgium)], The Netherlands: Kluwer Academic Publisher.

Haber, R. & Kevikczky, L. 1979. Parametric description of dynamic systems. Survey Paper. Prepr. *Joint American Control Conference*. (Denver: USA) pp. 681–686.

Haber, R. & Kevikczky, L. 1976. Identification of nonlinear dynamic systems identification. – Survey Paper. Prepr. 4th *IFAC SYMPOSIUM on Identification and System Parameter Estimation*. (Tbilisi: USSR) pp. 62–112.

Halim, S., Bisono, I.N., Sunyoto, D. & Gendo, I. 2009. Parameter estimation of space-time model using genetic algorithm. *Proceedings of the* 2009 *IEEE IEEM*, pp. 1371–1375.

Hammami, M.A. & Jerbi, H. 2001. Separation principle for nonlinear systems using a bilinear approximation. *Kybernetika*, 37, (2001), (5), pp. 565–573.

Hanson, B. 2000. IRT Parameter estimation using the EM algorithm. Available online: www.b-a-h.com/papers/note9801.com. October 6, 1998 (revised 9/27/2000) [Accessed 03 October 2013]

Hara, S. & Furuta, K. 1976. Minimal order state observers for bilinear systems. *International Journal of Control*, 1976, 24, (5), pp. 705–718.

Haseltine, E.L. & Rawlings, J.B. 2003. *A critical evaluation of extended Kalman filtering and Moving Horizon estimation*. Technical Report Number 2002–03. Texas-Wisconsin Modeling and Control Consortium (TWMCC). Department of Chemical Engineering, University of Wisconsin-Madison: Madison, WI, 2002.

Haseltine, E.L. & Rawlings, J.B. 2005. Critical evaluation of extended Kalman filtering and moving-horizon estimation. *American Chemical Society*, published on Web 06/29/2004, *Industrial & Engineering Chemistry Research* 2005**,** 44, pp. 2451–2460.

Hassan, M.F., Mahmoud, M.S., Singh, M.G. & Spathopolous, M.P. 1982. A two-level parameter estimation algorithm using the multiple projection approach. *Automatica*, 18, (5), pp. 621–630. 1982 International Federation of Automatic Control.

He, K., Yuan, Z. & Mu, C. 2011. A new affine transformation parameters estimation method. 2011 *Seventh International Conference on Natural Computation*, pp. 28–32.

Heaton, C.A. (ed.). 1984. *An introduction to industrial chemistry*. London: Leonard Hill.

Hendry, B.A. 1982a. Continuous countercurrent ion exchange for desalination and tertiary treatment of effluents and other brackish waters. *Water Science Technology*, (1982), 14, (6–7), pp. 535–352.

Hendry, B.A. 1982b. Ion exchange for the desalination and tertiary treatment of effluents. [Volume 1. Theory and Design]. Cape Town: University of Cape Town, December.

Hendry, B.A. 1982c. Ion exchange for the desalination and tertiary treatment of effluents. [Volume 2. Water reclamation by CCIX using nitric acid and ammonia as regenerants]. Cape Town: University of Cape Town, December.

Hendry, B.A. 1982d. Ion exchange for the desalination and tertiary treatment of effluents. [Volume 3. Water reclamation by CCIX using sulphuric acid and lime as regenerants]. Cape Town: University of Cape Town, December.

Hendry, B.A. 1982e. Ion exchange for the desalination and tertiary treatment of effluents. [Volume 4. Control and Instrumentation]. Cape Town: University of Cape Town, December.

Hendry, B.A. 1982f. Ion exchange for the desalination and tertiary treatment of effluents. [Volume 5. Applications of the CCIX process], University of Cape Town Department of Chemical Engineering. Cape Town: University of Cape Town, December.

Horn, F.J.M. 1967. Periodic countercurrent processes. *Industrial & Engineering Chemistry Process Design and Development* (*Ind. Eng. Chem. Process Des. Dev*), 6, (1), 1967, pp. 30–35.

Horn, F.J.M. & Lin, R.C. 1967. Periodic processes: a variational approach. *I & EC Process Design and Development*, 6, (1), January, pp.21–30.

Hou M. & Pugh, A.C. 1997. Observing state in bilinear systems: A UIO approach. Copyright *IFAC Fault Detection, Supervision and Safety for Technical Processes*, Kingston Upon Hull, UK, 1997, pp. 783–788. Available at: http://ac.els-cdn.com/, [Accessed: 17 January 2017].

Hou, M., Zitek, P. & Patton, R.J. 2002. An observer design for linear time-delay systems. *IEEE Transactions on Automatic Control*, 47, (1), January 2002, pp. 121–125.

Hritonenko, N. & Yatsenko, Y. 2003. *Applied Mathematical Modelling of Engineering Problems*. [Applied Optimization, Volume 81, edited by Pardalos, P.M. and Hearn, D.W.], Massachusetts, USA: Kluwer Academic Publishers.

Hsieh, C-S. 2003. General two-stage extended Kalman filters. *IEEE Transactions on Automatic Control*, 48, (2), February 2003, pp. 289–293.

Hsu, C.S. & Desai, U.B. 1984. Approximate reduced-order observers of bilinear systems. *Proceedings of 23rd Conference on Decision & Control*, Las Vegas, NV, pp. 798–799, December 1984.

Hu, X.1991. On state observers for nonlinear systems. *Systems & Control Letters*, 17, 1991, pp. 465–473.

Hu, Z., Qin, M., Wang, J. & Liu, Y. 2013. Unscented Kalman filter based on the decoupling of correlated noise. *Journal of Computational Information System*s, 9, 7, (2013), pp. 2941–2948. Available at http://www.Jofcis.com.

Humpherys, J., Redd, P. & West, J. 2012. A fresh look at the Kalman filter. *SIAM Review*, 54, (4), pp. 001–023, 2012 Society for Industrial and Applied Mathematics.

Ibrir, S. 2015. Simultaneous state and parameter estimation of a class of mimo nonlinear systems, Proceedings of the 8th *IEEE GCC Conference and Exhibition*, Muscat, Oman, 1-4 February, 2015.

IDA. 1997. Measurement, modeling, identification and optimized controller parameter settings of MSF desalination units. *World Congress of Desalination and Water Reuse*, pp. 52–81.

Ikonen, E. & Najim, K. 2002. A*dvance process identification and control*. N. Munro, (ed.) [Control Engineering. A series of reference books and Textbooks]. New York, NY: Marcel Dekker.

Iqbal, M., Bhatti, A.I., Ayubi, S.I. & Khan, Q. 2011. Robust parameter estimation of nonlinear systems using sliding-mode differentiator observer. *IEEE Transactions on Industrial Electronics*, 58, (2), pp. 680–689, February 2011.

Iqbal, M., Bhatti, A.I., Ayubi, S.I. & Khan, Q. 2011. Robust parameter estimation of nonlinear systems using sliding-mode differentiator observer. *IEEE Transactions on Industrial Electronics*, 58, (2), pp. 680–689, February 2011.

Iqbal, M., Bhatti, A. I., Iqbal, S., Khan, Q. & Kazmi, I. H. 2009. Parameter estimation of uncertain nonlinear MIMO three tank systems using higher order sliding modes. In the *Proceedings of* 7th *ICCA*, Christchurch, New Zealand, December 9–11, 2009, pp. 1931–1936.

Isidori, A. 1995. *Nonlinear control systems*. London: Springer.

Jaakkola, T.S. 2006. Course materials for 6.867 Machine Learning, Fall 2006. MIT OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology. Downloaded on [14 January 2016].

Jacob, C. 2010. Conditional least squares estimation in nonstationary nonlinear stochastic regression models. *The Annals of Statistics* 2010, 38, (1), pp. 566–597 DOI: 10.1214/09-AOS733, Institute of Mathematical Statistics, 2010.

Jamel, W., Khedher, A., Bouguila, N. & Othman, K.B. 2013. Observer design for simultaneous state and faults estimation. *International Journal on Computer Science & Engineering* (IJCSE), 5, (9), September 2013, pp. 830–846.

Jamel, W., Bouguila, N., Khedher, A. & Othman, K.B. 2010. Observer design for nonlinear systems represented by Takagi-Sugeno models. WSEAS *Transactions on Systems*, 9, (7), July 2010.

James, M.R. & Baras, J.S. 2006. An observer design for nonlinear control systems, *In Analysis and Optimization of Systems*, (Eds) A. Bensoussan and JL. Lions, Series: Lecture Notes in Control and Information Sciences, Vol.111, Heidelberg: Springer-Verlag, pp. 170–180.

James, M.R. & Petersen, I.R. 1998. Nonlinear state estimation for uncertain systems with an integral constraint. *IEEE Transactions on Signal Processing*, November 1998, 46, (11), pp. 2926–2937.

Jamshidi, M., Tarokh, M. & Shafai, B. 1992. *Computer aided analysis and design of linear control systems*. New Jersey: Prentice Hall.

Jaulin, L. 2002. Nonlinear bounded-error state estimation of continuous-time systems. *Automatica*: Technical Communique, 38, (2002), pp. 1079–1082. www.elsevier.com/locate/automatica.

Jazwinski, A. H. 1970. *Stochastic processes and filtering theory*. New York: Academic Press.

Jeong, C.S., Yaz, E.E. & Yaz, Y.I. 2011. Resilient observer design for discrete-time nonlinear systems with general criteria. 2011 *IEEE International Conference* *on Control Applications* (ICCA) – Part of 2011 *IEEE Multi-Conference on Systems & Control* Denver, CO, USA. September 28-30, 2011, pp. 1157–1162.

Jeppsson, U. 1996. *Modelling aspects of wastewater treatment processes*. Lund: Lund University.

Jiang, B. & Chowdhury, F.N. 2005. Parameter fault detection and estimation of a class of nonlinear systems using observers. *Journal of Franklin Institute*, 342 (2005), pp. 725–736.

Johansen, T.A. & Foss, B.A. 1995. Semi-empirical modeling of nonlinear dynamic systems through identification of operating regimes and local models. *Modeling, Control And Control*, 16, (4), pp. 213–232.

Joshi, K., Behal, A. Jain, A.K. & Sadagopan, R. 2005. Observers for a special class of bilinear systems: Design, analysis, and application. 2005 *American Control Conference*, Portland, OR, USA, FrC08.3, pp. 4808–4813, June 8–10, 2005.

Jouili, M., Jarray, K., Koubaa, Y. & Boussak, M. 2011. A Luenberger state observer for simultaneous estimation of speed and rotor resistance in sensorless indirect stator flux orientation control of induction motor drive. *IJCSI International Journal of Computer Science Issues*, 8, Issue 6, (3), November 2011, pp. 116–125.

Jover, J.M. & Kailath, T. 1986. A parallel architecture for Kalman filter measurement update and parameter estimation. *Automatica*, 22, (1), pp. 43-57, Great Britain: Pergamon Press Ltd, 1986 International Federation of Automatic Control.

Julier, S.J. & Uhlmann, J.K. 2004. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92, (3), March 2004, pp. 401–422.

Julier, S.J., Uhlmann, J.K. & Durrant-Whyte, H.F. 2000. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45, (3), pp. 477–482.

Julier, S.J. Uhlmann, J. K. & Durrant-Whyte, H.F. 1995. A new approach for filtering nonlinear systems. In the Proceedings of the 1995 *American Control Conference*, Seattle, Washington, pp. 1628–1632.

Juloski, A.L., Heemels, W.P.M.H. & Weiland, S. 2007. Observer design for a class of piecewise linear systems. *International Journal of Robust Nonlinear Control*. Wiley InterScience, DOI: 10.1002/rnc.1171. www.interscience.wiley.com.

Junkins, J.L. 1991. *An Introduction of optimal estimation of dynamic systems*. Alphen Aan Den Rijn: Sijhoff and Noordhoff International.

Kalman, R.E. 1960. A new approach to linear filtering and prediction problems. *Transactions on. ASME. Journal of Basic Engineering*, 82, (1960), pp. 35-45.

Kalman, R.E. & Bucy, R.S. 1961. New results in linear filtering and prediction problems. *Journal of Basic Engineering.*, 83D, pp.95–108. *Journal of Engineering Mechanics* © ASCE/April 2006/409.

Kamoun, S. 2007. Design of parametric and state estimation algorithms for stochastic systems, *International Journal of Sciences and Techniques of Automatic Control & Computer Engineering*, IJ-STA, 1, (2), December 2007, pp. 120–135.

Kao, A.S., Stenger, H.G., Georgakis, C. & Covert, K.L. 1992. State estimation and control of spray etching processes, J. Proc. Cont. 1992, 2, (2), pp. 87–101.

Kar, S., Moura, J.M.F. & Ramanan, K. 2012. Distributed parameter estimation in sensor networks: nonlinear observation models and imperfect communication. *IEEE Transactions* on *Information Theory*, 58, (6), June 2012, pp. 3575–3604.

Karafyllis, I. & Kravis, C. 2012. Global exponential observers for two classes of nonlinear systems, *Systems & Control Letters*, 61, (2012), pp. 797–806.

Kazantzis, N., Huynh, N. & Wright, R.A. 2005. Nonlinear observer design for slow states of a singularly perturbed system. *Computers* & *Chemical Engineering*, 29 (2005), pp. 797–806. www.elsevier.com/locate/compchemeng.

Kazantzis, N. & Kravaris, C. 1998. Nonlinear observer design Lyapunov's auxiliary theorem, *Systems & Control Letters*, 34, (1998), pp. 241–247.

Kazemi, M. & Arefi, M.M. 2017. A fast iterative recursive least squares algorithm for Wiener model identification of highly nonlinear systems. *ISA Transactions*, (67), March 2017, pp. 382–388.

Keating, J.P., Mason, R.L. & Sen, P.K. 1993. Pitman's *measure of closeness*. *a comparison of statistical estimators*. Philadelphia, USA: SIAM (Society for Industrial and Applied Mathematics).

Keesman, K.J. 2000. State and parameter estimation in biotechnical batch reactors. *IFAC Systems Identification*. Santa Barbara, California.

Keesman, K.J. & van den Brink, P. 2015. On-line parameter and state estimation in membrane bioreactor systems, *IFAC-Papers on Line*, 48–28, (2015), pp. 550–555.

Keller, H. 1987. Nonlinear observer design by transformation into a generalized observer canonical form. *International Journal of Control*, 1987, 46, (6), pp. 1915–1930.

Kenne, G., Ahmed-Ali, T., Lamnabhi-Lagarriguec, F. & Arzande, A. 2008. Nonlinear systems time-varying parameter estimation: Application to induction motors. *Electric Power Systems Research*, 78, (2008), pp. 1881–1888. www.elsevier.com/locate/epsr.

Khalil, M. Sarkar, A. Adhikari, S. & Poirel, D. 2015. The estimation of time-invariant parameters of noisy nonlinear oscillatory systems. *Journal of Sound & Vibration*, 344, (2015), pp. 81–100.

Khan, I.U., Wagg, D. & Sims, N.D. 2016. Nonlinear robust observer design using an invariant manifold approach, *Control Engineering Practice*, 55, October 2016, pp. 69–79.

Khodadadi, H. & Jazayeri-Rad, H. 2011. Applying a dual extended Kalman filter for the nonlinear state and parameter estimations of a continuous stirred tank reactor. *Computers & Chemical Engineering*, 35, (2011), pp. 2426–2436. www.elsevier.com/locate/compchemeng.

Kieffer, M., Walter, E & Simeonov, I. 2003. Guaranteed nonlinear parameter estimation for continuous-time dynamic models. Proceedings of the 14th *IFAC Symposium on System Identification*, Newcastle, Australia, 18, pp. 843–848.

Kim, K-H. 2013. Sequential parameter estimation scheme for a PWM inverter-fed IPMSM control. *In the Proceedings of the* 5th *International Conference on Advanced Science* & *Technology*, *ASTL*, (AST 2013), 20, pp. 57–60, © SERSC 2013.

Kirk, E.D. 2004. *Optimal control theory*. *An introduction*. [Dover edition, unabridged republication of thirteen printing of original work published by Prentice Hall Inc., Englewood Cliffs, New Jersey 1970]. New York: Dover Publications.

Kok, M., Dahlin, J., Schon, T.B. & Wills, A. 2015. Newton-based maximum likelihood estimation in nonlinear state space models, *IFAC-Papers on Line*, 48–28, (2015), pp. 398–403.

Kolas, S., Foss, B.A. & Schei, T.S. 2009. Constrained nonlinear state estimation based on the UKF approach. *Computers & Chemical Engineering*, 33, (2009), pp. 1386–1401.

Komatsu, K. & Takata, H. 2010. Design of nonlinear observer by using chebyshev interpolation based on formal linearization. *World Academy of Science, Engineering and Technology*, 46, pp. 885–890, 2010.

Komatsu, K. & Takata, H. 2009. Design of nonlinear observer by using augmented linear system based on formal linearization of polynomial type. *World Academy of Science, Engineering and Technology*, 35, pp. 241–244.

Korenberg, M.J. 1973. A New statistical method of nonlinear system identification. Prepr. 16th *Midwest Symposium on Circuit Theory*, XVIII., pp. 1.1–1.10.

Kortmann, M. & Unbehauen, H. 1988. Structure detection in the identification of nonlinear systems. *Automatique Productique Informatique Industrielle*, 22, pp. 5–25.

Kravaris, C., Hahn, J. & Chu, Y. 2013. Advances and selected recent developments in state and parameter estimation. *Computers & Chemical Engineering*, 51, (2013), pp. 111–123.

Kravaris, C., Sotiropoulos, V., Georgiou, C., Kazantzis, N., Xiao, M-Q. & Krener, A.J. 2004. Nonlinear observer design for state and disturbance estimation. *In Proceeding of the* 2004 *American Control Conference*, Boston, Massachusetts June 30–July 2, 2004, ThM10.5, pp. 2931–2936.

Kreisselmeier, G. & Engel, R. 2003. Nonlinear observers for autonomous Lipschitz continuous systems. *IEEE Transactions on Automatic Control*, 48, pp. 451–464.

Kreisselmeier, G. & Engel, R. 2003. Nonlinear observers for autonomous Lipschitz continuous systems. *IEEE Transactions on Automatic Control*, 48, pp. 451–464.

Krener, A.J. & Isidori, A. 1983. Linearization by output injection and nonlinear observers. *Systems and Control Letters* 3, pp. 47–52.

Krener, A.J. & Respondek, W. 1985. Nonlinear observer with linearizable error dynamics, *SIAM Journal of Control & Optimization*, 23, (2), 1985, pp. 197–216.

Krener, A.J. & Qiao, M-X. 2002. Observers for linearly unobservable nonlinear systems. *Systems & Control Letters*, 46, (2002), pp. 281–288.

Krokavec, D. & Filasova, A. 2007. Pole assignment in robust state observer design. *AT&P Journal plus2. Robust and Adaptive Control*, pp. 75–78.

Kulikov, G.Y. & Kulikova. M.V. 2017. Accurate continuous–discrete unscented Kalman filtering for estimation of nonlinear continuous-time stochastic models in radar tracking. *Signal Processing*, 139, October 2017, pp. 25–35.

Kulikov, G.Y. & Kulikova. M.V. 2017. Accurate continuous–discrete unscented Kalman filtering for estimation of nonlinear continuous-time stochastic models in radar tracking. *Signal Processing*, 139, October 2017, pp. 25–35.

Kumar, V., Jerome, E.J. & Ayyappan, S. 2013. Comparison of four state observer design algorithms for MIMO system. *Archives of Control Sciences* (ACS), 23, (LIX), 2013, (2), pp. 131–144.

Kunisch, K. 1988. A review of some recent results on the output least squares formulation of parameter estimation problems. *Automatica,* 24, (4), pp. 531–539. *International Federation of Automatic Control*, 1988.

Kushner, H.J. 1967. Approximations to optimal nonlinear filters. *IEEE Transactions on Automatic Control*, AC-12, October 1967, pp. 546–556.

Ladeveze, P. 2016. On reduced models in nonlinear solid mechanics, *European Journal of Mechanics A/Solids*, 60, (2016), pp. 227–237.

Lagrange, S., Jaulin, L., Vigneron, V. & Jutten, 2008. Nonlinear blind parameter estimation. *IEEE Transactions on Automatic Control*, 53, (3), pp 834–838.

Lalonde, R.J., Hartley, T.T. & De Abreu-Garcia, J.A. 1992. Least squares model reduction, *Journal of the Franklin Institute*, 329, (2), March 1992, pp. 215–240.

Lee, W. & Nam, K. 1991. Observer design for autonomous discrete-time nonlinear systems. *Systems & Control Letters*, 17, (1991), pp. 49–58.

Lee, S. & Park, M. 2003. State observer for MIMO nonlinear systems. *IEE Proceedings on Control Theory* & *Applications*, 150, (4), July 2003, pp. 421–426.

Lee, Y.L., Sung, S.W., Park, S. & Park, S. 2004. Input test signal design and parameter estimation method for Hammerstein-Wiener processes. *Industrial & Engineering Chemistry Research*, 43, pp. 7521–7530.

Lefebvre T., Bruyninckx, H. & De Schutter J. 2004. Kalman filters for non-linear systems: a comparison of performance. *International Journal of Control*, 77, (7), April 2004, pp. 639–653.

Leontaritis, I.J. & Billings, S.A. 1985. Input-output parametric models for nonlinear systems. Part I: Deterministic Nonlinear Systems. *International Journal of Control*, 41, (2), pp. 303–328.

Lewis, F.L. 1986. *Optimal estimation with an introduction to stochastic control theory*. New York: John Wiley & Sons Inc.

Li, J. & Ding, R. 2013. Parameter estimation methods for nonlinear systems. *Applied Mathematics & Computation*, 219, (2013), pp. 4278–4287.

Li, M., Liu, X. & Ding, F. 2017. The maximum likelihood least squares based iterative estimation algorithm for bilinear systems with autoregressive moving average noise, *Journal of the Franklin Institute*, 354, (12), August 2017, pp. 4861–4881.

Li, H., Mu, B. & Zuo, L. 2016. Recursive minimum component algorithms for parameter estimation of dynamic systems. *Neurocomputing*, 186, 19 April 2016, pp.251–257.

Liang, Y., An, D.X., Zhou, D.,H., & Pan, Q. 2004. A finite-horizon adaptive Kalman filter for linear systems with unknown disturbances. *Signal Processing*, 84, (2004), pp. 2175–2194. www.elsevier.com/locate/sigpro.

Liao, Y., Wang, D. & Ding, F. 2009. Data filtering based recursive least squares parameter estimation for ARMAX models. 2009 *International Conference on Communications and Mobile Computing*, IEEE Computing Society, pp. 331–335.

Lien, C. 2004. Robust observer-based control of systems with state perturbations via LMI approach. *IEEE Transactions on Automatic Control*, 49, pp. 1365–1370.

Lin, H., Zhai, G. & Antsaklis, P.J. 2003. Set-valued observer design for a class of uncertain linear systems with persistent disturbance and measurement noise. *International Journal of Control*, (2003), 76, (16), pp. 1644–1653.

Lin, W. & Wei, J. 2009. Observer design for a class of discrete-time nonlinear systems. *European Journal of Control*, (2009), 2, pp. 184–193, 2009 EUCA.

Linga, P., Al-Saifi, N. & Englezos, P. 2006. Comparison of the Luus-Jaakola optimization and Gauss-Newton methods for parameter estimation in ordinary differential equation models. *Industrial & Engineering Chemical Research* 2006, 45, pp. 4716–4725.

Liu, J. 2013. Robust moving horizon state estimation for nonlinear systems. 2013 *American Control Conference* (*ACC*), Washington, DC, USA, June 17–19, 2013, pp. 253–258.

Liu, G., Xu, X. & Wang, F. 2012. A parameter estimation method of nonlinear system. 2012 *IEEE Symposium on Electrical & Electronics Engineering*, (*EEESYM*), pp. 396–398.

Liu, M., Xiao, Y. & Ding, R. 2013. Iterative identification algorithm for Wiener nonlinear systems using the Newton method, *Applied Mathematical Modelling*, 37, (2013), pp. 6584–6591.

Liu, G., Xu, X. & Wang, F. 2012. A parameter estimation method of nonlinear system. In 2012 *IEEE Symposium on Electrical* & *Electronics Engineering* (EEESYM), pp. 696–698.

Ljung, L. 1978. Convergence analysis of parametric identification methods. *IEEE Transaction in Automatic Control* AC-23, pp. 770–783.

Lohmann, Th.W., Bock, H. G. & Schloder, J. P. 1992. Numerical methods for parameter estimation and optimal experiment design in chemical reaction systems. *Industrial & Engineering Chemical Research* 1992, 31, pp. 54–57.

Lototsky, S.V. & Rosovskii, B.L. 2000. Parameter estimation for stochastic evolution equations with non-commuting operators. In: V. Korolyuk, N. Portenko, and H. Syta (eds). *Skorokhod's Ideas in Probability Theory*, pp. 271–280. Institute of Mathematics of the National Academy of Sciences of Ukraine, Kiev, Ukraine, 2000.

Lu, J., Feng, C., Xu, S. & Chu, Y. 2006. Observer design for a class of uncertainstate-delayed nonlinear systems. *International Journal of Control Automation & Systems*, 4, (4), pp. 448–455, August 2006.

Lu, S., Ju, K.H., & Chon, K.H. 2001. A new algorithm for linear and nonlinear ARMA model parameter estimation using affine geometry. *IEEE Transactions on Biomedical Engineering*, 48, (10), October 2001, pp. 1116–1124.

Luenberger, D.G. 1964. Observing the state of a linear system. *IEEE Transactions on Military Electronics*, 1964, pp. 74–80.

Luus, R. 2001. Comments on "Global optimization for the parameter estimation of differential-algebraic systems". (Correspondence), *Industrial & Engineering Chemistry Research*, 40, pp. 488–489.

Luenberger, D.G. 1971. An introduction to observers. *IEEE Transactions on Automatic Control*, AC–16, (6), December 1971, pp. 596–602.

Luyben, W.L. 1990. *Process modeling, simulation and control for chemical engineers*, 2nd Edition. New York: McGraw Hill Publishing Co.

Luyben, W.L. 1990. *Process modeling, simulation and control for chemical engineers*, 2nd Edition. McGraw Hill Publishing Co.

Luyben, W. 1973. *Process modeling, simulation and control for chemical engineers*. New York: McGraw Hill.

Lynch, A.F. & Bortoff, S.A. 1997. Nonlinear observer design by approximate error linearization. *Systems & Control Letters*, 32, (1997), pp. 161–172.

Mahmoud, M.S. 1982. Design of observer-based controllers for a class of discrete systems. *Automatica*, 18, (3), pp. 323–328. Great Britain: Pergamon Press Lid. International Federation of Automatic Control, 1982.

Mahmud, M.A., Hossain, M. J. & Pota, H. R. 2011. Nonlinear observer design for interconnected power systems. *In the Proceedings of* 2011 *Australian Control Conference*. 10-11 November 2011, Melbourne, Australia.

Mahyuddin, M.N., Na, J., Herrmann, G., Ren, X. & Barber, P. 2012. An adaptive observer-based parameter estimation algorithm with application to road gradient and vehicle's mass estimation. *UKACC International Conference on Control* 2012, Cardiff, UK, 3-5 September 2012, pp. 102–107.

Mansouri, M.M., Nounou, H.N., Nounou, M N. & Datta, A.A. 2014. State and parameter estimation for nonlinear biological phenomena modeled by S-systems. *Digital Signal Processing*, 28, (2014), pp. 1–17.

Marquardt D.W., 1963. An Algorithm for least squares estimation of nonlinear parameters. *SIAM Journal*, 11, pp. 431–441.

Marquez, H.J. 2003. *Nonlinear control systems: Analysis and design*. New Jersey (NJ): Wiley and Sons Inc.

Martin-Lof, P.E.R. 1966. The definition of random sequences. *Information & Control* 9, (1966), pp. 602–619

Marvel, S.W. & Williams, C.M. 2012. Set membership state and parameter estimation for nonlinear differential equations with sparse discrete measurements. 2012 *IEEE International Conference on Systems, Man, and Cybernetics*, October 14-17, 2012, COEX, Seoul, Korea, pp. 72–77.

Mathworks Inc. 2001. Optimization toolbox. For use with MATLAB. Mathworks Inc. http://www.mathworks.com/products/optimization/.

Maybeck, P.S. 1990. The Kalman filter: An introduction to concepts. *Autonomous Robot Vehicles*. Cox, I.J. & Wilfong, G.T. (eds). New York: Springer-Verlag, pp. 194–204.

Maybeck, P. 1982. S.*Stochastic models, estimation, and control*, *Vol. 2*, Maybeck, P.S. (ed). New York: Academic Press.

Maybeck, P.S. 1979. *Stochastic models*, *estimation and control*, *Vol*. 1. New York: Academic Press Inc.

Mc Garvey, F.X. & Gonzalez, R. 1992. Ion exchange studies on strongly basic anion exchange resins prepared with tertiary amines of varying molecular weight. In *Ion Exchange Advances: Proceedings of IEX '92*, M.J. Slater (Ed), pp. 97–103.

McCusker, J.R., Currier, T. & Danai, K. 2011. Improved parameter estimation by noise compensation in the time-scale domain. *Signal Processing*, 91, pp. 72–84.

McQuain, M. 2012. Constrained optimization: The method of Lagrange multipliers. http://www.math.vt.edu/people/mcquain/1526_Lag_opt_2012.pdf [Accessed: 31 July 2016].

Meinhold, R.J. & Singpurwalla, N.D. 1983. Understanding the Kalman filter. *The American Statistician*, (an American Statistical Association publication), 37, (2), May 1983, 2006, pp.123–127. http://www.jstor.org/

Mendel, J. 1971. Computational requirements for a discrete Kalman filter. *IEEE Transactions on Automatic Control*, AC-16, (6), December 1971, pp. 748–758.

Mendow, G., Grosso, C.I., Sanchez, A. & Querini, C.A. 2017. Hybrid process for the purification of water contaminated with nitrites: Ion exchange plus catalytic reduction. *Chemical Engineering Research & Design*, 125, September 2017, pp. 348–360. https://doi.org/10.1016/j.cherd.2017.07.019

Meng, L., Li, L. & Veres, S.M. 2010. Aerodynamic parameter estimation of an unmanned aerial vehicle based on extended Kalman filter and its higher order approach. *IEEE*'2010, pp. 526–531.

Meskin, N., Nounou, H., Nounou, M. & Datta, A. 2013. Parameter estimation of biological phenomena: An unscented Kalman filter approach. *IEEE/ACM Transactions* on *Computational Biology* & *Bioinformatics*, 10, (2), March/April 2013, pp. 537–543.

Michalik, C., Chachuat, B. & Marquardt, W. 2009a. Incremental global parameter estimation in dynamical systems, *Industrial & Engineering Chemistry Research*, 48, pp. 5489–5497.

Michalik, C., Hannemann, R. & Marquardt, W. 2009b. Incremental single shooting: A robust method for the estimation of parameters in dynamical systems. *Computers and Chemical Engineering*, 33, (2009), pp. 1298–1305.

Miguez, J., Marino, I.P. & Vazquez, M.A. 2018. Analysis of a nonlinear importance sampling scheme for Bayesian parameter estimation in state-space models. *Signal Processing*, 142, (2018), pp. 281–291.

Millar, G.J., Miller, G.L., Couperthwaite, S.J., Dalzell, S. & Macfarlane, D. 2017. Determination of an engineering model for exchange kinetics of strong acid cation resin for the ion exchange of sodium chloride & sodium bicarbonate solutions. *Journal of Water Process Engineering*, 17, June 2017, pp. 197-206. https://doi.org/10.1016/j.jwpe.2017.04.011

Misawa, E.A. & Hedrick, J.K. 1989. Nonlinear observers–A state-of-the-art survey. *Journal of Dynamic Systems, Measurements and Control* [Electronic Resource], *ASME* (*American Society of Mechanical Engineers*), 1989, 113, (3), pp. 344–352.

Moraal, P.E. & Grizzle, J.W. 1995. Observer design for nonlinear systems with discrete-time measurements. *IEEE Transactions on Automatic Control*, 40, pp. 395–404.

Mortensen, R.E. 1968. Maximum-Likelihood recursive nonlinear filtering. *Journal of Optimization & Theory Applications*, November 1968, 2, (2), pp. 386–394.

Mukkula, A.R.G. & Paulen, R. 2017. Model-based design of optimal experiments for nonlinear systems in the context of guaranteed parameter estimation. *Computers & Chemical Engineering*, 99, (2017), pp. 198–213.

Muske, K.R. & Edgar, T.F. 1997. Nonlinear state estimation. In Henson, M.A. & Seborg, D.E. (eds). *Nonlinear process control*. Upper Saddle River, New Jersey: Prentice Hall, pp. 311–370.

Na, J., Herrmann, G., Ren, X., Mahyuddin, M.N. & Barber, P. 2011. Robust adaptive finite-time parameter estimation and control of nonlinear systems. 2011 *IEEE International Symposium on Intelligent Control* (ISIC), Part of 2011 *IEEE Multi-Conference on Systems & Control*, Denver, CO, (USA). September 28-30, 2011, pp. 1014–1019.

Naidu, D.S. 2003. *Optimal control systems*. Florida, (USA): CRC Press LLC.

Nakamori, S. & Hataji, A. 1982. Relation between filter using covariance information and Kalman filter. *Automatica*, 18, (4), pp. 479–483. Great Britain: 1982 International Federation of Automatic Control.

Nardi, M. & Hammouri, H. 2003. Design of a continuous-discrete observer for state affine systems. *Applied Mathematics Letters*, 16, (2003), pp. 967–974.

Nicholson, B., López-Negrete, R. & Biegler, L.T. 2014. On-line state estimation of nonlinear dynamic systems with gross errors, *Computers & Chemical Engineering*, 70, (2014), pp. 149–159.

Nicosia, S., Tomei, P. & Tornambe, A. 1989. An approximate observer for a class of nonlinear systems. *Systems & Control Letters*, 12, (1989), pp. 43–51.

Noh, D., Jo, N.H. & Seo, J.H. 2004. Nonlinear observer design by dynamic observer error linearization. *IEEE Transactions on Automatic Control*, 49, (10), October 2004, pp. 1746–1750.

Norgaard, M. Poulsen, N.K. & Ravn, O. 2000. New developments in state estimation for nonlinear systems. *Automatica*, 36 (2000), pp. 1627–1638. http://www.elsevier.com/locate/automatica.

Norton, J.P. 1986. *An Introduction to identification*. London: Academic Press.

Ober, R.J., Lin, Z. & Zou, Q. 2005. On the identifiability of bilinear systems, In Proceedings of *IEEE Conference on Circuits and Systems* (ISCAS 2005), pp. 3769–3772.

Olsson, G. & Newell, B. 1999. *Wastewater Treatment Systems. Modelling, Diagnosis and Control*. London (UK): IWA Puplishing.

Ortega, R., Bobtsov, A., Pyrkin, A. & Aranovskiy, S. 2015. A parameter estimation approach to state observation of nonlinear systems. *Systems & Control Letters*, 85, November 2015, pp. 84–94.

Pagilla, P.R. & Zhu, Y. 2004. Controller and observer design for Lipschitz nonlinear systems. *Proceedings of the* 2004 *American Control Conference*, *Boston*, *Massachusetts*, (USA), ThA13.2, pp. 2379–2384, June 30-July 2, 2004.

Palm III, W.J. 1986. *Control system engineering*. New York: John Wiley & Sons.

Pearson, R.K. & Ogunnaike, B.A. 1997. Nonlinear process identification. In Henson, M.A. & Seborg, D.E. (eds). *Nonlinear process control*. Upper Saddle River, New Jersey: Prentice Hall, 11–110.

Park, J., Lee, G. & Choi, Y. 2015. Determining ARMA model parameters for biomedical signal applications based on estimation of the equivalent AR model. *The* 12th *International Conference on Ubiquitous Robots and Ambient Intelligence* (URAI 2015), October 28–30, 2015, KINTEX, Goyang city, Korea, pp. 456–457.

Parslow, J., Cressie, N., Campbell, E.P., Jones, E. & Murray, L. 2013. Bayesian learning and predictability in a stochastic nonlinear dynamical model, *Ecological Applications*, 23, (4), 2013, pp. 679-698 ©2013 by the Ecological Society of America.

Peng, H., Ozaki, T., Toyoda, Y. & Oda, K. 2001. Modeling and control of systems with signal dependent nonlinear dynamics, *European Control Conference* (*ECC*), Porto, Portugal, 4–7 September, 2001, pp 42–47.

Pertew, A.M., Marquez, H & Zhao, Q. 2005a. Dynamic observers for nonlinear lipschitz systems. *Proceedings of* 16th *IFAC World Congress*, Prague, Czech Republic, 2005.

Pierre, A.D. 1969. *Optimization theory with applications*. New York: John Wiley and Sons, Inc.

Polak, E. 1997. *Optimization: Algorithms and consistent approximation.* Applied mathematics sciences, 124, Marsden, J.E, Sirovich, L. & (the late) John, F (eds). New York: Springer-Verlag.

Pottmann, M.H., Unbehauen, H. & Seborg D.E. 1993. Application of a general multi-model approach for identification of highly nonlinear process–A case study. *International Journal of Control*, 57, (1), pp. 97–120.

Poyton, A.A., Varziri, M.S., McAuley, K.B., McLellan, P.J., & Ramsay, J. O. 2006. Parameter estimation in continuous-time dynamic models using principal differential analysis. *Computers & Chemical Engineering*, 30, pp. 698–708.

Primbs, J. 1996. Survey of nonlinear observer design techniques. Citeseerx 6M. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.53.7247&rep=rep1&type=pdf.

Pun, L. 1969. *Introduction to optimization practice*. New York: John Wiley & sons Inc.

Rad, C.-R. & Hancu, O. 2017. An improved nonlinear modelling and identification methodology of a servo-pneumatic actuating system with complex internal design for high-accuracy motion control applications. *Simulation Modelling Practice & Theory*, 75 (2017), pp. 29–47.

Raghavan, S. & Hedrik, J.K. 1994. Observer design for a class of nonlinear systems. *International Journal of Control*, 59, (2), 1994, pp. 515–528.

Randall, E.W. 1984. A microprocessor-based data monitoring and control system for a continuous ion exchange plant, *Desalination*, 49, pp. 169–184.

Rawlings, J.B. & Lima, F.V. 2008. State estimation of linear and nonlinear dynamic systems. Part III: Nonlinear Systems: Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF). *Department of Chemical and Biological Engineering*, University of Wisconsin, Madison, AICES Regional School, RWTH Aachen, March 17, 2008, pp. 1–18.

Ray, G. & De, S. 1997. Online parameter estimation and two–level control of large–scale discrete time systems. *Kybernetika*, (1997) 33, (4), pp. 427–443.

Rhodes, I.B. 1979. A tutorial Introduction to estimation and filtering. *IEEE Transactions on Automatic Control*, AC-16, (6), December 1971, pp. 688–706.

Robenack, K. & Lynch, A.F. 2006. Observer design using a partial nonlinear observer canonical form. *International Journal of Applied Maths & Computer Science*, 2006, 16, (3), pp. 333–343.

Robertson, D.G. & Lee, J.H. 1995. A least squares formulation for state estimation. *Journal of Process Control*, 5, (4), pp. 291-299, Great Britain: Butterworth & Heinemann: Elsevier Science Ltd, (c) 1995.

Roberts, P.D. & Williams, T.W.C. 1981. On an algorithm for combined system optimisation and parameter estimation. *Automatica*, (1), pp 199–209. Pergamon Press Ltd, 1981 international Federation of Automatic Control.

Rochette, F. 2006. Novel technology: Technology review: Fresh approach to countercurrent ion exchange. *Filtration & Separation*, 43, (7), September 2006, pp. 18–19. https://doi.org/10.1016/S0015-1882(06)70946-6

Romanenko, A. & Castro, J.A.A.M. 2004. The unscented filter as an alternative to the EKF for nonlinear state estimation: A simulation case study. *Computers & Chemical Engineering*, 2004, 28, (3), 347–355.

Romanenko, A., Santos, L.O. & Afonso, P.A.F.N.A. 2004. Unscented Kalman filtering of a simulated pH System. *Industrial & Engineering Chemistry Research,* 43*,* pp. 7531–7538.

Romeres, D., Prando, G., Pillonetto, G. & Chiuso, A. 2016. On-line Bayesian system identification, 2016 *European Control Conference* (*ECC*), June 29–July 1, 2016, Aalborg, Denmark, pp. 1359–1364.

Routtenberg, T. & Tabrikian, J. 2010. Optimal Bayesian parameter estimation with periodic criteria. In *Proc. IEEE Sensor Array and Multichannel Signal Process. Workshop (SAM'10)*, October 2010, pp. 53–56.

Rowell, D. 2004. Discrete time observers and LQG control. Massachusetts Institute of Technology Department of Mechanical Engineering 2.151 Advanced System Dynamics and Control, 12/13/04. http://web.mit.edu/2.151/www/Handouts/Kalman.pdf. [Date Accessed: 20 December 2015].

Roxin, E.O. 1997. *Control theory and its applications*. *Stability and control: theory, methods and applications*, *Volume*. 4. A series of books and monographs on theory of stability and control, Martynyuk, A.A. & Lakshmikantham, V. (eds). The Netherlands: Gordon and Breach Science Publishers.

Rubinstein, H. 1978. Smoothing properties of discrete-time zero-lag Kalman filter. *Automatica*, 14, pp. 397–401. Great Britain: International Federation of Automatic Control.

Rumschinski, P., Shona-Laila, D., Borchers, S. & Findeisen, R. 2010. Influence of discretization errors on set-based parameter estimation. *In the proceedings* of 49th *IEEE Conference on Decision* & *Control*, December 15-17, 2010, Atlanta, GA, USA, pp. 296–301.

Saab, S.S. 2004. A heuristic Kalman filter for a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 49, (12), December 2004, pp. 2261–2265.

Saif, M. 1993. A disturbance accommodating estimator for bilinear systems. *Proceedings of American Control Conference*, San Francisco, California, 1993, Paper WP 11–16:30, pp. 945–949.

Safdarnejad, S.M. Gallacher, J.R. & Hedengren, J.D. 2016. Dynamic parameter estimation and optimization for batch distillation, *Computers & Chemical Engineering*, 86, (2016), pp. 18–32.

Salau, N.P.G., Trierweiler, J.O. & Secchi, A.R. 2014. Observability analysis and model formulation for nonlinear state estimation, *Applied Mathematical Modelling*, 38, (2014) pp. 5407–5420.

Salhi, H. & Kamoun, S. 2015. A recursive parametric estimation algorithm of multivariable nonlinear systems described by Hammerstein mathematical models, *Applied Mathematical Modelling*, 39, (2015), pp. 4951–4962.

Safdarnejad, S.M. Gallacher, J.R. & Hedengren, J.D. 2016. Dynamic parameter estimation and optimization for batch distillation. *Computers & Chemical Engineering*, 86, (2016), pp. 18–32.

Sanfelice, R.G. & Praly, L. 2009. Nonlinear observer design with an appropriate Riemannian metric. Joint 48th *IEEE Conference on Decision & Control* and 28th *Chinese Control Conference*, Shanghai, P.R. China, December 16–18, 2009, FrA07.2, pp. 6514–6519.

Sargantanis, J.G. & Karim, M.N. 1994. Multivariable iterative extended Kalman filter based adaptive control: Case study of solid substrate fermentation. *Industrial & Engineering Chemistry Research*, 33, pp. 878–888.

Sarkka, S. 2007. On Unscented Kalman filtering for state estimation of continuous-time nonlinear systems. *IEEE Transactions on Automatic Control*, October 2007, pp. 1–11.

Sarris, A. & Eisner, H.M. 1973. Parameter estimation of ARMA models using a computationally efficient maximum likelihood technique, *IEEE Conference on Decision and Control including the* 12th *Symposium on Adaptive Processes*, CDC (FA3–4), 1973, pp. 640–644.

Sasan, K., Brady, P.V., Krumhansl, J.L. & Nenoff, T.M. 2017. Removal of dissolved silica from industrial waters using inorganic ion exchangers. *Journal of*

Water Process Engineering, 17, June 2017, pp. 117–123. https://doi.org/10.1016/j.jwpe.2017.02.006.

Sawant, J. & Ginoya, D. 2010. dSPACE DSP DS-1104 based state observer design for position control of DC Servo Motor. dSPACE *User Conference* 2010 – India, 24 September 2010.

Schetzen, M. A 1974. Theory nonlinear system identification. *International Journal of Control*, 20, (4), pp. 577–592.

Schei, T.S. & Singstad, P. 1998. Nonlinear model predictive control of a batch polymerization process. Proceedings of the *American Control Conference*, Philadelphia, Pennsylvania, June 1998, pp. 3381–3385.

Schei, T.S. 2008. On-line estimation for process control and optimization applications. *Journal of Process Control*, 18, (2008), pp. 821–828.

Schennach, S.M. 2004. Estimation of nonlinear models with measurement error. *Econometrica*, 72, (1), (January, 2004), pp. 33–75.

Schetzen, M. A. 1974. Theory nonlinear system identification. *International Journal Control*, 20, (4), pp. 577–592.

Schoukens, J., Pintelon, R., Dobrowiecki, T. & Rolain, Y. 2005. Identification of linear systems with nonlinear distortions. *Automatica*, 41, (2005), pp. 491–504. www.elsevier.com/locate/automatica.

Schindler, M.R. & Phillips, D.R. 2009. Bayesian methods for parameter estimation in effective field theories. *Annals of Physics*, 324, (2009), pp. 682–708.

Schultze, M. & Horn, J. 20156. Modeling, state estimation and nonlinear model predictive control of cathode exhaust gas mass flow for PEM fuel cells, *Control Engineering Practice*, 49, (2016), pp. 76–86.

Schwaab, M., Biscaia Jr., E.C., Monteiro, J.L. & Pinto, J.C. 2008. Nonlinear parameter estimation through particle swarm optimization. *Chemical Engineering Science*, 63 (2008), pp. 1542–1552.

Seong, S.M. 2007. A Modified direct whitening method for ARMA model parameter estimation. *International Conference on Control, Automation and Systems*, 2007, October, 17–20, 2007, COEX, Seoul, Korea, pp. 2639–2642.

Seborg, D.A. & Henson, M.A. 1997. Introduction. In Henson, M.A. & Seborg, D.E. (eds). *Nonlinear process control*. Upper Saddle River, New Jersey: Prentice Hall, pp. 1–9.

Shahrokhi, M. & Fanaei, M.A. 2001. State estimation in a batch suspension polymerization reactor. *Iranian Polymer Journal*, 16 (3), (2001), pp. 173–187.

Shanshiashvilia, B. & Prangishvili, A. 2017. Structure identification of continuous nonlinear dynamical systems. *Procedia Computer Science*, 112, (2017), pp. 1032–1043. www.wlsevier.com/locate/procedia.

Shawash, J. & Selviah, D.R. 2013. Real-time nonlinear parameter estimation using the Levenberg–Marquardt algorithm on field programmable gate arrays. *IEEE Transactions on Industrial Electronics*, 60, (1), January 2013, pp. 170–176.

Shim, H. & Seo, J.H. 2003. Recursive observer design: Beyond the uniform observability. *IEEE Transactions on Automatic Control*, 48, (2), February 2003, pp. 294–298.

Shyamalagowri, M. & Rajeswari, R. 2016. Unscented Kalman filter based nonlinear state estimation case study–nonlinear process control reactor (continuous stirred tank reactor), 2016 10th *International Conference on Intelligent Systems & Control* (ISCO), 7–8 January 2016, Coimbatore, India, pp. 1–6.

Si, X-S., Wang, W., Hua, C-H. & Zhou, D-H. 2011. Remaining useful life estimation – A review on the statistical data driven approaches. *European Journal of Operational Research*, 213, (2011), pp. 1–14. www.elsevier.com/locate/ejor.

Simon, E. & Bertino, L. 2012. Gaussian anamorphosis extension of the DEnKF for combined state parameter estimation: Application to a 1D ocean ecosystem model. *Journal of Marine Systems*, 89, (2012), pp. 1–18. www.elsevier.com/locate/jmarsys. DOI:10.1016/j.jmarsys.2011.07.007.

Singh, A.K. & Hahn, J. 2005. Determining optimal sensor locations for state and parameter estimation for stable nonlinear systems. *Industrial & Engineering Chemistry Research*, 44, pp. 5645–5659.

Siouris, G. M. 1996. *An Engineering Approach to Optimal Control and Estimation Theory*. New York: John Wiley and Sons.

Slater, M.J. 1974. Continuous ion exchange in fluidized beds. *The Canadian Journal of Chemical Engineering*, 52, (1), February 1974, pp. 43–51.

Slotine, J.E. & Li W. 1991. *Applied nonlinear control*. New Jersey: Prentice Hall.

Snijders, J.G., van der Woude, J.W. & Westhuis, J. 2005. Nonlinear observer design for dynamic positioning. *Dynamic Positioning Conference–Control Systems I*, November 15-16, 2005. Dyanmic Positioning Committee, Marine Technology Society.

Soderström, T. & Stoica, P. 1989. *System Identification*. [Prentice Hall International Series in Systems and Control Engineering. Series editor, Grimble, M.J], Cambridge (UK): Prentice Hall International.

Song, B. & Hedrick, J.K. 2011. Nonlinear observer design for Lipschitz nonlinear systems. 2011 *American Control Conference*, on O'Farrell Street, San Francisco, CA, USA, June 29–July 01, 2011, pp. 2578–2583.

Sorenson, H.W. 1985. *Kalman filtering: Theory and application*, New York: IEEE Press.

Sorenson, H. 1970. Least square estimation from Gauss to Kalman. *IEEE Spectrum*, 7, July 1970, pp. 63–68.

Soroush, M. 1998. State and parameter estimations and their applications in process control. *Computers & Chemical Engineering*, 23, (2), 1 December 1998, pp. 229–245

Souley Ali, H., Rafaralahy, H., Zasadzinski, M., Halabi, S. & Darouach, M. 2005. Observer design for a class of stochastic bilinear systems with multiplicative noise. Proceedings of (2005) *American Control Conference*, June 8–10, 2005, Portland, OR, USA, FrA07.6, pp. 3641–3642.

Specker, T., Buchholz, M. & Dietmayer, K. 2014. Combined state and parameter estimation for adaptive control and feedback applications for a gait rehabilitation robot. 2014 *IEEE Conference on Control Applications* (CCA), Part of 2014 *IEEE Multi-Conference on Systems and Control*, October 8-10, 2014. Antibes, France, pp. 1287–1293.

Srinivasan, B., Francois, G. & Bonvin, D. 2011. Comparison of gradient estimation methods for real-time optimization. 21st *European Symposium on*

Computer Aided Process Engineering, Pistikopoulos, E.N. Georgiadis, M.C. Kokossis, A.C. (eds), Computer Aided Chemical Engineering, 29, 2011, pp. 607–611.

Stecha, J. 1997. Stochastic optimal control of ARMAX model. 1997 *European Control Conference* (*ECC*), 1-4 July 1997, Brussels, Belgium, pp. 482–487.

Steinebach, F., Ulmer, N., Decker, L., Aumann, L. & Morbidelli, M. 2017. Experimental design of a twin-column countercurrent gradient purification process. *Journal of Chromatography* A, 1492, 7 April 2017, pp. 19–26. https://doi.org/10.1016/j.chroma.2017.02.049.

Stone, J.J. & Womack, B.F. 1970. Identification of a class of nonlinear systems by use of piecewise continuous expansions. *SWIEEECO Record*, pp. 198–200.

Straka, O., Dunik, J., Simandl, M. 2014. Unscented Kalman filter with advanced adaptation of scaling parameter. *Automatica*, 50, (2014), pp. 2657–2664.

Streat, M. 1995. The waters were made sweet. Advances in ion exchange technology. *Industrial & Engineering Chemistry Research*, 34, pp. 2841–2848.

Strejc, V. 1982. State space theory of discrete linear control. *IEEE Transactions on Systems Man and Cybernetics*, 12, (5), September, 1982, pp. 684–684. (Book Review).

Strejc, V. 1981a. Trends in identification. *Automatica*, 17, (1), pp. 7–21.

Strejc, V. 1981b. *State space theory of discrete linear control*. New York: Wiley-Interscience, March 1981, Circa 500.

Strejc, V. 1980. Least squares parameter estimation. *Automatic,* 16, (5), International Federation of Automatic Control, (1980), pp. 535–550.

Strejc, V. 1977. Least squares in identification theory, *Kybernetika*, 13, (2), (1977), pp. 83–105.

Strejc, V. 1972. State space synthesis of discrete linear systems, *Kybernetika*, 8, (2), (1972), pp. 83–113.

Sundarapandian, V. 2011a. Exponential observers for Lotka-Volterra systems. *International Journal on Computer Science and Engineering* (IJCSE), 3, (3) March 2011, pp. 1351–1362.

Sundarapandian, V. 2011b. New Results on nonlinear observer design for Continuous-Time Nonlinear Systems. *International Journal of Mathematical Sciences* & *Applications*, 1, (3), pp. 1147–1151, September 2011. Mind Reader Publications. www.journalshub.com.

Sundarapandian, V. 2011c. New results on nonlinear observer design for discrete-time nonlinear systems. *International Journal of Mathematical Sciences* and *Applications*, 1, (3), pp. 1153–1157, September 2011. Mind Reader Publications. www.journalshub.com.

Sundarapandian, V. 2004. General observers for nonlinear systems, *Mathematical & Computer Modelling*, 39, (2004), pp. 97–105.

Sundarapandian, V. 2002. Observer design for discrete-time nonlinear systems. *Mathematical & Computer Modeling*, 35, (2002), pp. 37–44.

Sundstrom, D.W. & Klei, H. E. 1979. *Wastewater treatment*. Englewood Cliffs, NJ: Prentice Hall.

Suzdaleva, E. & Nagy, I. 2012. Recursive state estimation for hybrid systems, *Applied Mathematical Modelling*, 36, (2012), pp. 1347–1358.

Swerling, P. 1971. Modern state estimation methods from the viewpoint of the method of least squares. *IEEE Transactions on Automatic Control*, AC-16, (6), December 1971, pp. 707–719.

Tang, Y., Han, Z., Liu, F. & Guan, X. 2016. Identification and control of nonlinear system based on Laguerre-ELM Wiener model, *Commun Nonlinear Sci Numer Simulat*, 38, (2016), pp. 192–205.

Taniguchi, T., Eciolaza, L. & Sugeno, M. 2014. Full-order state observer design for nonlinear systems based on piecewise bilinear models. *International Journal of Modeling and Optimization*, 4, (2), April 2014, pp. 120–125.

Tashkova, K., Silc, J., Atanasova, N. & Dzeroski, S. 2012. Parameter estimation in a nonlinear dynamic model of an aquatic ecosystem with meta-heuristic optimization. *Ecological Modelling*, 226, (2012) pp. 36–61. www.elsevier.com/locate/ecolmodel

Tatiraju, S. & Soroush, M. 1998. Parameter estimator design with application to a chemical reactor. *Industrial & Engineering Chemistry Research*, 37, pp. 455–463.

Tatiraju, S. & Soroush, M. 1997a. Parameter estimation via inversion with application to a chemical reactor. *Proceedings of the American Control Conference*, (1997), 4, pp. 2429–2433.

Tatiraju, S. & Soroush, M. 1997b. Nonlinear state estimation in a polymerization reactor. *Industrial & Engineering Chemistry Research*, 36, (1997), pp. 2679–2690.

Telen, D., Vallerio, M., Cabianca, L., Houska, B., Van Impe, J., & Logist, F. 2015. Approximate robust optimal control of nonlinear dynamic systems under process noise. 2015 *European Control Conference* (*ECC*), July 15-17, 2015. Linz, Austria, pp. 1581–1586.

Thangavel, S., Paulen, R. & Engell, S. 2016. Robust nonlinear model predictive control with reduction of uncertainty via dual control. 2017 21st *International Conference on Process Control* (PC), June 6–9, 2017, Strbske Pleso, Slovakia, pp. 48–53.

Thathachar M.A.L. & Ramaswamy S. 1973. Identification of class nonlinear systems. *International Journal of Control*, 18, (4), pp. 741–752.

Thau, F.E. 1973. Observing the state of nonlinear dynamic systems. *International Journal of Control*, 1973, 17, (3), pp. 471–479.

Tibken, B. & Hofer, E.P. 1989. Systematic observer design for bilinear systems. *IEEE ISCAS '89*, *IEEE International Symposium on Circuits & Systems,* 4, pp. 1611–1616.

Treybal, R.E. 1980. *Mass transfer operations*, 3rd Ed. International Edition Singapore: McGraw-Hill.

Trinh, H., Fernando, T. & Nahavandi, S. 2006. Partial-state observers for nonlinear systems. *IEEE Transactions on Automatic Control*, 51, (11), November 2006, pp. 1808–1812.

Tse, E. & Athans, M. 1973. Observer theory for continuous-time linear systems. *Information & Control*, 22, (1973), pp. 405–434.

Tse, E. & Athans, M. 1970. Optimal minimal-order observer-estimators for discrete linear time-varying systems. *IEEE Translations on Automatic Control*, AC-15, (4), August 1970, pp. 416–426.

Tsinias, J. 1990. Further results on the observer design problem. *Systems & Control Letters*, 14, (5), pp. 411–418, June 1990.

Tsinias, J. 1989. Observer design for nonlinear systems. *Systems & Control Letters*, 13 (1989), pp. 135–142.

Tulsyana, A., Huanga, B., Gopalunib, R.B. & Forbes, J.F. 2013. On simultaneous on-line state and parameter estimation in non-linear state-space models, *Journal of Process Control*, 23, (2013), pp. 516–526.

Tzoneva R. 2004. Control systems IV: Class Notes for Electrical Engineering students. Bellville: Peninsula Technikon.

Tzoneva R. 2000. Control systems IV: Class Notes for Electrical Engineering students. Bellville: Peninsula Technikon.

Ubeda, D., Pinar, F.J., Canizares, P., Rodrigo, M.A., & Lobato, J. 2012. An easy parameter estimation procedure for modeling a HT-PEMFC. *International Journal of Hydrogen Energy*, 37, (2012), pp. 11308–11320. www.elsevier.com/locate/he.

Ugrinoskii, V.A. 2003. Observability of linear stochastic uncertain systems. *IEEE Transactions on Automatic Control*, 47, (12), December 2003, pp. 2264–2269.

UmaMageswari, A. Ignatious, J.J. & Vinodha, R. 2012. A comparitive study of Kalman filter, extended Kalman filter and unscented Kalman filter for harmonic analysis of the non-stationary signals. *International Journal of Scientific & Engineering Research*, 3, (7), July-2012, pp. 1–9.

Unbahanen, H & RAO, G. 1990. Continuous time approach to system identification–A Survey. *Automatica*, 26, pp. 23–35.

Ungarala, S., Chen, Z. & Li, K. 2006. Bayesian state estimation of nonlinear systems using approximate aggregate Markov chains. *Industrial & Engineering Chemistry Research*, 45, pp. 4208–4221.

Vafamand, N. & Safarinejadian, B. 2013. State and parameter estimation of CSTR using joint-UKF. 3rd *International Conference on Control, Instrumentation, and Automation* (ICCIA 2013), December 28–30, 2013, Tehran, Iran, pp. 165–169.

Vajda, S., Valkos, P. & Godfrey, K.R. 1987. Direct and indirect least squares methods in continuous-time parameter estimation. *Automatica,* 23, (6), International Federation of Automatic Control, (1987), pp. 707–718.

Varziri, M.S., McAuley, K.B. & McLellan, P.J. 2008a. Parameter estimation in continuous-time dynamic models in the presence of unmeasured states and nonstationary disturbances. *Industrial & Engineering Chemical Research*, 47*,* pp. 380–393.

Varziri, M.S., McAuley, K.B. & McLellan, P.J. 2008b. Approximate maximum likelihood parameter estimation for nonlinear dynamic models: Application to laboratory-scale nylon reactor model. Industrial & Engineering Chemical Research, 47, pp. 7274–7283.

Vassilaki, M & Bitsoris, G. 1999. Constrained feedback control of discrete-time systems described by ARMA models. 1999 *European Control Conference* (*ECC*), 31 August–3 September 1999, Karlsruhe, Germany, pp. 1027–1031.

Vayssettes, J., Mercere, G., Bury, Y. & Pommier-Budinger, V. 2015. Structured model identification algorithm based on constrained optimisation. 2015 *European Control Conference* (*ECC*), July 15-17, 2015. Linz, Austria, pp. 1285–1290.

Vries, D., Keesman, K.J. & Zwart, H. 2007. A Luenberger observer for an infinite dimensional bilinear system: A UV disinfection example. Preprints of the 3rd *IFAC Symposium on System, Structure & Control*, SSSC07, Foz do Iguassu, Brazil, October 17-19th, 2007.

Vries, D., Keesman, K.J. & Zwart, H. 2007. A Luenberger observer for an infinite dimensional bilinear system: A UV disinfection example., Preprints of the *3rd IFAC Symposium on System, Structure & Control*, SSSC07, Foz do Iguassu, Brazil, October 17-19th, 2007.

Vuchkov I.N., Velev K.D. & Tsochev V.K. 1985. Identification of Parametrically Dependent Process with Applications to Chemical Technology. Prepr. 7th *IFAC/IFORS Symposium on Identification and System Parameter Estimation* (York: UK), pp. 1089–1093.

Walcott, B.L. & Zak, S.H. 1987. State observation of nonlinear uncertain dynamical systems. *IEEE Transactions on Automatic Control*, AC-32, (2), February 1987, pp 166–170.

Walter, E. & Pronzato, L. 1995. Identifiabilities and nonlinearities. In Fossard, A.J. & Normand-Cyrot, D. (eds). *Nonlinear Systems: Modeling and estimation. Volume 1*. London (UK): Chapman & Hall, pp. 111-143. [English language translation revised by Mrs M.B. Groen-Garrer. Original French language edition: *Systèmes non linéaires*, coordonné par A.J. Fossard et D. Normand-Cyrot. (1993), France: Masson Éditeur].

Wang, J. 1996. Asymptotics of least-squares estimators for constrained nonlinear regression. *The Annals of Statistics*, 1996, 24, (3), pp. 1316–1326.

Wang, Z. & Burnhan, K.J. 2001. Discrete-time reduced-order estimator design for bilinear stochastic systems with error covariance assignment. *IMA Journal of Mathematical Control* & *Information*, (2001), 18, pp. 99–107.

Wang, H. Chen, N. & Ma, C. 2010. A brief review on item response theory model-based parameter estimation methods. *The* 5th *International Conference on Computer Science* & *Education* Hefei, China. August 24–27, 2010, WeM1.5, pp., 19–22.

Wang, D., Dai, J. & Ding, F. 2009. Gradient-based iterative parameter estimation for Box-Jenkins systems with finite measurement data. *Joint* 48th *IEEE Conference on Decision* and *Control* and 28th *Chinese Control Conference* Shanghai, P.R. China, December 16-18, 2009, WeA07.5, pp. 239–243.

Wang, Z. & Ho, D.W.C. 2003. Filtering on nonlinear time-delay stochastic systems. *Automatica*, 39, (2003), pp. 101–109. www.elsevier.com/locate/automatica.

Wang, F.-S., Su, T.-L., & Jang, H.-J. 2001. Hybrid differential evolution for problems of kinetic parameter estimation and dynamic optimization of an ethanol fermentation process. *Industrial* & *Engineering Chemistry Research* (2001), 40, pp 2876– 885.

Wang, Q., Wang, L. & Shen, Q. 2016. Modeling strategy of high order ARMA model, Proceedings of 2016 *IEEE Chinese Guidance*, *Navigation and Control Conference*, 12-14 August 2016, Nanjing, China, pp. 2001–2005.

Wang, Z. & Zhang, H. 2006. Design of bilinear observer for singular bilinear systems. *Journal of Control Theory and Applications*, 4, 2006, pp. 413–417.

Wang, D. & Zhang, W. 2015. Improved least squares identification algorithm for multivariable Hammerstein systems. *Journal of the Franklin Institute*, 352, (11), November 2015, pp. 5292–5307.

Weiss, M. & Preisig, H.A. 1998. Nonlinear system analysis applied to the numerical conditioning of dynamical models for physical processes. *Proceedings of the American Control Conference*, Philadelphia, Pennsylvania–June 1998, pp. 2667–2671.

Welch, G.B. 2001. *An Introduction to the Kalman filter*. ACM, Inc.

Welch, G. & Bishop, G. 2006. An Introduction to the Kalman filter. University of North Carolina, Chapel Hill, *Internal Report*, TR 95-041, July 24, 2006, pp. 1–16. Available from: http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf.

Williamson, D. 1977. Observation of bilinear systems with application to biological control. *Automatica*, 13, (1977), pp. 243–254.

Xia, X.H. & Gao, W.B. 1988. Nonlinear observer design by observer canonical forms. *International Journal of Control*, 47, (1988), pp. 1081–1100.

Xia, Q., Rao, M., Ying, Y. & Shen, X. 1994. Adaptive fading Kalman filter with an application. *Automatica*, 30, (8), pp. 1333–1338.

Xiang, J., Mueller, M.A. & Cheng, D. 2007. Parameter estimation of an electro-mechanical system using narrow-band data. *Dynamics of Continuous, Discrete and Impulsive Systems Series* B: *Applications & Algorithms* 14 pp. 15–25.

Xiao, M.-Q. 2006. A direct method for the construction of nonlinear discrete time observer with linearizable error dynamics. *IEEE Transactions on Automatic Control*, 51, pp. 128–135.

Xiao, M.-Q. 2005. A direct construction of nonlinear discrete-time observer with linearizable error dynamics. *Proceedings in 2005 American Control Conference*, 2005, Portland, OR (USA), June 8–10.

Xiao, M.-Q. 2005. A direct construction of nonlinear discrete-time observer with linearizable error dynamics. *Proceedings in: 2005 American Control Conference*, 2005, Portland, OR, USA, June 8–10.

Xiao, M., Kazantzis, N., Kravaris, C. & Krener, A.J. 2001. Nonlinear discrete-time observer design with linearizable error dynamics. *Preprint*, September, 2001.

Xiao, M., Kazantzis, N., Kravaris, C. & Krener, A.J. 2001. Nonlinear discrete-time observer design with linearizable error dynamics. *Preprint*, September, 2001.

Xiong, Y. & Saif, M. 2001. Sliding mode observer for nonlinear uncertain systems. *IEEE Transactions on Automatic Control*, 46, pp. 2012–2017.

Xiong, Y. & Saif, M. 2001. Sliding mode observer for nonlinear uncertain systems. *IEEE Transactions on Automatic Control*, 46, pp. 2012–2017.

Xiong K., *Wei*, C.L & Liu, L.D. 2012. Robust Kalman filtering for discrete-time nonlinear systems with parameter uncertainties. Aerospace Science and Technology, 18, (2012), pp. 15–24. Aerospace Science & Technology. www.elsevier.com/locate/aescte.

Xu, H. & Mannor, S. 2009. A Kalman filter design based on the performance/robustness trade-off. *IEEE Transactions on Automatic Control*, 54, (5), May 2009, pp. 1171–1175.

Xu, J-X. & Xu, J. 2004. Observer based learning control for a class of nonlinear systems with time-varying parametric uncertainties. *IEEE Transactions on Automatic Control*, 49, (2), February 2004, pp. 275–281.

Yang, F.W., Wang, Z.D. & Hung Y.S. 2002. Robust Kalman filtering for discrete time-varying uncertain systems with multiplicative noises. *IEEE Transactions on Automatic Control*, 47, (7), pp. 1179−1183.

Yang, S., Hu*,* Y-H., Nguyen*,* T.Q. & Tull*,* D.L. 2001. Maximum-Likelihood parameter estimation for Image Ringing-Artifact (IRA) removal. *IEEE Transactions on Circuits and Systems for Video Technology*, 11, (8), August 2001.

Yang, X., Peng, B., Zhou, H. & L. Yang, 2016. State estimation for nonlinear dynamic systems using Gaussian processes and pre-computed local linear models. Proceedings of 2016 *IEEE Chinese Guidance, Navigation & Control Conference*, August 12-14, 2016 Nanjing, China, pp. 1963–1968.

Yaz, E.E. Jeong C.S. & Alotaibi M. 2004. Resilient design of Thau's observer using LMIs. In the *Proceedings of American Control Conference*, 4, pp. 3482–3483, July 2004.

Yaz, E.E. Jeong, C.S. Bahakeem, A. & Yaz, Y.I. 2007. Discrete-time nonlinear observer design with general criteria. *Journal of the Franklin Institute*, 344, (2007) pp. 918–928.

Ying, Y.Q., Rao, M. & Sun, Y.X. 1990. State-disturbance composite observer for bilinear systems. American Control Conference, 1990, San Diego, CA, USA, 23–25 May 1990.

Yoshikawa, T. & Kobayashi, H. Comments on 'Optimal minimal-order observer estimators for discrete linear time-varying systems, *IEEE Transactions on Automatic control*, April 1972, pp. 272–273.

Young, P. 1981. Parameter estimation for continuous-time models: A survey. *Automatica*, SPECIAL ISSUE: IDENTIFICATION AND SYSTEM PARAMETER ESTIMATION Guest Editor: Rolf Isermann. *Automatica*, 17, (1), pp. 23–39.

Yu, W. & Liu, D. 1999. Time variant parameter estimation of a nonlinear system using a Quasi-Newton method. Proceedings of 14th *World Congress of IFAC*, China, Vol. H, pp. 55–60.

Yuksel, Y.O. & Bongiorno Jr., J.J. 1971. Observers for linear multivariable systems with applications. *IEEE Transactions on Automatic Control*, AC-16, (6), December 1971, pp. 603–612.

Zadeh L.A. 1953. A Contribution theory of nonlinear systems. *Journal of the Franklin Institute*, 255, pp. 387–408.

Zeitz, M. 1987. The extended Luenberger observer for nonlinear systems", *Systems & Control Letters*, 9, (2), August 1987, pp. 149–156.

Zemouche, A. & Boutayeb, M. 2013. On LMI conditions to design observers for Lipschitz nonlinear systems. *Automatica*, 49, (2013), pp. 585–591.

Zemouche, A., Boutayeb, M. & Bara G.I. 2005. Observer design for nonlinear systems: an approach based on the differential mean value theorem. *Proceedings of the* 44th *IEEE Conference on Decision and Control, and the European Control Conference* 2005 Seville, Spain, December 12-15, 2005, pp. 6353–6358.

Zhang, W., Li, J., Su, H. & Zhu, F. 2014. Reduced-Order observer design for one-sided Lipschitz nonlinear systems with unknown inputs. Proceedings of the

33rd *Chinese Control Conference*, July 28–30, 2014, Nanjing, China, pp. 2352–2356.

Zhang, Z. & Xu, S. 2015. Observer design for uncertain nonlinear systems with unmodeled dynamics, *Automatica*, 51, (2015), pp. 80–84.

Zhang, X., Zhu, Y., Yan, W. & Shao, H. 2010. Nonlinear parameter prediction and estimation of fossil power plant based on kernel partial least squares. In the *Proceedings of the 2010 IEEE International Conference on Information & Automation*, June 20–23, Harbin, China, pp. 1964–1967.

Zhang, X., Zhu, Y., Yan, W. & Shao, H. 2010. Nonlinear parameter prediction and estimation of fossil power plant based on kernel partial least squares. In the *Proceedings of the 2010 IEEE International Conference on Information & Automation*, June 20 - 23, Harbin, China, pp. 1964–1967.

Zhao, C., An, A. & Xu, Q. 2012. A hybrid differential evolution algorithm for nonlinear parameter estimation of kinetic systems. *IEEE* 2012, pp. 1696–1700.

Zhou, J. 2013. Adaptive observer design and parameter estimation for a class of uncertain systems. 2013. *IEEE* 8th *Conference on Industrial Electronics & Applications* (ICIEA), pp. 148–153.

Zhao, Z., Huang, B. & Liu, F. 2013. Bayesian method for state estimation of batch process with missing data. *Computers & Chemical Engineering*, 53, (2013), pp. 14–24.

Zheng, G. & Boutat, D. 2014. Adaptive observer for simultaneous estimation of state and parameter for a class of nonlinear systems, Proceedings of the 33rd *Chinese Control Conference*, July 28-30, 2014, Nanjing, China, pp. 1876–1880.

Zhou, J. 2013. Adaptive observer design and parameter estimation for a class of uncertain systems. 2013. *IEEE* 8th *Conference on Industrial Electronics & Applications* (ICIEA), pp. 148–153.

Zhou, J. & Men, B. 2011. Partial state observer design for a class of nonlinear descriptor systems. *International Journal of Information & Systems Sciences*, 7, (4), Pages 357–369. 2011 Institute for Scientific Computing and Information.

Zhu, F. & Han, Z. 2002. A note on observers for Lipschitz nonlinear systems. *IEEE Transactions on Automatic Control*, 47, (10), October 2002, pp. 1751–1754.

Zhu, F. & Han, Z. 2002. A note on observers for Lipschitz nonlinear systsems. *IEEE Transactions on Automatic Control*, 47, (10), October 2002, pp. 1751–1754.

Zhu, Q.M. 2005. An implicit least squares algorithm for nonlinear rational model parameter estimation. *Applied Mathematical Modelling*, 29, (2005), pp. 673–689. www.elsevier.com/locate/apm.

Zhu, Y. & Pagilla, P.R. 2006. Adaptive controller and observer design for a class of nonlinear systems. *Transactions of the ASME*, *Journal of Dynamic Systems*, *Measurement*, & *Control*, 128, September 2006, pp. 712–717.

Zhuang, L., Pan, F. & Ding, F. 2012. Parameter and state estimation algorithm for single-input single-output linear systems using the canonical state space models, *Applied Mathematical Modelling*, 36, (2012), pp. 3454–3463.

APPENDICES

APPENDIX A: PROGRAMS FOR DATA AND MODEL REFORMULATION FOR ESTIMATION PROBLEMS

APPENDIX A.1: MATLAB software program for H^+ fractional change for the 6 stage system of a LOW to HIGH concentration step change

```
% Program used for manupulation of data from a Continuous Countercurrent Ion
% Exchange (CCIX) process used for desalination of water.
% The program tests the H+ LOW to HIGH step change in the feed concentration.
%
% Author: NM. Dube
% Dated: 16 April 2011
%
% Data vectors for a 6 stage system, T is the upflow period in (min),
% The first run is a LOW to HIGH (H+) step change in the feed concentration
%
cycle1 = 1:1.0:18;
T = 17;
cycle2 = T*cycle1;

% Stage liquid concentration data
 stage1 = [0.221 0.577 0.730 0.847 0.920 0.936 0.968 0.974 0.981 0.989 0.988
           0.997 1.000 1.000 1.000 1.000 1.000 1.000];
 stage2 = [0.000 0.140 0.314 0.523 0.656 0.766 0.842 0.886 0.900 0.933 0.958
           0.963 0.982 0.974 0.991 0.994 0.993 0.993];
 stage3 = [0.000 0.004 0.066 0.184 0.295 0.454 0.601 0.690 0.758 0.804 0.877
           0.881 0.951 0.965 0.972 0.981 1.000 0.988];
 stage4 = [0.000 0.000 0.004 0.035 0.082 0.168 0.277 0.361 0.440 0.522 0.698
           0.784 0.899 0.931 0.966 0.966 0.991 0.991];
 stage5 = [0.000 0.000 0.000 0.003 0.020 0.052 0.113 0.167 0.207 0.340 0.474
           0.547 0.780 0.860 0.899 0.905 0.975 0.973];
 stage6 = [0.000 0.000 0.000 0.000 0.000 0.001 0.024 0.033 0.063 0.124 0.167
           0.233 0.482 0.539 0.672 0.779 0.940 0.972];
%
% Graph plots
subplot (2,1,1,figure (1))
plot (cycle1,stage1,'-*','LineWidth',1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
```

```
%
% Labels for the plot and specifying font size for each
%
title('H+ fraction in liquid in a 6 stage system [a LOW-HIGH concentration step
change]','fontsize',16)%'FontWeight', 'bold')
xlabel('cycle number [number]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('H+ fraction in liquid [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
%subplot(3,2,2)
plot(cycle1,stage2,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage3,'-s','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage5,'-p','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
subplot (2,1,2,figure(1))
plot (cycle2,stage1,'-*','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
title('Liquid concentration per stage in a 6 stage system [a LOW-HIGH concentration
step change]','Fontsize',16)%,'FontWeight', 'bold')
xlabel('cycles [min]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Liquid concentration [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle2,stage2,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage3,'-s','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
```

```
%
plot(cycle2,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage5,'-p','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
%----------------------------------Program end-------------------------------------------
```

APPENDIX A.2: MATLAB software program for H^+ fractional change for the 6 stage system of a HIGH to LOW concentration step change

```
% Program used for manupulation of data from Continuous Countercurrent Ion
% Exchange (CCIX) process used for desalination.
% The program is an H+ HIGH to LOW step change in the feed concentration.
%
% Author: NM Dube
% Dated: 16 April 2011
%
% Data vectors for a 6 stage system, T is the upflow period in (min),
% The second run is a HIGH to LOW (H+) step change in the feed concentration
cycle1 = 1:1:27;
T = 15;
cycle2 = T*cycle1;
%
% Corrected values; values corrected using (interpolation averaging method)
stage1 = [0.014 0.002 0.031 0.036 0.063 0.062 0.088 0.119 0.152 0.183 0.209 0.240
          0.280 0.332 0.372 0.413 0.449 0.525 0.591 0.634 0.670 0.694 0.775 0.812
          0.836 0.903 0.931];
stage2 = [0.009 0.003 0.017 0.052 0.059 0.085 0.117 0.152 0.191 0.250 0.295 0.350
          0.490 0.476 0.538 0.577 0.645 0.664 0.752 0.780 0.809 0.829 0.868 0.920
          0.939 0.959 1.000];
stage3 = [0.014 0.021 0.057 0.079 0.126 0.170 0.235 0.312 0.403 0.471 0.591 0.631
          0.693 0.716 0.756 0.764 0.840 0.874 0.881 0.916 0.938 0.961 0.995 1.000
          1.000 1.000 1.000];
stage4 = [0.013 0.059 0.141 0.204 0.264 0.388 0.475 0.560 0.668 0.739 0.790 0.825
          0.836 0.875 0.916 0.918 0.942 0.953 0.965 0.968 1.000 0.987 1.000 1.000
          1.000 1.000 1.000];
stage5 = [0.077 0.115 0.185 0.311 0.482 0.612 0.703 0.782 0.853 0.879 0.904 0.927
          0.942 0.962 0.968 0.981 0.990 1.000 1.000 1.000 1.000 1.000 1.000 1.000
          1.000 1.000 1.000];
stage6 = [0.085 0.254 0.571 0.764 0.882 0.901 0.940 1.000 0.995 1.000 1.000 1.000
          1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
          1.000 1.000 1.000];
%
% Graph plots
subplot (2,1,1)
```

```
plot (cycle1,stage1,'-*','LineWidth',1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
title('H+ fraction in liquid in a 6 stage system [a HIGH-LOW concentration step
change]','FontSize',16)%'FontWeight', 'bold')
xlabel('cycle number [number]','FontSize',16)%,'FontWeight', 'bold')
ylabel('H+ fraction in liquid [meq/l]','FontSize',16)%,'FontWeight', 'bold'))
hold on
%
plot(cycle1,stage2,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage3,'-s','LineWidth',1.5,'MarkerSize',7)
hold on
%
plot(cycle1,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage5,'-p','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
% Graph plots
subplot (2,1,2)
plot (cycle2,stage1,'-*','LineWidth',1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
title('Liquid concentration per stage in a 6 stage system [a HIGH-LOW concentration
step change]','Fontsize',16)%,'FontWeight', 'bold')
xlabel('cycles [min]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Liquid concentration [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle2,stage2,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage3,'-s','LineWidth',1.5,'MarkerSize',7)
hold on
%
```

```
plot(cycle2,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage5,'-p','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
%----------------------------------Program end-------------------------------------------
```

APPENDIX A.3: MATLAB software program for the H^+ fractional change for the 12 stage system moving from a LOW to HIGH concentration step change

```
%-----------------------------------------------------------------------------------------------
% Program used for manupulation of data from a Continuous Countercurrent Ion
Exchange (CCIX) process
% used for desalination of water.
% The program is an H+ LOW to HIGH step change in the feed concentration for a
12 stage system.
%
% Author: NM Dube
% Dated: 16 April 2011
%
% Data vectors for a 12 stage system, T is the upflow period in (min),
% The third run is a LOW to HIGH (H+) step change in the feed concentration
cycle1 = 1:1:23;
T = 15;
cycle2 = T*cycle1;
%
% Stage liquid concentration - corrected data
 stage1 = [0.000 0.254 0.517 0.698 0.726 0.776 0.910 0.849 0.948 0.954 0.960
              0.980 0.986 0.984 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
              1.000];
 stage2 = [0.012 0.013 0.046 0.088 0.238 0.299 0.633 0.725 0.778 0.814 0.821
              0.797 0.946 0.946 0.960 0.965 0.955 0.969 0.973 0.992 0.992 0.984
              1.000];
 stage3 = [0.000 0.024 0.000 0.000 0.008 0.021 0.171 0.283 0.403 0.523 0.673
              0.737 0.841 0.892 0.918 0.934 0.937 0.904 0.980 0.973 0.983 0.995
              1.000];
 stage4 = [0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.001 0.044 0.147 0.210
              0.279 0.532 0.633 0.727 0.805 0.830 0.861 0.879 0.937 0.967 0.960
              1.000];
 stage5 = [0.010 0.000 0.015 0.014 0.000 0.000 0.000 0.000 0.026 0.044 0.061
              0.100 0.144 0.235 0.340 0.506 0.581 0.657 0.800 0.858 0.914 0.975
              1.000];
 stage6 = [0.014 0.020 0.057 0.000 0.000 0.000 0.000 0.011 0.000 0.000 0.014
              0.010 0.006 0.060 0.083 0.140 0.205 0.271 0.374 0.470 0.637 0.814
              1.000];
```

```
% Graph plots
subplot (2,1,1)
plot (cycle1,stage1,'-*','LineWidth', 1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
% Labels for the plot and specifying font size for each
title('H+ fraction in liquid in a 12 stage system [a LOW-HIGH concentration step
change]','fontsize',16)%'FontWeight', 'bold')
xlabel('cycle number [number]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('H+ fraction in liquid [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle1,stage2,'-o','LineWidth', 1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage3,'-s','LineWidth', 1.5,'MarkerSize',7)
hold on
%
plot(cycle1,stage4,'-d','LineWidth', 1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage5,'-p','LineWidth', 1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage6,'-^','LineWidth', 1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
subplot (2,1,2)
plot (cycle2,stage1,'-*','LineWidth', 1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
title('Liquid concentration per stage in a 12 stage system [a LOW-HIGH concentration
step change]','Fontsize',16)%,'FontWeight', 'bold')
xlabel('cycles [min]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Liquid concentration [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle2,stage2,'-o','LineWidth', 1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
```

```
plot(cycle2,stage3,'-s','LineWidth', 1.5,'MarkerSize',7)
hold on
%
plot(cycle2,stage4,'-d','LineWidth', 1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage5,'-p','LineWidth', 1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage6,'-^','LineWidth', 1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
%----------------------------------Program end-------------------------------------------
```

APPENDIX A.4: MATLAB software program for the H^+ fractional change for the 12 stage system moving from HIGH to LOW concentration step change

```
%-----------------------------------------------------------------------------------------------
% Program used for manupulation of data from Continuous Countercurrent Ion
% Exchange (CCIX) process used for desalination of water.
% The program is an H+ HIGH to LOW step change in the feed concentration for a
12 stage system.
%
% Author: NM Dube
% Dated: 16 April 2011
%
% Data vectors for a 12 stage system, T is the upflow period in (min),
% The fourth run is a HIGH to LOW (H+) step change in the feed concentration.
cycle1 = 1:1:21;
T = 15;
cycle2 = T*cycle1;
%
% Stage liquid concentration
 stage1 = [0.058 0.066 0.037 0.075 0.037 0.073 0.074 0.089 0.113 0.139 0.146
            0.170 0.201 0.227 0.300 0.332 0.405 0.554 0.648 0.760 0.898];
 stage2 = [0.020 0.009 0.016 0.024 0.026 0.071 0.085 0.164 0.191 0.245 0.322
            0.376 0.448 0.548 0.620 0.596 0.643 0.839 0.894 1.000 1.000];
 stage3 = [0.002 0.033 0.026 0.134 0.092 0.310 0.188 0.380 0.499 0.569 0.674
            0.690 0.756 0.815 0.846 0.887 0.885 0.956 0.973 1.000 1.000];
 stage4 = [0.000 0.101 0.083 0.204 0.337 0.489 0.569 0.764 0.823 0.861 0.902
            0.909 0.953 0.962 0.986 1.000 1.000 1.000 1.000 1.000 1.000];
 stage5 = [0.000 0.000 0.307 0.567 0.641 0.762 0.787 0.875 0.897 0.926 0.949
            0.982 0.972 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000];
 stage6 = [0.000 0.000 0.210 0.540 0.780 0.810 0.780 0.931 1.020 0.991 1.080
            1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000];
%
% Graph plots
%
subplot (2,1,1)
plot (cycle1,stage1,'-*','LineWidth',1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
% Labels for the plot and specifying font size for each
```

```
title('H+ fraction in liquid in a 12 stage system [a HIGH-LOW concentration step
change]','fontsize',16)%'FontWeight', 'bold')
xlabel('cycle number [number]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('H+ fraction in liquid [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle1,stage2,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage3,'-s','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage5,'-p','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
subplot (2,1,2)
plot (cycle2,stage1,'-*','LineWidth',1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
title('Liquid concentration per stage in a 12 stage system [a HIGH-LOW concentration
step change]','Fontsize',16)%,'FontWeight', 'bold')
xlabel('cycles [min]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Liquid concentration [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle2,stage2,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage3,'-s','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%)
plot(cycle2,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
```

```
plot(cycle2,stage5,'-p','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
%-------------------------------------Program end---------------------------------------------
```

APPENDIX A.5: MATLAB software program for the Na^+ fractional change values of the 6 stage system for a LOW to HIGH concentration step change

```
% --------------------------------------------------------------------------------------------------
% Program used for manupulation of data from Continuous Countercurrent Ion
% Exchange (CCIX) process used for desalination of water.
%
% The program is a Na LOW to HIGH step change in the feed concentration for a 6
stage system.
%
% Author: NM Dube
% Dated: 16 April 2011
%
% Data vectors for a 6 stage system, T is the upflow period in (min),
% The first run is a LOW to HIGH (Na) step change in the feed concentration
 cycle1 = 1:1.0:18;
 T = 17;
 cycle2 = T*cycle1;

% Stage liquid concentration
 stage1 = [0.221 0.577 0.730 0.847 0.920 0.936 0.968 0.974 0.981 0.989 0.988
           0.997 1.000 1.000 1.000 1.000 1.000 1.000];
 stage2 = [0.000 0.140 0.314 0.523 0.656 0.766 0.842 0.886 0.900 0.933 0.958
           0.963 0.982 0.974 0.991 0.994 0.993 0.993];
 stage3 = [0.000 0.004 0.066 0.184 0.295 0.454 0.601 0.690 0.758 0.804 0.877
           0.881 0.951 0.965 0.972 0.981 1.000 0.988];
 stage4 = [0.000 0.000 0.004 0.035 0.082 0.168 0.277 0.361 0.440 0.522 0.698
           0.784 0.899 0.931 0.966 0.966 0.991 0.991];
 stage5 = [0.000 0.000 0.000 0.003 0.020 0.052 0.113 0.167 0.207 0.340 0.474
           0.547 0.780 0.860 0.899 0.905 0.975 0.973];
 stage6 = [0.000 0.000 0.000 0.000 0.000 0.001 0.024 0.033 0.063 0.124 0.167
           0.233 0.482 0.539 0.672 0.779 0.940 0.972];
%
% Obtain Na+ fractional change in liquid concentration
stage1 = 1- stage1;
stage2 = 1- stage2;
stage3 = 1- stage3;
stage4 = 1- stage4;
```

```
stage5 = 1- stage5;
stage6 = 1- stage6;
%
% Graph plots
subplot (2,1,1,figure (1))
plot (cycle1,stage1,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
% Labels for the plot and specifying font size for each
title('Na+ fraction in liquid in a 6 stage system [a LOW-HIGH concentration step
change]','fontsize',16)%'FontWeight', 'bold')
xlabel('cycle number [number]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Na+ fraction in liquid [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle1,stage2,'-<','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage3,'-s','LineWidth',1.5,'MarkerSize',7)
hold on
%
plot(cycle1,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage5,'-p','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
subplot (2,1,2,figure(1))
plot (cycle2,stage1,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
title('Liquid concentration per stage in a 6 stage system [a LOW-HIGH concentration
step change]','Fontsize',16)%,'FontWeight', 'bold')
xlabel('cycles [min]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Liquid concentration [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle2,stage2,'-<','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
```

```
hold on
%
plot(cycle2,stage3,'-s','LineWidth',1.5,'MarkerSize',7)
hold on
%
plot(cycle2,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage5,'-p','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
%----------------------------------Program end-------------------------------------------
```

APPENDIX A.6: MATLAB software program for the Na^+ fractional change values of the 6 stage system for a HIGH to LOW concentration step change

```
%----------------------------------------------------------------------------------------------------
% Program used for manupulation of data from continuous countercurrent ion
% exchange (CCIX) process used for desalination of water.
% The program is a Na HIGH to LOW step change in the feed concentration for a 6
stage system.
%
% Author: NM Dube
% Dated: 16 April 2011
%
% Data vectors for a 6 stage system, T is the upflow period in (min),
% The second run is a HIGH to LOW (Na) step change in the feed concentration
cycle1 = 1:1:28;
T = 15;
cycle2 = T*cycle1;

%
% Corrected values that were dropping from the norm using (interpolation) averaging
method
 stage1 = [0.014 0.002 0.031 0.036 0.063 0.062 0.088 0.119 0.152 0.183 0.209
           0.240 0.280 0.332 0.372 0.413 0.449 0.525 0.591 0.634 0.670 0.694 0.775
           0.812 0.836 0.903 0.931];
 stage2 = [0.009 0.003 0.017 0.052 0.059 0.085 0.117 0.152 0.191 0.250 0.295
           0.350 0.490 0.476 0.538 0.577 0.645 0.664 0.752 0.780 0.809 0.829 0.868
           0.920 0.939 0.959 1.000];
 stage3 = [0.014 0.021 0.057 0.079 0.126 0.170 0.235 0.312 0.403 0.471 0.591
           0.631 0.693 0.716 0.756 0.764 0.840 0.874 0.881 0.916 0.938 0.961 0.995
           1.000 1.000 1.000 1.000];
 stage4 = [0.013 0.059 0.141 0.204 0.264 0.388 0.475 0.560 0.668 0.739 0.790
           0.825 0.836 0.875 0.916 0.918 0.942 0.953 0.965 0.968 1.000 0.987 1.000
           1.000 1.000 1.000 1.000];
 stage5 = [0.077 0.115 0.185 0.311 0.482 0.612 0.703 0.782 0.853 0.879 0.904
           0.927 0.942 0.962 0.968 0.981 0.990 1.000 1.000 1.000 1.000 1.000 1.000
           1.000 1.000 1.000 1.000];
 stage6 = [0.085 0.254 0.571 0.764 0.882 0.901 0.940 1.000 0.995 1.000 1.000
               1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
               1.000 1.000 1.000 1.000];
%
% Obtain Na+ fractional change in liquid concentration
stage1 = 1- stage1;
stage2 = 1- stage2;
stage3 = 1- stage3;
stage4 = 1- stage4;
stage5 = 1- stage5;
stage6 = 1- stage6;
%
% Graph plots
subplot (2,1,1)
plot(cycle1,stage1,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
title('Na+ fraction in liquid in a 6 stage system [a HIGH-LOW concentration step
        change]','fontsize',16)%'FontWeight', 'bold')
xlabel('cycle number [number]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Na+ fraction in liquid [meq/l]','Fontsize',16)%,'FontWeight', 'bold'))
hold on
%
plot(cycle1,stage2,'-<','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage3,'-s','LineWidth',1.5,'MarkerSize',7)
hold on
%
plot(cycle1,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage5,'-*','LineWidth',1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
% Graph plots
subplot (2,1,2)
```

```
plot(cycle2,stage1,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
title('Liquid concentration (Na) per stage in a 6 stage system [a HIGH-LOW
        concentration step change]','Fontsize',16)%,'FontWeight', 'bold')
xlabel('cycles [min]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Liquid concentration [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle2,stage2,'-<','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage3,'-s','LineWidth',1.5,'MarkerSize',7)
hold on
%
plot(cycle2,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage5,'-*','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
%---------------------------------Program end------------------------------------------
```

APPENDIX A.7: MATLAB software program for the Na^+ fractional change values of the 12 stage system for a LOW to HIGH concentration step change

```
% ---------------------------------------------------------------------------------------------------
% Program used for manupulation of data from a continuous countercurrent ion
% exchange (CCIX) process used for desalination of water.
% The program is a Na LOW to HIGH step change in the feed concentration for a 12
stage system.
%
% Author: NM Dube
% Dated: 16 April 2011
%
% Data vectors for a 12 stage system, T is the upflow period in (min),
% The third run is a LOW to HIGH (Na) step change in the feed concentration
 cycle1 = 1:1:23;
 T = 15;
 cycle2 = T*cycle1;
%
%
% Stage liquid concentration - corrected data
stage1 = [0.000 0.254 0.517 0.698 0.726 0.776 0.910 0.849 0.948 0.954 0.960 0.980
          0.986 0.984 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000];
stage2 = [0.012 0.013 0.046 0.088 0.238 0.299 0.633 0.725 0.778 0.814 0.821 0.797
          0.946 0.946 0.960 0.965 0.955 0.969 0.973 0.992 0.992 0.984 1.000];
stage3 = [0.000 0.024 0.000 0.000 0.008 0.021 0.171 0.283 0.403 0.523 0.673 0.737
          0.841 0.892 0.918 0.934 0.937 0.904 0.980 0.973 0.983 0.995 1.000];
stage4 = [0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.001 0.044 0.147 0.210 0.279
          0.532 0.633 0.727 0.805 0.830 0.861 0.879 0.937 0.967 0.960 1.000];
stage5 = [0.010 0.000 0.015 0.014 0.000 0.000 0.000 0.000 0.026 0.044 0.061 0.100
          0.144 0.235 0.340 0.506 0.581 0.657 0.800 0.858 0.914 0.975 1.000];
stage6 = [0.014 0.020 0.057 0.000 0.000 0.000 0.000 0.011 0.000 0.000 0.014 0.010
          0.006 0.060 0.083 0.140 0.205 0.271 0.374 0.470 0.637 0.814 1.000];
%
% Obtain Na+ fractional change in liquid concentration
stage1 = 1- stage1;
stage2 = 1- stage2;
stage3 = 1- stage3;
stage4 = 1- stage4;
```

```
stage5 = 1- stage5;
stage6 = 1- stage6;
%
% Graph plots
subplot (2,1,1)
plot (cycle1,stage1,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
% Labels for the plot and specifying font size for each
title('Na+ fraction in liquid in a 12 stage system [a LOW-HIGH concentration step
change]','fontsize',16)%'FontWeight', 'bold')
xlabel('cycle number [number]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Na+ fraction in liquid [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle1,stage2,'-<','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage3,'-s','LineWidth',1.5,'MarkerSize',7)
hold on
%
plot(cycle1,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage5,'-*','LineWidth',1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
subplot (2,1,2)
plot (cycle2,stage1,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
title('Liquid concentration (Na) per stage in a 12 stage system [a LOW-HIGH
    concentration step change]','Fontsize',16)%,'FontWeight', 'bold')
xlabel('cycles [min]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Liquid concentration [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle2,stage2,'-<','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
```

```
hold on
%
plot(cycle2,stage3,'-s','LineWidth',1.5,'MarkerSize',7)
hold on
%
plot(cycle2,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage5,'-*','LineWidth', 1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage6,'-^','LineWidth', 1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
%----------------------------------Program end-------------------------------------------
```

APPENDIX A.8: MATLAB software program for the Na^+ fractional change values for the 12 stage system moving from high to low concentration step change

```
% -----------------------------------------------------------------------------------------------
% Program used for manupulation of data from continuous countercurrent ion
    exchange (CCIX) process used for desalination of water.
% The program is a Na HIGH to LOW step change in the feed concentration for a 12
   stage system.
%
% Author: NM Dube
% Dated: 16 April 2011
%
% Data vectors for a 12 stage system, T is the upflow period in (min),
% The fourth run is a HIGH to LOW (Na) step change in the feed concentration
 cycle1 = 1:1:21;
 T = 15;
 cycle2 = T*cycle1;
%
% Stage liquid concentration
 stage1 = [0.058 0.066 0.037 0.075 0.037 0.073 0.074 0.089 0.113 0.139 0.146
              0.170 0.201 0.227 0.300 0.332 0.405 0.554 0.648 0.760 0.898];
 stage2 = [0.020 0.009 0.016 0.024 0.026 0.071 0.085 0.164 0.191 0.245 0.322
              0.376 0.448 0.548 0.620 0.596 0.643 0.839 0.894 1.000 1.000];
 stage3 = [0.002 0.033 0.026 0.134 0.092 0.310 0.188 0.380 0.499 0.569 0.674
              0.690 0.756 0.815 0.846 0.887 0.885 0.956 0.973 1.000 1.000];
 stage4 = [0.000 0.101 0.083 0.204 0.337 0.489 0.569 0.764 0.823 0.861 0.902
              0.909 0.953 0.962 0.986 1.000 1.000 1.000 1.000 1.000 1.000];
 stage5 = [0.000 0.000 0.307 0.567 0.641 0.762 0.787 0.875 0.897 0.926 0.949
              0.982 0.972 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000];
 stage6 = [0.000 0.000 0.210 0.540 0.780 0.810 0.780 0.931 1.020 0.991 1.080
              1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000];
%
% Obtain Na+ fractional change in liquid concentration
stage1 = 1- stage1;
stage2 = 1- stage2;
stage3 = 1- stage3;
stage4 = 1- stage4;
stage5 = 1- stage5;
```

```
stage6 = 1- stage6;
% Graph plots
%
subplot (2,1,1)
plot (cycle1,stage1,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
% Labels for the plot and specifying font size for each
title('Na+ fraction in liquid in a 12 stage system [a HIGH-LOW concentration step
change]','fontsize',16)%'FontWeight', 'bold')
xlabel('cycle number [number]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Na+ fraction in liquid [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle1,stage2,'-<','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage3,'-s','LineWidth',1.5,'MarkerSize',7)
hold on
%
plot(cycle1,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage5,'-*','LineWidth',1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle1,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
subplot (2,1,2)
plot (cycle2,stage1,'-o','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
grid on
title('Liquid concentration (Na) per stage in a 12 stage system [a HIGH-LOW
concentration step change]','Fontsize',16)%,'FontWeight', 'bold')
xlabel('cycles [min]','Fontsize',16)%,'FontWeight', 'bold')
ylabel('Liquid concentration [meq/l]','Fontsize',16)%,'FontWeight', 'bold')
hold on
%
plot(cycle2,stage2,'-<','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
```

```
%
plot(cycle2,stage3,'-s','LineWidth',1.5,'MarkerSize',7)
hold on
%)
plot(cycle2,stage4,'-d','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage5,'-*','LineWidth',1.5,'MarkerSize',8,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
%
plot(cycle2,stage6,'-^','LineWidth',1.5,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
%
%----------------------------------Program end------------------------------------------
```

APPENDIX B: PROGRAMS FOR PARAMETER ESTIMATION OF THE BILINEAR AFFINE MODEL

APPENDIX B.1: Program for parameter estimation using the measured data of the process output

```
% -----------------------------------------------------------------------------------------------
% Program for solution of parameter estimation for a CCIX (continuous
% countercurrent ion exchange) process,
% CCIX in this application is used for water desalination.
% A model has been developed for rea- time optimal control and therefore,
% parameter estimation must be performed.
% The model is based on the six stages UCT (University of Cape Town) model,
% and the current project has 6 stages.
% In this program z is the output variable and z_bar is the measured output.
% The program estimates parameters using an output-based method.
% ----------------------------------------------------------------------------------------------
% Author: NM Dube
% Dated: 23 June 2011
% Updated: 21 February 2016
%
%---------------------------------------------------------------------------------------------
% Initializing of the model equation from available data obtained from UCT project
%
  N = 6;                    % number of stages of the process
  h = 32.93;                  % resin holdups (units in litres)
  H = 42.809;                 % liquid holdups (units in litres)
  Fl = 2000/60;               % liquid flow rate (units in litres/min)
%
  conc_in = 0.435;            % sodium concentration at startup of process (eq/l)
% conc_in = 0.055
% conc_in = 0.110
% conc_in = 0.435
% conc_in = 0.550
% conc_in = 0.754
% conc_in = 1.000
  T = 17;                     % liquid upflow time (units in min)
%
%------------------------------------------------------------------------------------
% Initializing variables used in the model formulas with data from UCT project
% %
```

```
  a = 230*[0.8 0.6 0.5 0.4 0.3 0.25];        % (try this for better results)
  b = 60*[0.01 0.02 0.05 0.08 0.1 0.2];      % ionic liquid fraction in the solution
%
%
  d = 2/3;                                   % resin to liquid fraction relationship
  FR = (h*d)/T;                              % initial control variable, resin flowrate,
%                                                 %(units in litres/hour)
%------------------------------------------------------------------------------------
% Declaration of parameters from the model equations as determined by,
% dx/dt = Ax(t)+B1x(t)u(t)+Bu(t)+W(t)w(t) and
% determinaton of parameters are l(i), m(i), m(i(j)) and k(i)
%
  for i = 1:N
      l(i) = Fl/(H+(a(i)*h));
      mi(i) = a(i)/(H+(a(i)*h));
  end
%
  for i = 1:N-1
      k(i) = (b(i+1) - b(i))/(H+a(i)*h);
      mij(i) = a(i+1)/(H+(a(i)*h));
  end
     k(N) = -b(N)/(H+a(N)*h);
%------------------------------------------------------------------------------------
% Initialize unknown parameters using the declared parameters above,
% there will be (N*4) – 1 parameters the parameter mij(6) is thus equated to 0.
%
  p(1) = l(1);
  p(2) = mij(1);
  p(3) = mi(1);
  p(4) = k(1);
  p(5) = l(2);
  p(6) = mij(2);
  p(7) = mi(2);
  p(8) = k(2);
  p(9) = l(3);
  p(10) = mij(3);
  p(11) = mi(3);
  p(12) = k(3);
  p(13) = l(4);
  p(14) = mij(4);
  p(15) = mi(4);
  p(16) = k(4);
  p(17) = l(5);
  p(18) = mij(5);
  p(19) = mi(5);
  p(20) = k(5);
  p(21) = l(6);
  p(22) = mi(6);
  p(23) = k(6);
%
%-----------------------------------------------------------------------------------------------
% Measured data
% Stage H+ fractional change in liquid concentration using data obtained from UCT
Project volume 4
% stage1 = [0.221 0.577 0.730 0.847 0.920 0.936 0.968 0.974 0.981 0.989 0.988
             0.997 1.000 1.000 1.000 1.000 1.000 1.000];
% stage2 = [0.000 0.140 0.314 0.523 0.656 0.766 0.842 0.886 0.900 0.933 0.958
             0.963 0.982 0.974 0.991 0.994 0.993 0.993];
% stage3 = [0.000 0.04 0.066 0.184 0.295 0.454 0.601 0.690 0.758 0.804 0.877
             0.881 0.951 0.965 0.972 0.981 1.000 0.988];
% stage4 = [0.000 0.000 0.004 0.035 0.082 0.168 0.277 0.361 0.440 0.522 0.698
             0.784 0.899 0.931 0.966 0.966 0.991 0.991];
% stage5 = [0.000 0.000 0.000 0.003 0.020 0.052 0.113 0.167 0.207 0.340 0.474
             0.547 0.780 0.860 0.899 0.905 0.975 0.973];
% stage6 = [0.000 0.000 0.000 0.000 0.000 0.001 0.024 0.033 0.063 0.124 0.167
             0.233 0.482 0.539 0.672 0.779 0.940 0.972];
%
% Obtain Na+ fractional change in liquid concentration
% stage1 = 1- stage1;
% stage2 = 1- stage2;
% stage3 = 1- stage3;
% stage4 = 1- stage4;
% stage5 = 1- stage5;
% stage6 = 1- stage6;
%-----------------------------------------------------------------------------------------------
% Introducing measurement vector expanded for the full state trajectory
```

```
states_bar = [1.0000  1.0000  1.0000  1.0000  1.0000  1.0000  1.0000  1.0000  1.0000
              1.0000...
0.7800 0.7500 0.7180 0.6870 0.6550 0.6220 0.5900 0.5580 0.5270 0.4930...
0.4620 0.4300 0.4120 0.3980 0.3850 0.3720 0.3570 0.3440 0.3290 0.3160...
0.3030 0.2870 0.2750 0.2640 0.2530 0.2420 0.2320 0.2210 0.2100 0.2000...
0.1885 0.1780 0.1680 0.1570 0.1490 0.1430 0.1360 0.1290 0.1240 0.1170...
0.1100 0.1040 0.0970 0.0900 0.0840 0.0790 0.0780 0.0760 0.0750 0.0740...
0.0730 0.0710 0.0685 0.0670 0.0660 0.0650 0.0640 0.0600 0.0580 0.0540...
0.0520 0.0480 0.0460 0.0440 0.0400 0.0370 0.0340 0.0330 0.0320 0.0310...
0.0300 0.0290 0.0285 0.0280 0.0275 0.0270 0.0265 0.0260 0.0255 0.0250...
0.0245 0.0240 0.0235 0.0230 0.0225 0.0220 0.0210 0.0200 0.0190 0.0185...
0.0180 0.0170 0.0165 0.0160 0.0155 0.0150 0.0145 0.0140 0.0130 0.0120...
0.0110 0.0110 0.0115 0.0116 0.0117 0.0118 0.0119 0.0120 0.0121 0.0125...
0.0127 0.0130 0.0120 0.0110 0.0095 0.0090 0.0080 0.0070 0.0060 0.0050...
0.0045 0.0040 0.0035 0.0030 0.0025 0.0020 0.0018 0.0016 0.0012 0.0010...
0.0008 0.0005 0.0002 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000...
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000...
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000...
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000...
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000...
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000;
%
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 0.9890 0.9770 0.9640 0.9520 0.9380 0.9260 0.9140 0.9110 0.8880...
0.8750 0.8630 0.8480 0.8320 0.8160 0.8010 0.7850 0.7690 0.7540 0.7380...
0.7230 0.7070 0.6910 0.6740 0.6550 0.6360 0.6170 0.5980 0.5795 0.5610...
0.5420 0.5230 0.5050 0.4850 0.4700 0.4585 0.4460 0.4340 0.4230 0.4100...
0.3985 0.3870 0.3750 0.3630 0.3510 0.3390 0.3300 0.3200 0.3100 0.3000...
0.2900 0.2800 0.2700 0.2600 0.2505 0.2405 0.2320 0.2250 0.2180 0.2110...
0.2050 0.1970 0.1910 0.1840 0.1770 0.1700 0.1640 0.1570 0.1540 0.1490...
0.1450 0.1420 0.1370 0.1340 0.1290 0.1260 0.1220 0.1170 0.1150 0.1140...
0.1120 0.1100 0.1080 0.1070 0.1060 0.1050 0.1040 0.1030 0.1020 0.1000...
0.0970 0.0950 0.0920 0.0880 0.0860 0.0830 0.0790 0.0770 0.0740 0.0710...
0.0670 0.0650 0.0640 0.0610 0.0580 0.0560 0.0540 0.0520 0.0490 0.0470...
0.0450 0.0440 0.0430 0.0420 0.0410 0.0405 0.0400 0.0390 0.0385 0.0380...
0.0375 0.0370 0.0365 0.0360 0.0345 0.0330 0.0315 0.0290 0.0270 0.0260...
0.0240 0.0230 0.0210 0.0180 0.0170 0.0185 0.0200 0.0210 0.0220 0.0230...
0.0240 0.0245 0.0248 0.0250 0.0255 0.0255 0.0240 0.0230 0.0210 0.0190...
```

```
0.0180 0.0160 0.0150 0.0140 0.0120 0.0100 0.0085 0.0080 0.0075 0.0070...
0.0068 0.0065 0.0063 0.0060 0.0058 0.0055 0.0053 0.0052 0.0053 0.0058...
0.0060 0.0062 0.0063 0.0065 0.0068 0.0070 0.0073 0.0075 0.0078 0.0080...
0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080 0.0080;
%
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 0.9980 0.9940 0.9895 0.9860 0.9830 0.9785 0.9750 0.9720 0.9680...
0.9650 0.9610 0.9580 0.9560 0.9540 0.9520 0.9485 0.9470 0.9440 0.9430...
0.9390 0.9370 0.9350 0.9270 0.9160 0.9055 0.8950 0.8850 0.8740 0.8640...
0.8530 0.8430 0.8320 0.8210 0.8110 0.8010 0.7905 0.7805 0.7705 0.7605...
0.7505 0.7405 0.7305 0.7205 0.7105 0.6985 0.6850 0.6700 0.6560 0.6420...
0.6270 0.6130 0.5980 0.5840 0.5700 0.5560 0.5420 0.5280 0.5150 0.5020...
0.4880 0.4750 0.4620 0.4490 0.4350 0.4230 0.4090 0.3970 0.3890 0.3810...
0.3740 0.3650 0.3570 0.3490 0.3410 0.3330 0.3250 0.3170 0.3090 0.3040...
0.2970 0.2910 0.2850 0.2780 0.2740 0.2670 0.2600 0.2550 0.2480 0.2430...
0.2380 0.2340 0.2295 0.2255 0.2220 0.2170 0.2140 0.2090 0.2050 0.2010...
0.1970 0.1900 0.1840 0.1770 0.1710 0.1640 0.1580 0.1510 0.1445 0.1375...
0.1310 0.1250 0.1240 0.1235 0.1230 0.1225 0.1220 0.1215 0.1210 0.1205...
0.1200 0.1190 0.1180 0.1150 0.1085 0.1030 0.0955 0.0895 0.0840 0.0770...
0.0710 0.0650 0.0580 0.0520 0.0480 0.0470 0.0460 0.0450 0.0440 0.0430...
0.0420 0.0390 0.0380 0.0370 0.0355 0.0350 0.0345 0.0340 0.0335 0.0330...
0.0320 0.0310 0.0305 0.0295 0.0285 0.0280 0.0275 0.0270 0.0260 0.0255...
0.0250 0.0245 0.0240 0.0230 0.0220 0.0210 0.0195 0.0180 0.0170 0.0150...
0.0140 0.0130 0.0100 0.0080 0.0070 0.0050 0.0030 0.0010 0.0000 0.0000...
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000;
%
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 1.0000 0.9998 0.9995 0.9994 0.9992 0.9990 0.9988 0.9985 0.9980...
0.9978 0.9975 0.9970 0.9950 0.9920 0.9880 0.9860 0.9840 0.9800 0.9770...
0.9750 0.9730 0.9685 0.9660 0.9640 0.9580 0.9545 0.9500 0.9460 0.9420...
0.9370 0.9340 0.9285 0.9250 0.9205 0.9145 0.9060 0.8985 0.8920 0.8840...
0.8760 0.8680 0.8600 0.8540 0.8450 0.8370 0.8290 0.8190 0.8090 0.7990...
0.7895 0.7795 0.7700 0.7600 0.7500 0.7405 0.7305 0.7220 0.7140 0.7060...
0.6985 0.6902 0.6840 0.6760 0.6680 0.6610 0.6540 0.6460 0.6380 0.6315...
0.6240 0.6170 0.6095 0.6030 0.5955 0.5890 0.5820 0.5750 0.5680 0.5600...
0.5530 0.5450 0.5380 0.5305 0.5230 0.5150 0.5090 0.5010 0.4940 0.4870...
0.4790 0.4645 0.4480 0.4320 0.4160 0.4000 0.3850 0.3690 0.3530 0.3370...
```

```
0.3210 0.3050 0.2960 0.2880 0.2805 0.2703 0.2650 0.2570 0.2490 0.2420...
0.2340 0.2260 0.2180 0.2090 0.1985 0.1880 0.1780 0.1680 0.1570 0.1470...
0.1370 0.1270 0.1160 0.1060 0.0990 0.0970 0.0940 0.0910 0.0880 0.0850...
0.0830 0.0790 0.0770 0.0740 0.0710 0.0680 0.0650 0.0620 0.0580 0.0550...
0.0530 0.0480 0.0460 0.0430 0.0390 0.0360 0.0340 0.0340 0.0340 0.0340...
0.0340 0.0340 0.0340 0.0340 0.0340 0.0340 0.0340 0.0330 0.0320 0.0290...
0.0270 0.0250 0.0230 0.0200 0.0175 0.0160 0.0140 0.0120 0.0080 0.0080...
0.0080 0.0080 0.0080 0.0080 0.0000 0.0080 0.0000 0.0080 0.0080 0.0080;
%
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 1.0000 1.0000 0.9999 0.9998 0.9995 0.9993 0.9991 0.9990 0.9988...
0.9985 0.9983 0.9981 0.9980 0.9970 0.9950 0.9940 0.9930 0.9900 0.9880...
0.9870 0.9860 0.9840 0.9830 0.9820 0.9780 0.9760 0.9740 0.9700 0.9670...
0.9645 0.9620 0.9580 0.9560 0.9540 0.9500 0.9460 0.9410 0.9350 0.9300...
0.9240 0.9185 0.9140 0.9070 0.9030 0.8970 0.8920 0.8860 0.8820 0.8770...
0.8720 0.8670 0.8625 0.8570 0.8520 0.8475 0.8420 0.8380 0.8330 0.8280...
0.8260 0.8230 0.8180 0.8140 0.8120 0.8070 0.8045 0.8000 0.7970 0.7940...
0.7815 0.7690 0.7570 0.7450 0.7330 0.7220 0.7090 0.6970 0.6855 0.6740...
0.6620 0.6490 0.6380 0.6250 0.6130 0.6020 0.5885 0.5770 0.5650 0.5530...
0.5410 0.5280 0.5210 0.5140 0.5080 0.5020 0.4950 0.4880 0.4820 0.4750...
0.4680 0.4620 0.4550 0.4390 0.4180 0.3970 0.3760 0.3550 0.3340 0.3130...
0.2920 0.2710 0.2500 0.2290 0.2160 0.2085 0.2020 0.1950 0.1870 0.1800...
0.1730 0.1660 0.1580 0.1520 0.1440 0.1380 0.1350 0.1320 0.1280 0.1245...
0.1215 0.1175 0.1145 0.1105 0.1070 0.1040 0.1020 0.1010 0.0995 0.0990...
0.0980 0.0975 0.0970 0.0965 0.0960 0.0955 0.0950 0.0940 0.0875 0.0820...
0.0750 0.0680 0.0630 0.0560 0.0490 0.0430 0.0370 0.0310 0.0250 0.0250...
0.0250 0.0250 0.0250 0.0250 0.0250 0.0250 0.0250 0.0250 0.0250 0.0250;
%
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000...
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0.9999 0.9998 0.9997 0.9997...
0.9996 0.9995 0.9993 0.9990 0.9987 0.9985 0.9980 0.9960 0.9945 0.9930...
0.9900 0.9875 0.9860 0.9845 0.9825 0.9798 0.9775 0.9760 0.9750 0.9745...
```

```
0.9740 0.9735 0.9725 0.9720 0.9700 0.9690 0.9680 0.9670 0.9660 0.9645...
0.9620 0.9580 0.9560 0.9540 0.9510 0.9470 0.9450 0.9430 0.9395 0.9370...
0.9320 0.9260 0.9210 0.9155 0.9090 0.9040 0.8980 0.8940 0.8880 0.8830...
0.8770 0.8730 0.8680 0.8650 0.8615 0.8570 0.8530 0.8490 0.8455 0.8420...
0.8370 0.8340 0.8280 0.8230 0.8165 0.8105 0.8050 0.7980 0.7930 0.7865...
0.7820 0.7750 0.7685 0.7520 0.7295 0.7070 0.6850 0.6630 0.6400 0.6170...
0.5950 0.5730 0.5500 0.5270 0.5150 0.5100 0.5050 0.4995 0.4950 0.4895...
0.4845 0.4790 0.4745 0.4690 0.4640 0.4555 0.4440 0.4320 0.4195 0.4075...
0.3960 0.3840 0.3720 0.3595 0.3480 0.3360 0.3250 0.3155 0.3060 0.2965...
0.2860 0.2770 0.2670 0.2570 0.2480 0.2380 0.2285 0.2175 0.2030 0.1880...
0.1745 0.1600 0.1455 0.1310 0.1170 0.1020 0.0870 0.0730 0.0595 0.0554...
0.0520 0.0480 0.0430 0.0355 0.0305 0.0275 0.0210 0.0160 0.0123 0.0105];
%
%----------------------------------------------------------------------------------
% Initializing the state vector (x0), initial state at the input  and the full trajectory
  x0 = states_bar(:,1);           % declaring initial state variables based on measured
                                    states
  states = states_bar;            % declaring current state variables based on measured
                                  states
  z_bar = states_bar(6,:);    % declaring measured output from measured data
%
%-----------------------------------------------------------------------------------------------
% Initializing constants required for the gradient method of minimizing the error
between
% the output of the experimental data and model output
%
  K = 200;                 % number of steps in the optimization trajectory based on
                              sampling period,
M = 1000;             % total number of iterations for stopping the iteration procedure,
%
  dt = 0.012;                  % realistic sampling period used with supplied data
%
  grad = 2.0;                 % step of the gradient procedure,
%
  delta = 0.001;           % small increment of parameter values as being tested for
                             optimal values,
  epsilon = 0.0001;     % error for stopping the iteration,
%
```

```
 r = 23;                    % total number of parameters since N = 6.
 %-------------------------------------------------------------------------------------------
% Initialize disturbance and control vectors full trajectories
 p =
     [p(1);p(2);p(3);p(4);p(5);p(6);p(7);p(8);p(9);p(10);p(11);p(12);p(13);p(14);p(15);p(1
     6);p(17);p(18);p(19);p(20);p(21);p(22);p(23)];
 p_initial = p
 FR = FR*ones(1,K);
 states_in = conc_in*ones(1,K+1);
%
%--------------------------------------------------------------------------------------------
% Iteration procedure start and initializing index of iteration (j)
  j = 1;
  e0 = cputime;
  while j <= M
    % Solving the state model equation starting with initial values of parameters
    x0 = states(:,1);
    for k = 1:K-1             % at the end of this iteration states should be xER^(NxK+1)
        FRk = FR(k);
        statesk = states(:,k);
        statesk_in =states_in(k);
        states(:,k+1) = iestate(statesk,FRk,statesk_in,x0,p,dt);
    end
    %
    z = states(6,:);
    %-----------------------------------------------------------------------------------------
    % calculating the error between measured and calculated output trajectories
    error = z_bar - z;
    % calculate the least squares differences
    Jp = 0;
    for k = 1:K
        Jp = Jp + sum(error(:,k).*error(:,k));
    end
    %
    %------------------------------------------------------------------------------------
    % calculation of the deviated value of the criterion and
    % calculation of the deviated value of the parameters
    for i = 1:r
        p(i) = p(i)+ delta;
        % calculation of the value of the output with the deviated parameter
        x0 = states(:,1);
        for k = 1:K-1
            FRk = FR(k);
            statesk = states(:,k);
            statesk_in = states_in(k);
            states(:,k+1) =  iestate(statesk,FRk,statesk_in,x0,p,dt);
        end
        %
        z = states(6,:);
        %-------------------------------------------------------------------------------------
        % calculation of the deviated value of the criterion dJp
        error = z_bar - z;
         dJp = 0;
        for k = 1:K
            dJp  =  dJp  +  sum(error(:,k).*error(:,k));                  %  element  by  element
                                                                            multiplication?
        end
        %------------------------------------------------------------------------------------------
        % calculation of the gradient of the criterion
        grad_dJp = (dJp - Jp)/delta;
        % form a vector of the criterion gradients for all parameters
        grad_dJpv(i) = grad_dJp;
        % calculate the old value of the parameters
        p(i) = p(i) - delta;
    end
    %
    %-----------------------------------------------------------------------------------------------
    % calculation of the gradient for the fastest descent method and
    % calculation of the weighted sums
     S = grad_dJpv*grad_dJpv';
     S = sqrt(S);
    %---------------------------------------------------------------------------------------------
    % calculation of the direction of the gradient, dpER^r = dpER^23
     dp = -(grad*grad_dJpv)'/S;
    % check if the achieved solution is an optimal one using the norm
    % calculation of the norm
```

```
      dnorm = norm(dp);
    % check termination of the calculation
    if dnorm <= epsilon
         x0 = states(:,1);
        for k = 1:K-1
            FRk = FR(k);
            statesk = states(:,k);
            statesk_in = states_in(k);
            states(:,k+1) = iestate(statesk,FRk,statesk_in,x0,p,dt);
        end
    break
    else
        % calculate the improved estimate of the parameters
        p = p + dp;
        grad = grad/1.009;
        j = j+1;
    end

end
%-------------------------------------------------------------------------------------------
% Display optimal parameters calculated
Jk1 = 0;
for k = 1:K
    error(k) = states_bar(6,k)-states(6,k);
    Jk(k) = (error*error');
    Jk1 = Jk1 + Jk(k);
end
e1 = cputime;
% for k = 1:K-1
%     Jkx(k) = Jk(k)+Jk(k+1);
% end
 error = abs(error);
 z = states(6,:);
 %j
 p_final = p;
 Jk1
 e = e1 - e0
%  a1 = (Fl-p(1)*H)/p(1)*h
%  b6 = -b(N)

%-----------------------------------------------------------------------------------------------
% Plot the output trajectories
%
figure(1)
k = 1:K;
plot(k,states_bar(6,:),'Linewidth',2.5)
hold on
plot(k,states(6,:),':','Linewidth',2.5)
hold on
plot(error,'o','MarkerSize',7,'Linewidth',2.5)
text('Position',[20.0 0.93],'String','x_6','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[50.0 0.88],'String','x^~_6','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[40.0 0.10],'String','e_x_6','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
xlabel('Discrete time [k]','Fontsize',16)
ylabel('Process & estimated output & error difference [eq/l]','Fontsize',16)
title('Dynamic behaviour of the output of the process','Fontsize',16)
grid on
hold off
%
%----------------------------------------------------------------------------------------------------
figure(2)
k = 1:2:K;
plot(error,'o','MarkerSize',7,'Linewidth',2.5)
text('Position',[80.0 0.05],'String','e_x_6','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
xlabel('Discrete time[k]','Fontsize',16)
ylabel('Error difference between outputs [eq/l]','Fontsize',16)
title('Error difference between measured and estimated outputs','Fontsize',16)
grid on
%
%------------------------------------------------------------------------------------------
figure(3)
k = 1:K;
```

```
subplot(3,2,1)
plot(k,states(1,:),':')
hold on
plot(k,states_bar(1,:))
text('Position',[30.0 0.13],'String','x1bar','Fontsize',12)
text('Position',[60.0 0.20],'String','x^~_1','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('True & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
subplot(3,2,2)
plot(k,states(2,:),':')
hold on
plot(k,states_bar(2,:))
text('Position',[60.0 0.15],'String','x2bar','Fontsize',12)
text('Position',[30.0 0.90],'String','x^~_2','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('True & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
subplot(3,2,3)
plot(k,states(3,:),':')
hold on
plot(k,states_bar(3,:))
text('Position',[60.0 0.25],'String','x3bar','Fontsize',12)
text('Position',[40.0 0.10],'String','x^~_3','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('True & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
subplot(3,2,4)
plot(k,states(4,:),':')
hold on
plot(k,states_bar(4,:))
text('Position',[80.0 0.15],'String','x4bar','Fontsize',12)
text('Position',[50.0 0.75],'String','x^~_4','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('True & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
subplot(3,2,5)
plot(k,states(5,:),':')
hold on
plot(k,states_bar(5,:))
text('Position',[60.0 0.70],'String','x5bar','Fontsize',12)
text('Position',[40.0 0.50],'String','x^~_5','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('True & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
subplot(3,2,6)
plot(k,z_bar)
hold on
plot(k,z,':')
text('Position',[20.0 1.35],'String','x^~_6','Fontsize',12)
text('Position',[60.0 0.95],'String','x6bar','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('True & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
%-------------------------------------Program-end------------------------------------------

% ---------------------------------------------------------------------------------------------
% The function is used to calculate the values of the states for parameter estimation,
% the considered trajectory period is (K = 100).
% The function is called in the main program (see details below) under the function
name iestate()
% The main program is called "Parameter_Estimation6a_Output.m"
%
% Author: NM Dube
% Dated: 23 June 2011
% Updated: 21 February 2016
%
%-----------------------------------------------------------------------------------------------
% declare the function iestate()
function states = iestate(statesk,FRk,statesk_in,x0,p,dt)
%
% solving the model state equation using the current parameter values
x1 = statesk(1) + dt*(p(1)*statesk_in - p(1)*statesk(1) + p(2)*FRk*statesk(2) -
        p(3)*FRk*statesk(1) + p(4)*FRk);
x2 = statesk(2) + dt*(p(5)*statesk(1) - p(5)*statesk(2) + p(6)*FRk*statesk(3) -
        p(7)*FRk*statesk(2) + p(8)*FRk);
x3 = statesk(3) + dt*(p(9)*statesk(2) - p(9)*statesk(3) + p(10)*FRk*statesk(4) -
        p(11)*FRk*statesk(3) + p(12)*FRk);
x4 = statesk(4) + dt*(p(13)*statesk(3) - p(13)*statesk(4) + p(14)*FRk*statesk(5) -
        p(15)*FRk*statesk(4) + p(16)*FRk);
x5 = statesk(5) + dt*(p(17)*statesk(4) - p(17)*statesk(5) + p(18)*FRk*statesk(6) -
        p(19)*FRk*statesk(5) + p(20)*FRk);
x6 = statesk(6) + dt*(p(21)*statesk(5) - p(21)*statesk(6) + 0 - p(22)*FRk*statesk(6) +
        p(23)*FRk);
%---------------------------------------------------------------------------------
% declaring the state variables for the state vector at k+1, k=1:K
states = [x1 x2 x3 x4 x5 x6]';
%
%---------------------------------------Function---Ends-----------------------------------------
```

APPENDIX B.2: Program for parameter estimation using the state vector measurements (the least squares method)

```
% -----------------------------------------------------------------------------------------------------
% Program for solution of parameter estimation for a CCIX(continuous
countercurrent
% ion exchange) process, CCIX in this application is used for water desalination.
% A model has been developed for real-time optimal control and therefore,
% parameter estimation must be performed.
% The model is based on the six stages UCT (University of Cape Town) model,
% and the current project has 6 stages.
% The program estimates parameters using an state vector-based method (the least
squares method).
%-------------------------------------------------------------------------------------------------------
% Author: NM Dube
% Dated: 22 August 2011
% Updated: 21 November 2016
%---------------------------------------------------------------------------------------------------------
% Initializing of the model equation from available data obtained from UCT project
%
  N = 6;                    % number of stages of the process
  H = 42.809;               % liquid holdups (units in litres)
  h = 32.93;                % resin holdups (units in litres)
  Fl = 2000/60;             % liquid flow rate (units in litres/min)
  %
  conc_in = 0.435;          % sodium concentration at startup of process (meq/l)
% conc_in = 0.055
% conc_in = 0.110
% conc_in = 0.435
% conc_in = 0.550
% conc_in = 0.754
% conc_in = 1.000
  T = 17;                   % liquid upflow time (units in hours)
%-------------------------------------------------------------------------------------------------
% Initializing variables used in the model formulas with data from UCT project
%
  a = 230*[0.8 0.6 0.5 0.4 0.3 0.02];        % (try this for better results)
  b = 60*[0.01 0.02 0.05 0.08 0.1 0.2];       % ionic liquid fraction in the solution
%
```

```
  d = 2/3;                         % resin to liquid fraction relationship
  FR = (h*d)/T;                        % initial control variable, resin flowrate,
                                                  %(units in litres/hour)
%-----------------------------------------------------------------------------------------------
% Declaration of parameters from the model equations as determined by,
% dx/dt = Ax(t)+B1x(t)u(t)+Bu(t)+W(t)w(t)
% Parameters are l(i), m(i), m(i(j)) and k(i)
%
for i = 1:N
    l(i) = Fl/(H+a(i)*h);
    mi(i) = a(i)/(H+a(i)*h);
end
%
for i = 1:N-1
    k(i) = (b(i+1) - b(i))/(H+a(i)*h);
    mij(i) = a(i+1)/(H+a(i)*h);
end
    k(N) = -b(N)/(H+a(N)*h);

%-------------------------------------------------------------------------------------
% Initialize unknown parameters using the declared parameters above,
% there will be (N*4) – 1 parameters the parameter mij(6) is thus equated to 0.
%
  p(1) = l(1);
  p(2) = mij(1);
  p(3) = mi(1);
  p(4) = k(1);
  p(5) = l(2);
  p(6) = mij(2);
  p(7) = mi(2);
  p(8) = k(2);
  p(9) = l(3);
  p(10) = mij(3);
  p(11) = mi(3);
  p(12) = k(3);
  p(13) = l(4);
  p(14) = mij(4);
  p(15) = mi(4);
  p(16) = k(4);
  p(17) = l(5);
  p(18) = mij(5);
  p(19) = mi(5);
  p(20) = k(5);
  p(21) = l(6);
  p(22) = mi(6);
  p(23) = k(6);
%
%------------------------------------------------------------------------------------------
%
% Display parameters to see how they look like initially, (if required)
%
 p = [p(1) p(2) p(3) p(4) p(5) p(6) p(7) p(8) p(9) p(10) p(11) p(12) p(13) p(14) p(15)
      p(16) p(17) p(18) p(19) p(20) p(21) p(22) p(23)];
 p_initial = p'
%
%----------------------------------------------------------------------------------------------------
%
% Original data in H+ fractional change in liquid concentration using data obtained
from UCT Project volume 4
% stage1 = [0.221 0.577 0.730 0.847 0.920 0.936 0.968 0.974 0.981 0.989 0.988
                  0.997 1.000 1.000 1.000 1.000 1.000 1.000];
% stage2 = [0.000 0.140 0.314 0.523 0.656 0.766 0.842 0.886 0.900 0.933 0.958
                  0.963 0.982 0.974 0.991 0.994 0.993 0.993];
% stage3 = [0.000 0.04 0.066 0.184 0.295 0.454 0.601 0.690 0.758 0.804 0.877
                  0.881 0.951 0.965 0.972 0.981 1.000 0.988];
% stage4 = [0.000 0.000 0.004 0.035 0.082 0.168 0.277 0.361 0.440 0.522 0.698
                  0.784 0.899 0.931 0.966 0.966 0.991 0.991];
% stage5 = [0.000 0.000 0.000 0.003 0.020 0.052 0.113 0.167 0.207 0.340 0.474
                  0.547 0.780 0.860 0.899 0.905 0.975 0.973];
% stage6 = [0.000 0.000 0.000 0.000 0.000 0.001 0.024 0.033 0.063 0.124 0.167
                  0.233 0.482 0.539 0.672 0.779 0.940 0.972];
%
% Obtain Na+ fractional change in liquid concentration
% stage1 = 1- stage1;
% stage2 = 1- stage2;
% stage3 = 1- stage3;
% stage4 = 1- stage4;
% stage5 = 1- stage5;
% stage6 = 1- stage6;
%
%-----------------------------------------------------------------------------------------------
% Introducing measurement vector expanded for the full state trajectory based on
   Na+ data obtained above
% states_bar is the measured states
%
 states_bar = [1.000 1.000 1.000 1.000 0.760 0.750 0.686 0.622 0.558 0.494...
        0.490 0.398 0.371 0.371 0.344 0.316 0.228 0.264 0.243 0.220...
        0.178 0.157 0.143 0.130 0.116 0.104 0.090 0.078 0.077 0.074...
        0.071 0.068 0.065 0.060 0.055 0.048 0.043 0.037 0.033 0.032...
        0.029 0.028 0.027 0.026 0.025 0.024 0.023 0.022 0.020 0.018...
        0.017 0.016 0.015 0.014 0.013 0.012 0.012 0.012 0.012 0.012...
        0.012 0.012 0.008 0.007 0.005 0.004 0.003 0.002 0.002 0.002...
        0.001 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000...
        0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000...
        0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000;
    %
        1.000 1.000 1.000 1.000 1.000 0.988 0.964 0.938 0.914 0.887...
        0.863 0.832 0.800 0.770 0.738 0.707 0.674 0.635 0.598 0.561...
        0.524 0.485 0.458 0.435 0.410 0.386 0.364 0.340 0.320 0.300...
        0.280 0.261 0.241 0.225 0.212 0.197 0.184 0.170 0.157 0.149...
        0.142 0.134 0.126 0.117 0.114 0.110 0.107 0.105 0.103 0.100...
        0.094 0.088 0.083 0.076 0.070 0.065 0.061 0.056 0.053 0.047...
        0.043 0.042 0.041 0.039 0.038 0.037 0.036 0.034 0.029 0.025...
        0.023 0.018 0.018 0.022 0.023 0.024 0.025 0.025 0.023 0.019...
        0.016 0.014 0.010 0.008 0.008 0.007 0.007 0.006 0.005 0.005...
        0.006 0.006 0.007 0.007 0.006 0.006 0.007 0.007 0.007 0.007;
    %
        1.000 1.000 1.000 1.000 1.000 0.997 0.989 0.983 0.975 0.967...
        0.961 0.956 0.952 0.946 0.942 0.937 0.927 0.906 0.885 0.864...
        0.842 0.820 0.800 0.781 0.760 0.741 0.720 0.698 0.670 0.641...
        0.613 0.584 0.555 0.528 0.503 0.475 0.448 0.424 0.397 0.382...
        0.365 0.348 0.335 0.317 0.304 0.292 0.278 0.266 0.255 0.243...
        0.234 0.226 0.217 0.209 0.201 0.190 0.177 0.164 0.151 0.138...
        0.124 0.123 0.122 0.121 0.120 0.119 0.115 0.103 0.089 0.077...
        0.065 0.053 0.047 0.045 0.043 0.039 0.037 0.035 0.034 0.033...
        0.032 0.029 0.028 0.026 0.025 0.024 0.023 0.021 0.018 0.0015...
        0.013 0.008 0.008 0.006 0.002 0.005 0.006 0.008 0.010 0.012;
    %
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 0.999 0.999 0.998 0.997 0.996 0.995 0.988 0.984 0.977...
        0.973 0.966 0.958 0.950 0.942 0.934 0.925 0.915 0.898 0.884...
        0.868 0.854 0.837 0.818 0.798 0.779 0.760 0.740 0.722 0.706...
        0.682 0.676 0.661 0.646 0.632 0.617 0.603 0.598 0.574 0.560...
        0.546 0.530 0.516 0.501 0.486 0.464 0.433 0.400 0.368 0.337...
        0.305 0.288 0.274 0.257 0.242 0.227 0.209 0.188 0.167 0.147...
        0.126 0.105 0.096 0.092 0.085 0.079 0.074 0.067 0.062 0.055...
        0.049 0.043 0.036 0.034 0.034 0.034 0.034 0.034 0.033 0.028...
        0.025 0.020 0.015 0.012 0.008 0.008 0.008 0.008 0.008 0.008;
    %
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 0.999 0.999 0.999 0.998...
        0.998 0.997 0.996 0.993 0.988 0.986 0.983 0.978 0.974 0.967...
        0.962 0.956 0.950 0.942 0.930 0.918 0.907 0.897 0.886 0.876...
        0.867 0.857 0.847 0.837 0.828 0.822 0.815 0.807 0.800 0.793...
        0.769 0.745 0.721 0.697 0.674 0.648 0.625 0.602 0.578 0.553...
        0.528 0.515 0.502 0.488 0.475 0.462 0.439 0.397 0.355 0.313...
        0.272 0.228 0.208 0.195 0.180 0.166 0.152 0.138 0.133 0.125...
        0.117 0.110 0.104 0.100 0.098 0.097 0.096 0.095 0.094 0.082...
        0.068 0.055 0.044 0.031 0.025 0.025 0.025 0.026 0.027 0.027;
    %
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 0.998 0.993 0.988 0.984 0.980 0.977 0.974...
        0.972 0.971 0.970 0.970 0.968 0.964 0.959 0.953 0.948 0.943...
        0.937 0.927 0.914 0.905 0.894 0.882 0.872 0.865 0.857 0.850...
        0.842 0.834 0.823 0.810 0.798 0.787 0.775 0.750 0.705 0.660...
        0.618 0.574 0.526 0.510 0.500 0.490 0.480 0.470 0.457 0.433...
        0.408 0.384 0.360 0.335 0.315 0.295 0.278 0.257 0.237 0.218...
        0.190 0.130 0.102 0.072 0.057 0.051 0.046 0.039 0.034 0.027];
%
%----------------------------------------------------------------------------------------------------
```

```
% Initializing constants required for the gradient method of minimizing the error
between
% the output of the experimental data and model output
%
  K = 100;                 % number of steps in the optimization trajectory based on
                               sampling period
  M = 2000;                % total number of iterations for stopping the iteration procedure
  %
  % dt = 2.35;             % sampling period (original sampling period is 2.35min)
  dt = 0.012;              % more practical sampling period
  %
  delta_p = 0.001;         % small increment of parameter values as being tested for
                               optimal values
  epsilon = 0.0001;        % error for stopping the iteration
  grad = 0.82;             % step of the gradient procedure (10.78 was best), the best
                               program used 10.82
  %
  r = 23;
%
%--------------------------------------------------------------------------------------------------
% State space and control vectors full trajectory initial values
  FR = FR*ones(1,K);                         % the control input
  states_in = conc_in*ones(1,K+1);           % the disturbance input
  states = states_bar;                       % states_bar is the measured states
%
%--------------------------------------------------------------------------------------------------
% Iteration procedure start and initializing index of iteration (j)
  j = 1;
  e1 = cputime;
  x0 = states(:,1);
  while j <= M
    % Solving the state model equation starting with initial values of parameters
    %
    for k = 1:K-1          % at the end of this iteration states should be xER^(NxK+1)
        FRk = FR(k);
        statesk_in = states_in(k);
        statesk = states(:,k);
        statesk1(:,k+1) = iestatek1(statesk,FRk,statesk_in,x0,p,dt);
```

```
end
%
%---------------------------------------------------------------------------------------------
% Calculating the full trajectories error between measured and calculated states
     based on estimated parameters
error = abs(states_bar - statesk1);
%
Jp = 0;
% Calculate the least squares differences
for k = 1:K
    Jp = Jp + sum(error(:,k).*error(:,k));
end

%---------------------------------------------------------------------------------------------
% Calculation of the deviated value of the criterion and
% Calculation of the deviated value of the parameters
for i = 1:r
    p(i) = p(i)+ delta_p;
    % Calculation of the value of the states with the deviated parameters
     x0 = statesk1(:,1);
    for k = 1:K-1
        FRk = FR(k);
        statesk = states(:,k);
        statesk_in = states_in(k);
        statesk1(:,k+1) =  iestatek1(statesk,FRk,statesk_in,x0,p,dt);
    end
    %-----------------------------------------------------------------------------------------
    % Calculation of the deviated value of the criterion dJp based on the error
      error = abs(states_bar - statesk1);
      dJp = 0;
    for k = 1:K
        dJp = dJp + sum(error(:,k).*error(:,k));
    end
    %-----------------------------------------------------------------------------------------
    % Calculation of the gradient of the criterion
      grad_dJp = (dJp - Jp)/delta_p;
    %
    % Form a vector of the criterion gradients for all newly calculated parameters
```

```
      grad_dJpv(i) = grad_dJp;
    %
    % Calculate the old value of the parameters
      p(i) = p(i) - delta_p;
end
%
%-----------------------------------------------------------------------------------------------
% Calculation of the gradient for the fastest descent method and
% Calculation of the weighted sums
  S = grad_dJpv*grad_dJpv';
  S = sqrt(S);
%----------------------------------------------------------------------------------------------
% Calculation of the direction of the gradient, dpER^r = dpER^23
  dp = -(grad*grad_dJpv)/S;
%
% Check if the achieved solution is an optimal one using the norm
% Calculation of the norm
  dnorm = norm(dp);
%
% Check termination of the calculation
if dnorm <= epsilon
   x0 = statesk1(:,1) ;
   for k = 1:K-1
       FRk = FR(k);
       statesk = states(:,k);
       statesk_in = states_in(k);
       statesk1(:,k+1) = iestatek1(statesk,FRk,statesk_in,x0,p,dt);
    end
break
else
    % Calculate the improved estimate of the parameters
    p = p + dp;
    %alpha = alpha/2;      % was used as test in fast reducing step factor
    %
    x0 = statesk1(:,1);     % set next initial state to current state
    %
    j = j+1;                % increment optimization index
end
```

```
  end
%
%-----------------------------------------------------------------------------------------------
% Display optimal parameters calculated
p_final = p';
%j=j
%
Jk1 = 0;
for k = 1:K
    error(:,k) = states_bar(:,k)- statesk1(:,k);
    Jk = (error(:,k)'*error(:,k));
    Jk1 = Jk1 + Jk;
end
e2 = cputime;
%
%--------------------------------------------------------------------------------------------------
% Display optimal parameters calculated
% j
% p_final = p'
  error = abs(error);
  Jk1
  e = e2 - e1
%-------------------------------------------------------------------------------------------------
% Plot the output trajectories
%   figure(1)
%   k = 1:K;
%   plot(k,states_bar,'-o','Linewidth',2.5,'MarkerSize',6)
%   grid on
%   xlabel('Discrete time [k]','Fontsize',16)
%   ylabel('Process true state [eq/l]','Fontsize',16)
%   title('System dynamic behaviour based on real state of the process','Fontsize',16)
%
%---------------------------------------------------------------------------------------------------
  figure(2)
  i = 1:K;
  plot(i,error,':o','Linewidth',2.5,'MarkerSize',5)
  text('Position',[10.0 0.15],'String','e_x_1 - e_x_6','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  hold on
  %
  plot(i,statesk1,'-o','Linewidth',2.5,'MarkerSize',5)
  text('Position',[15.0 0.35],'String','x_1','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[25.0 0.45],'String','x_2','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[35.0 0.5],'String','x_3','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[45.0 0.65],'String','x_4','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[55.0 0.70],'String','x_5','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[65.0 0.82],'String','x_6','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%
  xlabel('Discrete time [k]','Fontsize',16)
  ylabel('Estimated states & error difference [eq/l]','Fontsize',16)
  title('System dynamic behaviour based on estimated model','Fontsize',16)
  grid on
  hold off
%
 figure(3)
 subplot(2,1,1)
 i = 1:K;
 plot(i,states_bar,':o','Linewidth',2.5,'MarkerSize',5)
 text('Position',[15.0 0.35],'String','x_1','FontName','Times New Roman',
       'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[25.0 0.45],'String','x_2','FontName','Times New Roman',
       'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[35.0 0.5],'String','x_3','FontName','Times New Roman',
       'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[45.0 0.65],'String','x_4','FontName','Times New Roman',
       'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[55.0 0.70],'String','x_5','FontName','Times New Roman',
       'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[65.0 0.82],'String','x_6','FontName','Times New Roman',
       'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 xlabel('Discrete time [k]','Fontsize',14)
 ylabel('Process true state [eq/l]','Fontsize',14)
 title('Dynamic behaviour of the process','Fontsize',14)
 grid on
%
 subplot(2,1,2)
 i = 1:K;
 plot(i,statesk1,'-o','Linewidth',2.5,'MarkerSize',5)
 text('Position',[15.0 0.35],'String','x^~_1','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[25.0 0.45],'String','x^~_2','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[35.0 0.5],'String','x^~_3','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[45.0 0.65],'String','x^~_4','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[55.0 0.70],'String','x^~_5','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[65.0 0.82],'String','x^~_6','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 xlabel('Discrete time [k]','Fontsize',14)
 ylabel('Estimated state [eq/l]','Fontsize',14)
 title('Dynamic behaviour of the process','Fontsize',14)
 grid on
 hold off
 %---------------------------------------------------------------------------------------------------
% Individual plot (if preferred)
%  figure(3)
%  subplot(2,1,1)
%  i = 1:K;
%  plot(i,states_bar(1,:),':o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[15.0 0.35],'String','x_1','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  plot(i,states_bar(2,:),':o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[25.0 0.45],'String','x_2','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  hold on
%  plot(i,states_bar(3,:),':o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[35.0 0.5],'String','x_3','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  hold on
%  plot(i,states_bar(4,:),':o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[45.0 0.65],'String','x_4','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  hold on
%  plot(i,states_bar(5,:),':o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[55.0 0.70],'String','x_5','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  hold on
%  plot(i,states_bar(6,:),':o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[65.0 0.82],'String','x_6','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  hold on
%  hold off
%
%  subplot(2,1,2)
%  i = 1:K;
%  plot(i,statesk1(1,:),'-o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[15.0 0.35],'String','x^~_1','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  hold on
%  plot(i,statesk1(2,:),'-o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[25.0 0.45],'String','x^~_2','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  hold on
%  plot(i,statesk1(3,:),'-o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[35.0 0.5],'String','x^~_3','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  hold on
%  plot(i,statesk1(4,:),'-o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[45.0 0.65],'String','x^~_4','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  hold on
%  plot(i,statesk1(5,:),'-o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[55.0 0.70],'String','x^~_5','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
```

```
%  hold on
%  plot(i,statesk1(6,:),'-o','Linewidth',2.5,'MarkerSize',5)
%  text('Position',[65.0 0.82],'String','x^~_6','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%  hold on
%  hold off
%-------------------------------------------------------------------------------------------------
  figure(4)
  k = 1:K;
  plot(k,error,'o','Linewidth',2.5,'MarkerSize',6)
  text('Position',[80.0 0.05],'String','e_x_1 - e_x_6','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[10.0 0.8],'String','e_x_6','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  grid on
  xlabel('Discrete time [k]','Fontsize',16)
  ylabel('Error difference between states','Fontsize',16)
  title('Error difference between real and model states of the process','Fontsize',16)
%
%------------------------------------------------------------------------------------------------
figure(5)
k = 1:K;
subplot(3,2,1)
plot(k,statesk1(1,:),':')
hold on
plot(k,states_bar(1,:))
text('Position',[30.0 0.15],'String','x1bar','Fontsize',12)
text('Position',[60.0 0.20],'String','x1','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('Real & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
subplot(3,2,2)
plot(k,statesk1(2,:),':')
hold on
plot(k,states_bar(2,:))
text('Position',[60.0 0.15],'String','x2bar','Fontsize',12)
text('Position',[30.0 0.40],'String','x2','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('Real & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
subplot(3,2,3)
plot(k,statesk1(3,:),':')
hold on
plot(k,states_bar(3,:))
text('Position',[60.0 0.25],'String','x3bar','Fontsize',12)
text('Position',[40.0 0.55],'String','x3','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('Real & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
subplot(3,2,4)
plot(k,statesk1(4,:),':')
hold on
plot(k,states_bar(4,:))
text('Position',[80.0 0.15],'String','x4bar','Fontsize',12)
text('Position',[50.0 0.35],'String','x4','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('Real & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
subplot(3,2,5)
plot(k,states_bar(5,:))
hold on
plot(k,statesk1(5,:),':')
text('Position',[60.0 0.25],'String','x5bar','Fontsize',12)
text('Position',[40.0 0.30],'String','x5','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('Real & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
```

```
%
subplot(3,2,6)
plot(k,statesk1(6,:))
hold on
plot(k,states_bar(6,:),':')
text('Position',[20.0 0.35],'String','x6','Fontsize',12)
text('Position',[60.0 0.40],'String','x6bar','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('Real & estimated states [eq/l]')
title('Dynamic behaviour of the process')
hold off
%
%----------------------------------Program end-----------------------------------------------

% -------------------------------------------------------------------------------------------------
% The function is used to calculate the values of the states for parameter estimation,
% the considered trajectory period is (K = 100).
% The function is called in the main program (see details below) under the function
    name iestate()
% The main program is called "Parameter_Estimation6a_States.m"
%
% Author: NM Dube
% Dated: 22 August 2011
% Updated: 21 November 2016
%------------------------------------------------------------------------------------------------------
% declare the function iestate()
function statesk1 = iestatek1(statesk,FRk,statesk_in,x0,p,dt)
%
% solving the model state equation using the current parameter values
 x1 = statesk(1) + dt*(p(1)*statesk_in - p(1)*statesk(1) + p(2)*FRk*statesk(2) -
       p(3)*FRk*statesk(1) + p(4)*FRk);
 x2 = statesk(2) + dt*(p(5)*statesk(1) - p(5)*statesk(2) + p(6)*FRk*statesk(3) -
       p(7)*FRk*statesk(2) + p(8)*FRk);
 x3 = statesk(3) + dt*(p(9)*statesk(2) - p(9)*statesk(3) + p(10)*FRk*statesk(4) -
       p(11)*FRk*statesk(3) + p(12)*FRk);
 x4 = statesk(4) + dt*(p(13)*statesk(3) - p(13)*statesk(4) + p(14)*FRk*statesk(5) -
       p(15)*FRk*statesk(4) + p(16)*FRk);
 x5 = statesk(5) + dt*(p(17)*statesk(4) - p(17)*statesk(5) + p(18)*FRk*statesk(6) -
        p(19)*FRk*statesk(5) + p(20)*FRk);
 x6 = statesk(6) + dt*(p(21)*statesk(5) - p(21)*statesk(6) + 0 - p(22)*FRk*statesk(6) +
        p(23)*FRk);
%-----------------------------------------------------------------------------------------------------------
% declaring the state variables for the state vector at k+1, k=1:K
statesk1 = [x1 x2 x3 x4 x5 x6]';
%
%--------------------------------------------Function end-------------------------------------
```

APPENDIX B.3: Program for parameter estimation using the state vector measurements (direct method)

```
% ------------------------------------------------------------------------------------
% Program for linear parameter estimation for a CCIX(continuous countercurrent
% ion exchange) process, based on direct method of solving a p-vector;
% CCIX in this application is used for water desalination.
% A model has been developed for real-time optimal control and therefore,
% parameter estimation must be performed.
% In this program unknown parameters are estimated using a direct solution, based
on
% measured initial states, process input and measured states.
% r = 23 is the total number of parameters since N = 6.
% The model is based on the six stages UCT (University of Cape Town) model, with
6 stages.
% The program estimates parameters using an state vector-based method (the direct
method).
%
% Author: NM Dube
% Dated: 6 September 2011
% Updated: 31 December 2016
%
%------------------------------------------------------------------------------------
% Initializing of the model equation from available data obtained from UCT project
%
  h = 32.93;            % resin holdups (units in litres)
%
%------------------------------------------------------------------------------------
%
  d = 2/3;              % resin to liquid fraction constant
 conc_in = 0.435;        % sodium concentration at startup of process (eq/l)
% conc_in = 0.055
% conc_in = 0.110
% conc_in = 0.435
% conc_in = 0.550
% conc_in = 0.754
% conc_in = 1.000
  T = 17;               % liquid upflow period (min)
  FR = (h*d)/T           % the control variable, resin flowrate (units in litres/min)
 Tf = T/17
%
%-------------------------------------------------------------------------------------
% Initializing constants required for the gradient method of minimizing the error
   between
% the output of the experimental data and model output
%
   dt = 2.35/195            % sampling period (dt = 2.7, real sampling based on
                                  minutes),
  K = 100;                 % total number of points in the full trajectory,
%
%-------------------------------------------------------------------------------------
% Measured data
% Stage H+ fractional change in liquid concentration using data obtained from UCT
Project volume 4
 stage1 = [0.221 0.577 0.730 0.847 0.920 0.936 0.968 0.974 0.981 0.989 0.988
             0.997 1.000 1.000 1.000 1.000 1.000 1.000];
 stage2 = [0.000 0.140 0.314 0.523 0.656 0.766 0.842 0.886 0.900 0.933 0.958
             0.963 0.982 0.974 0.991 0.994 0.993 0.993];
 stage3 = [0.000 0.04 0.066 0.184 0.295 0.454 0.601 0.690 0.758 0.804 0.877 0.881
             0.951 0.965 0.972 0.981 1.000 0.988];
 stage4 = [0.000 0.000 0.004 0.035 0.082 0.168 0.277 0.361 0.440 0.522 0.698
             0.784 0.899 0.931 0.966 0.966 0.991 0.991];
 stage5 = [0.000 0.000 0.000 0.003 0.020 0.052 0.113 0.167 0.207 0.340 0.474
             0.547 0.780 0.860 0.899 0.905 0.975 0.973];
 stage6 = [0.000 0.000 0.000 0.000 0.000 0.001 0.024 0.033 0.063 0.124 0.167
             0.233 0.482 0.539 0.672 0.779 0.940 0.972];
%
% Obtain Na+ fractional change in liquid concentration
 stage1 = 1- stage1;
 stage2 = 1- stage2;
 stage3 = 1- stage3;
 stage4 = 1- stage4;
 stage5 = 1- stage5;
 stage6 = 1- stage6;
%-------------------------------------------------------------------------------------
% Introducing measurement vector expanded for the full state trajectory
states_bar = [1.000 1.000 1.000 1.000 0.820 0.750 0.686 0.622 0.558 0.524...
```

```
        0.490 0.401 0.371 0.361 0.354 0.300 0.268 0.254 0.237 0.220...
        0.178 0.157 0.143 0.130 0.116 0.104 0.090 0.078 0.077 0.074...
        0.071 0.068 0.065 0.060 0.055 0.048 0.043 0.037 0.033 0.032...
        0.029 0.028 0.027 0.026 0.025 0.024 0.023 0.022 0.020 0.018...
        0.017 0.016 0.015 0.014 0.013 0.012 0.012 0.012 0.012 0.012...
        0.012 0.012 0.008 0.007 0.005 0.004 0.003 0.002 0.002 0.002...
        0.001 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000...
        0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000...
        0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000;
%
        1.000 1.000 1.000 1.000 1.000 0.988 0.964 0.938 0.914 0.887...
        0.863 0.832 0.800 0.770 0.738 0.707 0.674 0.635 0.598 0.561...
        0.524 0.485 0.458 0.435 0.410 0.386 0.364 0.340 0.320 0.300...
        0.280 0.261 0.241 0.225 0.212 0.197 0.184 0.170 0.157 0.149...
        0.142 0.134 0.126 0.117 0.114 0.110 0.107 0.105 0.103 0.100...
        0.094 0.088 0.083 0.076 0.070 0.065 0.061 0.056 0.053 0.047...
        0.043 0.042 0.041 0.039 0.038 0.037 0.036 0.034 0.029 0.025...
        0.023 0.018 0.018 0.022 0.023 0.024 0.025 0.025 0.023 0.019...
        0.016 0.014 0.010 0.008 0.008 0.007 0.007 0.006 0.005 0.005...
        0.006 0.006 0.007 0.007 0.006 0.006 0.007 0.007 0.007 0.007;
%
        1.000 1.000 1.000 1.000 1.000 0.997 0.989 0.983 0.975 0.967...
        0.961 0.956 0.952 0.946 0.942 0.937 0.927 0.906 0.885 0.864...
        0.842 0.820 0.800 0.781 0.760 0.741 0.720 0.698 0.670 0.641...
        0.613 0.584 0.555 0.528 0.503 0.475 0.448 0.424 0.397 0.382...
        0.365 0.348 0.335 0.317 0.304 0.292 0.278 0.266 0.255 0.243...
        0.234 0.226 0.217 0.209 0.201 0.190 0.177 0.164 0.151 0.138...
        0.124 0.123 0.122 0.121 0.120 0.119 0.115 0.103 0.089 0.077...
        0.065 0.053 0.047 0.045 0.043 0.039 0.037 0.035 0.034 0.033...
        0.032 0.029 0.028 0.026 0.025 0.024 0.023 0.021 0.018 0.015...
        0.013 0.008 0.008 0.006 0.002 0.005 0.006 0.008 0.001 0.002;

        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 0.999 0.999 0.998 0.997 0.996 0.995 0.988 0.984 0.977...
        0.973 0.966 0.958 0.950 0.942 0.934 0.925 0.915 0.898 0.884...
        0.868 0.854 0.837 0.818 0.798 0.779 0.760 0.740 0.722 0.706...
        0.682 0.676 0.661 0.646 0.632 0.617 0.603 0.598 0.574 0.560...
        0.546 0.530 0.516 0.501 0.486 0.464 0.433 0.400 0.368 0.337...
        0.277 0.268 0.257 0.249 0.242 0.227 0.209 0.188 0.167 0.147...
        0.126 0.105 0.096 0.092 0.085 0.079 0.074 0.067 0.062 0.055...
        0.049 0.043 0.036 0.034 0.034 0.034 0.034 0.034 0.033 0.028...
        0.025 0.020 0.015 0.012 0.008 0.008 0.008 0.008 0.008 0.008;
%
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 0.994...
        0.992 0.981 0.970 0.970 0.968 0.964 0.960 0.955 0.951 0.948...
        0.945 0.939 0.924 0.915 0.894 0.882 0.872 0.865 0.857 0.850...
        0.842 0.834 0.823 0.810 0.798 0.787 0.775 0.750 0.722 0.690...
        0.678 0.646 0.626 0.590 0.582 0.573 0.566 0.532 0.517 0.493...
        0.467 0.452 0.423 0.418 0.401 0.374 0.331 0.290 0.282 0.256...
        0.232 0.228 0.208 0.195 0.180 0.166 0.152 0.138 0.133 0.125...
        0.117 0.110 0.104 0.100 0.088 0.077 0.066 0.055 0.044 0.032...
        0.028 0.020 0.014 0.011 0.009 0.007 0.005 0.002 0.002 0.002;
%
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 0.999 0.999 0.999 0.998...
        0.998 0.997 0.996 0.993 0.988 0.986 0.983 0.978 0.974 0.967...
        0.962 0.956 0.950 0.942 0.930 0.918 0.907 0.897 0.886 0.876...
        0.867 0.857 0.847 0.837 0.828 0.822 0.815 0.807 0.800 0.793...
        0.769 0.745 0.721 0.697 0.674 0.648 0.625 0.602 0.578 0.553...
        0.528 0.515 0.502 0.488 0.475 0.462 0.439 0.397 0.355 0.313...
        0.299 0.284 0.278 0.266 0.251 0.247 0.233 0.221 0.211 0.205...
        0.200 0.198 0.189 0.176 0.166 0.142 0.130 0.122 0.114 0.108...
        0.102 0.091 0.080 0.071 0.064 0.050 0.032 0.021 0.018 0.011];
%
%-----------------------------------------------------------------------------------------------
% Initializing the state and control vectors full trajectory initial values
  FR = FR*ones(1,K);
  states_in = conc_in*ones(1,K+1);
  states = states_bar;              % declaring initial state variables
%
%----------------------------------------------------------------------------------------------------
%
  e0 = cputime;
  Jk1 = 0;
  for k = 1:K-1
```

```
    FRk = FR(k);
    statesk = states(:,k);
    statesk_in =states_in(k);
    %
    % Calculate the F(k) matrix
     Fk = [(statesk_in-statesk(1)) statesk(2)*FRk -(statesk(1)*FRk) FRk 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0;
            0 0 0 0  (statesk(1)-statesk(2)) statesk(3)*FRk -(statesk(2)*FRk) FRk 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0;
            0 0 0 0 0 0 0 0 (statesk(2)-statesk(3)) statesk(4)*FRk -(statesk(3)*FRk) FRk 0
0 0 0 0 0 0 0 0 0 0 0;
            0 0 0 0 0 0 0 0 0 0 0 0 (statesk(3)-statesk(4)) statesk(5)*FRk -
(statesk(4)*FRk) FRk 0 0 0 0 0 0 0 0;
            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (statesk(4)-statesk(5)) statesk(6)*FRk -
(statesk(5)*FRk) FRk 0 0 0;
            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (statesk(5)-statesk(6)) -(statesk(6)*FRk)
FRk];
    %
    % Work out the parameters for each k-moment
    Ek = Fk'*(states(:,k+1)-states(:,k))/dt;
    P1 = Fk'*Fk;
    p = Ek\P1;              % constant parameters
    pk(:,k) = p';           % or this can be left out and use p above to see p vales at each
k moment,
                            % but if used in that manner p values are not stored.
%
%------------------------------------------------------------------------------------------------
    x1 = statesk(1) + dt*(p(1)*statesk_in - p(1)*statesk(1) + p(2)*FRk*statesk(2) -
            p(3)*FRk*statesk(1) + p(4)*FRk);
    x2 = statesk(2) + dt*(p(5)*statesk(1) - p(5)*statesk(2) + p(6)*FRk*statesk(3) -
            p(7)*FRk*statesk(2) + p(8)*FRk);
    x3 = statesk(3) + dt*(p(9)*statesk(2) - p(9)*statesk(3) + p(10)*FRk*statesk(4) -
             p(11)*FRk*statesk(3) + p(12)*FRk);
    x4 = statesk(4) + dt*(p(13)*statesk(3) - p(13)*statesk(4) + p(14)*FRk*statesk(5) -
            p(15)*FRk*statesk(4) + p(16)*FRk);
    x5 = statesk(5) + dt*(p(17)*statesk(4) - p(17)*statesk(5) + p(18)*FRk*statesk(6) -
            p(19)*FRk*statesk(5) + p(20)*FRk);
    x6 = statesk(6) + dt*(p(21)*statesk(5) - p(21)*statesk(6) + p(22)*FRk*0 -
            p(22)*FRk*statesk(6) + p(23)*FRk);
%
%----------------------------------------------------------------------------------------------------
%  Declaring the state variables for the state vector at k+1, k=1:K using calculated
      parameters
    states_est(:,k) = [x1 x2 x3 x4 x5 x6]';
    error(:,k) = states_bar(:,k)- states_est(:,k);
    %
    Jk = (error(:,k)'*error(:,k));            % calculate the least square error
    Jk1 = Jk1 + Jk;                          % calculate the improved error
  end
  e1 = cputime;                              % end-point for processing time
%
%-----------------------------------------------------------------------------------------------
% Display of optimalilty test values
  error = abs(error);
  e = e1 - e0
  Jk1
% Plot the output trajectories
% pk
%
 figure(1)
 i = 1:K;
 plot(i,states_bar,':o','Linewidth',2.5,'MarkerSize',5)
 text('Position',[15.0 0.35],'String','x_1','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[25.0 0.45],'String','x_2','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[35.0 0.5],'String','x_3','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[45.0 0.65],'String','x_4','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[55.0 0.70],'String','x_5','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[65.0 0.82],'String','x_6','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 xlabel('Discrete time [k]','Fontsize',14)
```

```
ylabel('Process true state [eq/l]','Fontsize',14)
title('Dynamic behaviour of the process','Fontsize',14)
grid on
%
figure(2)
subplot(2,1,1)
i = 1:K;
plot(i,states_bar,':o','Linewidth',2.5,'MarkerSize',5)
text('Position',[15.0 0.35],'String','x_1','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[25.0 0.45],'String','x_2','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[35.0 0.5],'String','x_3','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[45.0 0.65],'String','x_4','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[55.0 0.70],'String','x_5','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[65.0 0.82],'String','x_6','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
xlabel('Discrete time [k]','Fontsize',14)
ylabel('Process true state [eq/l]','Fontsize',14)
title('Dynamic behaviour of the process','Fontsize',14)
grid on
%
 subplot(2,1,2)
 i = 1:K-1;
 plot(i,states_est,'-o','Linewidth',2.5,'MarkerSize',5)
 text('Position',[15.0 0.35],'String','x^~_1','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[25.0 0.45],'String','x^~_2','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[35.0 0.5],'String','x^~_3','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[45.0 0.65],'String','x^~_4','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[55.0 0.70],'String','x^~_5','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
```

```
  text('Position',[65.0 0.82],'String','x^~_6','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  xlabel('Discrete time [k]','Fontsize',14)
  ylabel('Estimated state [eq/l]','Fontsize',14)
  title('Dynamic behaviour of the process','Fontsize',14)
  grid on
  hold off
 %
  figure(3)
  i = 1:K-1;
  plot(i,error,':o','Linewidth',2.5,'MarkerSize',5)
  text('Position',[10.0 0.1],'String','e_x_1 - e_x_6','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  hold on
  %
  i = 1:K;
  plot(i,states_bar(1,:),'-ob','Linewidth',2.5,'MarkerSize',5)
  text('Position',[15.0 0.35],'String','x_1','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  hold on
  plot(i,states_bar(2,:),'-ob','Linewidth',2.5,'MarkerSize',5)
  text('Position',[25.0 0.45],'String','x_2','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  hold on
  plot(i,states_bar(3,:),'-ob','Linewidth',2.5,'MarkerSize',5)
  text('Position',[35.0 0.5],'String','x_3','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  hold on
  plot(i,states_bar(4,:),'-ob','Linewidth',2.5,'MarkerSize',5)
  text('Position',[45.0 0.65],'String','x_4','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  hold on
  plot(i,states_bar(5,:),'-ob','Linewidth',2.5,'MarkerSize',5)
  text('Position',[55.0 0.70],'String','x_5','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  hold on
  plot(i,states_bar(6,:),'-ob','Linewidth',2.5,'MarkerSize',5)
 text('Position',[65.0 0.82],'String','x_6','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 hold on
 xlabel('Discrete time [k]','Fontsize',16)
 ylabel('Estimated states & error difference [eq/l]','Fontsize',16)
 title('System error & dynamic behaviour based on estimated model','Fontsize',16)
 grid on
 hold off
% ---------------------------------------------------------------------------------------------
figure(4)
k = 1:K-1;
plot(k,error,':o','Linewidth',2.5)
text('Position',[50.0 0.02],'String','e_x_1 - e_x_6','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
xlabel('Discrete time [k]','Fontsize',16)
ylabel('Error difference between states [eq/l]','Fontsize',16)
title('Error difference between measured & calculated states','Fontsize',16)
grid on
hold on
%
figure (5)
subplot(3,2,1)
k = 1:K-1;
plot(k,states_est(1,:),':')
hold on
k = 1:K;
plot(k,states_bar(1,:))
text('Position',[30.0 0.13],'String','x1bar','Fontsize',12)
text('Position',[15.0 0.40],'String','x1','Fontsize',12)
xlabel('Discrete time [k]')
ylabel('Real & estimated states[eq/l]')
title('Dynamic behaviour of the process')
hold off

subplot(3,2,2)
k = 1:K-1;
plot(k,states_est(2,:),':')
hold on
```

```
 k = 1:K;
 plot(k,states_bar(2,:))
 text('Position',[60.0 0.15],'String','x2bar','Fontsize',12)
 text('Position',[15.0 0.65],'String','x2','Fontsize',12)
 xlabel('Discrete time [k]')
 ylabel('Real & estimated states[eq/l]')
 title('Dynamic behaviour of the process')
 hold off
%
 subplot(3,2,3)
 k = 1:K-1;
 plot(k,states_est(3,:),':')
 hold on
 k = 1:K;
 plot(k,states_bar(3,:))
 text('Position',[60.0 0.35],'String','x3bar','Fontsize',12)
 text('Position',[27.0 0.5],'String','x3','Fontsize',12)
 xlabel('Discrete time [k]')
 ylabel('Real & estimated states[eq/l]')
 title('Dynamic behaviour of the process')
 hold off
%
 subplot(3,2,4)
 k = 1:K-1;
 plot(k,states_est(4,:),':')
 hold on
 k = 1:K;
 plot(k,states_bar(4,:))
 text('Position',[80.0 0.15],'String','x4bar','Fontsize',12)
 text('Position',[50.0 0.5],'String','x4','Fontsize',12)
 xlabel('Discrete time [k]')
 ylabel('Real & estimated states[eq/l]')
 title('Dynamic behaviour of the process')
 hold off
%
 subplot(3,2,5)
 k = 1:K-1;
 plot(k,states_est(5,:),':')
 hold on
 k = 1:K;
 plot(k,states_bar(5,:))
 text('Position',[65.0 0.60],'String','x5bar','Fontsize',12)
 text('Position',[55.0 0.70],'String','x5','Fontsize',12)
 xlabel('Discrete time [k]')
 ylabel('Real & estimated states[eq/l]')
 title('Dynamic behaviour of the process')
 hold off
%
 subplot(3,2,6)
 k = 1:K-1;
 plot(k,states_est(6,:),':')
 hold on
 k = 1:K;
 plot(k,states_bar(6,:))
 text('Position',[60.0 0.55],'String','x6','Fontsize',12)
 text('Position',[70.0 0.50],'String','x6bar','Fontsize',12)
 xlabel('Discrete time [k]')
 ylabel('Real & estimated states[eq/l]')
 title('Dynamic behaviour of the process')
 hold off
%-----------------------------------------------Program end--------------------------------
```

APPENDIX B.4: Program for parameter estimation using the state vector measurements (Lagrange method)

```
% ---------------------------------------------------------------------------------------------------------
% Program for solution of parameter estimation for a CCIX(continuous
countercurrent
% ion exchange) process, CCIX in this application is used for water desalination.
% A model has been developed for real-time optimal control of the process and
therefore,
% parameter estimation must be performed.
% The model is based on the six stages UCT (University of Cape Town) model,
% and the current project has 6 stages.
% The program estimates parameters using an the Lagrange method.
%
% Author: NM Dube
% Dated: 23 June 2011
% Updated: 15 December 2016
%
%----------------------------------------------------------------------------------------------------
% Initializing of the model equation from available data
%
  h = 0.3293;                % resin holdups (units in litres)
  H = 0.42809;               % liquid holdups (units in litres)
  Fl = 2000/60;              % liquid flow rate (units in litres/min)
%
  conc_in = 0.435;           % disturbance input
  T = 17;                    % liquid upflow time (units in min)
  d = 2/3;                   % resin to liquid fraction relationship
  FR = (h*d)/T;              % initial control variable, resin flowrate.
%
%-------------------------------------------------------------------------------------------------
% Initializing variables used in the model formulas with data from UCT project
%
  a = 230*[0.8 0.6 0.5 0.4 0.3 0.25];        % (try this for better results)
  b = 60*[0.01 0.02 0.05 0.08 0.1 0.2];      % ionic liquid fraction in the solution
%
%----------------------------------------------------------------------------------------------------
% Declaration of parameters from the model equations as determined by,
% dx/dt = Ax(t)+B1x(t)u(t)+Bu(t)+W(t)w(t) and
% determinaton of parameters are l(i), m(i), m(i(j)) and k(i)
%
N = 6;                      % number of stages of the process
for i = 1:N
   l(i) = Fl/(H+(a(i)*h));
   mi(i) = a(i)/(H+(a(i)*h));
end
%
for i = 1:N-1
   r(i) = (b(i+1) - b(i))/(H+a(i)*h);
   mij(i) = a(i+1)/(H+(a(i)*h));
end
   r(N) = -b(N)/(H+a(N)*h);
%
%-----------------------------------------------------------------------------------------------
% Initialization of model parameters A, B, B1 W and C using calculated parameters
li, mi and mij
 A(1,1) = -l(1);
 A(2,1) = l(2);
 A(2,2) = -l(2);
 A(3,2) = l(3);
 A(3,3) = -l(3);
 A(4,3) = l(4);
 A(4,4) = -l(4);
 A(5,4) = l(5);
 A(5,5) = -l(5);
 A(6,5) = l(6);
 A(6,6) = -l(6);
%
 B1(1,1) = -mi(1);
 B1(1,2) = mij(1);
 B1(2,2) = -mi(2);
 B1(2,3) = mij(2);
 B1(3,3) = -mi(3);
 B1(3,4) = mij(3);
 B1(4,4) = -mi(4);
 B1(4,5) = mij(4);
 B1(5,5) = -mi(5);
 B1(5,6) = mij(5);
 B1(6,6) = -mi(6);
%
 B = [r(1) r(2) r(3) r(4) r(5) r(6)]';
 W = [l(1) 0 0 0 0 0]';
 C = [0 0 0 0 0 1];
%
%-----------------------------------------------------------------------------------------------------
% Initialize unknown parameters using the declared parameters above,
% there will be (N*4)- 1 parameters the parameter mij(6) is thus equated to 0.
%
   p(1) = l(1);
   p(2) = mij(1);
   p(3) = mi(1);
   p(4) = r(1);
   p(5) = l(2);
   p(6) = mij(2);
   p(7) = mi(2);
   p(8) = r(2);
   p(9) = l(3);
   p(10) = mij(3);
   p(11) = mi(3);
   p(12) = r(3);
   p(13) = l(4);
   p(14) = mij(4);
   p(15) = mi(4);
   p(16) = r(4);
   p(17) = l(5);
   p(18) = mij(5);
   p(19) = mi(5);
   p(20) = r(5);
   p(21) = l(6);
   p(22) = mi(6);
   p(23) = r(6);
%
%------------------------------------------------------------------------------------------------
% Factor the parameters as required
%
```

```
   p0 =
       [p(1);p(2);p(3);p(4);p(5);p(6);p(7);p(8);p(9);p(10);p(11);p(12);p(13);p(14);p(15);p(
       16);p(17);p(18);p(19);p(20);p(21);p(22);p(23)];
   p_initial = p0
%
%--------------------------------------------------------------------------------------------------
% Initializing constants required for the gradient method of minimizing the error
between
% the output of the experimental data and model output
%
%----------------------------------------------------------------------------------------------------
% Measured data
% Stage H+ fractional change in liquid concentration using data obtained from UCT
Project volume 4
  stage1 = [0.221 0.577 0.730 0.847 0.920 0.936 0.968 0.974 0.981 0.989 0.988
              0.997 1.000 1.000 1.000 1.000 1.000 1.000];
  stage2 = [0.000 0.140 0.314 0.523 0.656 0.766 0.842 0.886 0.900 0.933 0.958
              0.963 0.982 0.974 0.991 0.994 0.993 0.993];
  stage3 = [0.000 0.04 0.066 0.184 0.295 0.454 0.601 0.690 0.758 0.804 0.877 0.881
              0.951 0.965 0.972 0.981 1.000 0.988];
  stage4 = [0.000 0.000 0.004 0.035 0.082 0.168 0.277 0.361 0.440 0.522 0.698
              0.784 0.899 0.931 0.966 0.966 0.991 0.991];
  stage5 = [0.000 0.000 0.000 0.003 0.020 0.052 0.113 0.167 0.207 0.340 0.474
              0.547 0.780 0.860 0.899 0.905 0.975 0.973];
  stage6 = [0.000 0.000 0.000 0.000 0.000 0.001 0.024 0.033 0.063 0.124 0.167
              0.233 0.482 0.539 0.672 0.779 0.940 0.972];
%
% Obtain Na+ fractional change in liquid concentration
   stage1 = 1- stage1;
   stage2 = 1- stage2;
   stage3 = 1- stage3;
   stage4 = 1- stage4;
   stage5 = 1- stage5;
   stage6 = 1- stage6;
%
%-------------------------------------------------------------------------------------------
% Introducing measurement vector expanded for the full state trajectory
states_bar = [1.000 1.000 1.000 1.000 0.760 0.750 0.686 0.622 0.558 0.494...
```

```
    0.490 0.398 0.371 0.371 0.344 0.316 0.228 0.264 0.243 0.220...
    0.178 0.157 0.143 0.130 0.116 0.104 0.090 0.078 0.077 0.074...
    0.071 0.068 0.065 0.060 0.055 0.048 0.043 0.037 0.033 0.032...
    0.029 0.028 0.027 0.026 0.025 0.024 0.023 0.022 0.020 0.018...
    0.017 0.016 0.015 0.014 0.013 0.012 0.012 0.012 0.012 0.012...
    0.012 0.012 0.008 0.007 0.005 0.004 0.003 0.002 0.002 0.002...
    0.001 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000...
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000...
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000;
%
    1.000 1.000 1.000 1.000 1.000 0.988 0.964 0.938 0.914 0.887...
    0.863 0.832 0.800 0.770 0.738 0.707 0.674 0.635 0.598 0.561...
    0.524 0.485 0.458 0.435 0.410 0.386 0.364 0.340 0.320 0.300...
    0.280 0.261 0.241 0.225 0.212 0.197 0.184 0.170 0.157 0.149...
    0.142 0.134 0.126 0.117 0.114 0.110 0.107 0.105 0.103 0.100...
    0.094 0.088 0.083 0.076 0.070 0.065 0.061 0.056 0.053 0.047...
    0.043 0.042 0.041 0.039 0.038 0.037 0.036 0.034 0.029 0.025...
    0.023 0.018 0.018 0.022 0.023 0.024 0.025 0.025 0.023 0.019...
    0.016 0.014 0.010 0.008 0.008 0.007 0.007 0.006 0.005 0.005...
    0.006 0.006 0.007 0.007 0.006 0.006 0.007 0.007 0.007 0.007;
%
    1.000 1.000 1.000 1.000 1.000 0.997 0.989 0.983 0.975 0.967...
    0.961 0.956 0.952 0.946 0.942 0.937 0.927 0.906 0.885 0.864...
    0.842 0.820 0.800 0.781 0.760 0.741 0.720 0.698 0.670 0.641...
    0.613 0.584 0.555 0.528 0.503 0.475 0.448 0.424 0.397 0.382...
    0.365 0.348 0.335 0.317 0.304 0.292 0.278 0.266 0.255 0.243...
    0.234 0.226 0.217 0.209 0.201 0.190 0.177 0.164 0.151 0.138...
    0.124 0.123 0.122 0.121 0.120 0.119 0.115 0.103 0.089 0.077...
    0.065 0.053 0.047 0.045 0.043 0.039 0.037 0.035 0.034 0.033...
    0.032 0.029 0.028 0.026 0.025 0.024 0.023 0.021 0.018 0.0015...
    0.013 0.008 0.008 0.006 0.002 0.005 0.006 0.008 0.010 0.012;
%
    1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
    1.000 0.999 0.999 0.998 0.997 0.996 0.995 0.988 0.984 0.977...
    0.973 0.966 0.958 0.950 0.942 0.934 0.925 0.915 0.898 0.884...
    0.868 0.854 0.837 0.818 0.798 0.779 0.760 0.740 0.722 0.706...
    0.682 0.676 0.661 0.646 0.632 0.617 0.603 0.598 0.574 0.560...
    0.546 0.530 0.516 0.501 0.486 0.464 0.433 0.400 0.368 0.337...
```

```
        0.305 0.288 0.274 0.257 0.242 0.227 0.209 0.188 0.167 0.147...
        0.126 0.105 0.096 0.092 0.085 0.079 0.074 0.067 0.062 0.055...
        0.049 0.043 0.036 0.034 0.034 0.034 0.034 0.034 0.033 0.028...
        0.025 0.020 0.015 0.012 0.008 0.008 0.008 0.008 0.008 0.008;
    %
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 0.999 0.999 0.999 0.998...
        0.998 0.997 0.996 0.993 0.988 0.986 0.983 0.978 0.974 0.967...
        0.962 0.956 0.950 0.942 0.930 0.918 0.907 0.897 0.886 0.876...
        0.867 0.857 0.847 0.837 0.828 0.822 0.815 0.807 0.800 0.793...
        0.769 0.745 0.721 0.697 0.674 0.648 0.625 0.602 0.578 0.553...
        0.528 0.515 0.502 0.488 0.475 0.462 0.439 0.397 0.355 0.313...
        0.272 0.228 0.208 0.195 0.180 0.166 0.152 0.138 0.133 0.125...
        0.117 0.110 0.104 0.100 0.098 0.097 0.096 0.095 0.094 0.082...
        0.068 0.055 0.044 0.031 0.025 0.025 0.025 0.026 0.027 0.027;
    %
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 0.998 0.993 0.988 0.984 0.980 0.977 0.974...
        0.972 0.971 0.970 0.970 0.968 0.964 0.959 0.953 0.948 0.943...
        0.937 0.927 0.914 0.905 0.894 0.882 0.872 0.865 0.857 0.850...
        0.842 0.834 0.823 0.810 0.798 0.787 0.775 0.750 0.705 0.660...
        0.618 0.574 0.526 0.510 0.500 0.490 0.480 0.470 0.457 0.433...
        0.408 0.384 0.360 0.335 0.315 0.295 0.278 0.257 0.237 0.218...
        0.190 0.130 0.102 0.072 0.057 0.051 0.046 0.039 0.034 0.027];
 %
%-----------------------------------------------------------------------------------------------------
% Initializing sampling parameters
%
    dt = 2.35/195;         % sampling period in (min)at 100 sampling points, dt = 0.012;
    K = 100;        % total number of sampling points in the full sampling period,
    M = 2000;
% Initializing the Lagrange multipliers
%
  lambda1 = 0.110;
  lambda2 = 0.445;
  lambda3 = 0.122;
  lambda4 = 0.662;
  lambda5 = 0.343;
  lambda6 = 0.500;
  lambda = [lambda1 lambda2 lambda3 lambda4 lambda5 lambda6]';
%
%-----------------------------------------------------------------------------------------------------
% Initializing optimization procedure parameters
%
   %grad_p = 0.043;              % gradient step for parameters,
   grad_p = 1.0;                 % gradient step for parameters (original value),
   grad_lambda = 1.0;            %
   e_p = 0.001;                        % epsilon for parameters
   e_lambda = 0.001;                   % epsilon for lambda (Lagrange multiplier)
%
%-------------------------------------------------------------------------------------------------
% Initialize process parameter, control input, disturbance and state vectors full
      trajectories
  p = p0*ones(1,K);
  FR = FR*ones(1,K);
  states_in = conc_in*ones(1,K);
  states = states_bar;                  % declaring current state variables based on
                                        measured states
  lambdak = lambda*ones(1,K);
%-------------------------------------------------------------------------------------------------
% Iteration procedure start and initializing index of iteration (j)
  j = 1;
  e0 = cputime;
  while j <= M                         % beginning of iteration M
%
% Solving the state model equation starting with initial values of parameters
%
  for k = 1:K
     FRk = FR(k);                       % process control input, FR(k)
     statesk_in = states_in(k);         % process disturbances, xf(k)
     statesk = states(:,k);             % states(k) = x(k), the process states
     lambdak1 = lambdak(:,k);           % Lagrange multipliers
     %
     % Collect values for k points, se we need to use vectors
```

```
      La1_p(:,k) = dt*lambdak1(1)*[(statesk_in - statesk(1)) statesk(2)*FRk -
                         statesk(1)*FRk FRk]';
      La2_p(:,k) = dt*lambdak1(2)*[(statesk(1) - statesk(2)) statesk(3)*FRk -
                         statesk(2)*FRk FRk]';
      La3_p(:,k) = dt*lambdak1(3)*[(statesk(2) - statesk(3)) statesk(4)*FRk -
                         statesk(3)*FRk FRk]';
      La4_p(:,k) = dt*lambdak1(4)*[(statesk(3) - statesk(4)) statesk(5)*FRk -
                         statesk(4)*FRk FRk]';
      La5_p(:,k) = dt*lambdak1(5)*[(statesk(4) - statesk(5)) statesk(6)*FRk -
                         statesk(5)*FRk FRk]';
      La6_p(:,k) = dt*lambdak1(6)*[(statesk(5) - statesk(6)) -statesk(6)*FRk FRk]';
    %
    % Formulate the full La_p vector
     La_p(:,k) = [La1_p(:,k);La2_p(:,k);La3_p(:,k);La4_p(:,k);La5_p(:,k);La6_p(:,k)];
    %
    % Test parameters gradients for stopping iteration
    %
    if abs(La_p(:,k)) <= e_p | j >= M
       p(:,k) = p(:,k);
    else
       p(:,k) = p(:,k) - grad_p*La_p(:,k);
    end
  end
%
  for k = 1:K-1
      FRk = FR(k);                    % process control input, FR(k)
      statesk_in = states_in(k);      % process disturbances, xf(k)
      statesk = states(:,k);          % states(k) = x(k), the process states
      statesk1 = states(:,k+1);       % states(k+1) = x(k+1), the process state
    %
    % Determining Lagrange multipliers gradients
     La_lambda(:,k) = (-statesk1 + dt*(A*statesk + B1*statesk*FRk + B*FRk +
                            W*statesk_in));
    %
    % Test condition for stoping iteration for Lagrange multipliers
    %
     if abs(La_lambda) <= e_lambda | j >= M
        lambda = lambda;
     else
       lambda = lambda + grad_lambda*La_lambda(:,k);
    end
   end
%
  j = j+1;
  x0 = statesk(:,1);
  for k = 1:K-1
      FRk = FR(k);
      statesk = states(:,k);
      statesk_in = states_in(k);
      est_statesk(:,k+1) = curr_states(statesk,FRk,statesk_in,x0,p,dt);
  end
%
  end
%-------------------------------------------------------------------------------------------
% Display optimal parameters calculated
   Jk1 = 0;
   for i = 1:K
     error(:,i) = states_bar(:,i)-est_statesk(:,i);
     Jk = (error(:,i)'*error(:,i));
     Jk1 = Jk1 + Jk;
   end
 %
   error = abs(error);
   e1 = cputime;
   e = e1 - e0
   Jk1
   %t = inv(dt/2.35)
%
%    j = j
%  p_final = p(:,K)
%
%---------------------------------------------------------------------------------------------
%
 figure(1)
 i = 1:K;
 plot(i,states_bar,':o','Linewidth',2.5,'MarkerSize',5)
```

```
text('Position',[15.0 0.35],'String','x_1','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[25.0 0.45],'String','x_2','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[35.0 0.5],'String','x_3','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[45.0 0.65],'String','x_4','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[55.0 0.70],'String','x_5','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[65.0 0.82],'String','x_6','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
xlabel('Discrete time [k]','Fontsize',14)
ylabel('Process true state [eq/l]','Fontsize',14)
title('Dynamic behaviour of the process','Fontsize',14)
grid on
%
  figure(2)
  subplot(2,1,1)
  i = 1:K;
  plot(i,states_bar,':o','Linewidth',2.5,'MarkerSize',5)
  hold on
  grid on
  text('Position',[15.0 0.35],'String','x_1','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[25.0 0.45],'String','x_2','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[35.0 0.5],'String','x_3','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[45.0 0.65],'String','x_4','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[55.0 0.70],'String','x_5','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  text('Position',[65.0 0.82],'String','x_6','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
  xlabel('Discrete time [k]','Fontsize',14)
  ylabel('Process true state [eq/l]','Fontsize',14)
  title('Dynamic behaviour of the process','Fontsize',14)
```

```
%
   subplot(2,1,2)
   i = 1:K;
   plot(i,est_statesk,'-o','Linewidth',2.5,'MarkerSize',5)
   hold on
   text('Position',[15.0 0.35],'String','x^~_1','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
   text('Position',[25.0 0.45],'String','x^~_2','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
   text('Position',[35.0 0.5],'String','x^~_3','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
   text('Position',[45.0 0.65],'String','x^~_4','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
   text('Position',[55.0 0.70],'String','x^~_5','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
   text('Position',[65.0 0.82],'String','x^~_6','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
   xlabel('Discrete time [k]','Fontsize',14)
   ylabel('Estimated state [eq/l]','Fontsize',14)
   title('Dynamic behaviour of the process','Fontsize',14)
   grid on
   hold off
 %
    figure(3)
    i = 1:K;
    plot(i,error,':o','Linewidth',2.5,'MarkerSize',5)
    text('Position',[10.0 0.1],'String','e_x_1 - e_x_6','FontName','Times New Roman',
'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
    hold on
  %
    plot(i,states_bar(1,:),'-ob','Linewidth',2.5,'MarkerSize',5)
    text('Position',[15.0 0.35],'String','x_1','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
    hold on
    plot(i,states_bar(2,:),'-ob','Linewidth',2.5,'MarkerSize',5)
    text('Position',[25.0 0.45],'String','x_2','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
    hold on
    plot(i,states_bar(3,:),'-ob','Linewidth',2.5,'MarkerSize',5)
    text('Position',[35.0 0.5],'String','x_3','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
    hold on
    plot(i,states_bar(4,:),'-ob','Linewidth',2.5,'MarkerSize',5)
    text('Position',[45.0 0.65],'String','x_4','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
    hold on
    plot(i,states_bar(5,:),'-ob','Linewidth',2.5,'MarkerSize',5)
    text('Position',[55.0 0.70],'String','x_5','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
    hold on
    plot(i,states_bar(6,:),'-ob','Linewidth',2.5,'MarkerSize',5)
    text('Position',[65.0 0.82],'String','x_6','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
    hold on
    xlabel('Discrete time [k]','Fontsize',16)
    ylabel('Estimated states & error difference [eq/l]','Fontsize',16)
    title('System error & dynamic behaviour based on estimated model','Fontsize',16)
    grid on
    hold off
%
    figure(4)
    k = 1:K;
    plot(k,error,':o','Linewidth',2.5,'MarkerSize',5)
    text('Position',[10.0 1.0],'String','e_x_6','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
    text('Position',[50.0 0.05],'String','e_x_1 - e_x_6','FontName','Times New Roman',
         'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
    grid on
    xlabel('Discrete time [k]','Fontsize',16)
    ylabel('Error between real & measured states','Fontsize',16)
    title('Error difference between real and model states of the process','Fontsize',16)
%
   figure(5)
   k = 1:K;
   plot(k,p,'o','Linewidth',2.5,'MarkerSize',5)
```

```
  grid on
  xlabel('Time [k]','Fontsize',16)
  ylabel('Estimated parameter values','Fontsize',16)
  title('Parameter values over full process trajectory','Fontsize',16)
 %
  figure(6)
  k = 1:K;
  subplot(3,2,1)
  plot(k,est_statesk(1,:),':')
  hold on
  plot(k,states_bar(1,:))
  text('Position',[30.0 0.13],'String','x1bar','Fontsize',12)
  text('Position',[60.0 0.20],'String','x^~_1','Fontsize',12)
  xlabel('Discrete time [k]')
  ylabel('Real & estimated states[eq/l]')
  title('Dynamic behaviour of the process')
  hold off
%
  subplot(3,2,2)
  plot(k,est_statesk(2,:),':')
  hold on
  plot(k,states_bar(2,:))
  text('Position',[60.0 0.15],'String','x2bar','Fontsize',12)
  text('Position',[30.0 0.70],'String','x^~_2','Fontsize',12)
  xlabel('Discrete time [k]')
  ylabel('Real & estimated states[eq/l]')
  title('Dynamic behaviour of the process')
  hold off
%
  subplot(3,2,3)
  plot(k,est_statesk(3,:),':')
  hold on
  plot(k,states_bar(3,:))
  text('Position',[60.0 0.25],'String','x3bar','Fontsize',12)
  text('Position',[40.0 0.10],'String','x^~_3','Fontsize',12)
  xlabel('Discrete time [k]')
  ylabel('Real & estimated states[eq/l]')
  title('Dynamic behaviour of the process')
  hold off
%
  subplot(3,2,4)
  plot(k,est_statesk(4,:),':')
  hold on
  plot(k,states_bar(4,:))
  text('Position',[80.0 0.15],'String','x4bar','Fontsize',12)
  text('Position',[50.0 0.75],'String','x^~_4','Fontsize',12)
  xlabel('Discrete time [k]')
  ylabel('Real & estimated states[eq/l]')
  title('Dynamic behaviour of the process')
  hold off
%
  subplot(3,2,5)
  plot(k,est_statesk(5,:),':')
  hold on
  plot(k,states_bar(5,:))
  text('Position',[60.0 0.70],'String','x5bar','Fontsize',12)
  text('Position',[40.0 0.50],'String','x^~_5','Fontsize',12)
  xlabel('Discrete time [k]')
  ylabel('Real & estimated states[eq/l]')
  title('Dynamic behaviour of the process')
  hold off
%
  subplot(3,2,6)
  plot(k,est_statesk(6,:),':')
  hold on
  plot(k,states_bar(6,:))
  text('Position',[20.0 0.55],'String','x^~_6','Fontsize',12)
  text('Position',[60.0 0.65],'String','x6bar','Fontsize',12)
  xlabel('Discrete time [k]')
  ylabel('Real & estimated states[eq/l]')
  title('Dynamic behaviour of the process')
  hold off
%
%----------------------------------------------Program -ends------------------------------

% -----------------------------------------------------------------------------------------------
% The function is used to calculate the values of the states for parameter estimation,
% the considered trajectory period is (K = 100).
% The function is called in the main program under the function name iestate()
% The main program is called "Parameter_Estimation6a_Lagrang2.m"
%
% Author: NM Dube
% Dated: 23 June 2011
% Updated: 15 December 2016
%-------------------------------------------------------------------------------------------
% declare the function iestate()
function est_statesk = curr_states(statesk,FRk,statesk_in,x0,p,dt)
%
% solving the model state equation using the current parameter values
 x1 = statesk(1) + dt*(p(1)*statesk_in - p(1)*statesk(1) + p(2)*FRk*statesk(2) -
        p(3)*FRk*statesk(1) + p(4)*FRk);
 x2 = statesk(2) + dt*(p(5)*statesk(1) - p(5)*statesk(2) + p(6)*FRk*statesk(3) -
        p(7)*FRk*statesk(2) + p(8)*FRk);
 x3 = statesk(3) + dt*(p(9)*statesk(2) - p(9)*statesk(3) + p(10)*FRk*statesk(4) -
        p(11)*FRk*statesk(3) + p(12)*FRk);
 x4 = statesk(4) + dt*(p(13)*statesk(3) - p(13)*statesk(4) + p(14)*FRk*statesk(5) -
        p(15)*FRk*statesk(4) + p(16)*FRk);
 x5 = statesk(5) + dt*(p(17)*statesk(4) - p(17)*statesk(5) + p(18)*FRk*statesk(6) -
        p(19)*FRk*statesk(5) + p(20)*FRk);
 x6 = statesk(6) + dt*(p(21)*statesk(5) - p(21)*statesk(6) + 0 - p(22)*FRk*statesk(6) +
        p(23)*FRk);
%--------------------------------------------------------------------------------------------
% declaring the state variables for the state vector at k+1, k=1:K
 est_statesk = [x1 x2 x3 x4 x5 x6]';
%
%------------------------------------------------Program   ends--------------------------------
```

APPENDIX C: PROGRAMS FOR PARAMETER ESTIMATION OF THE NONLINEAR MODEL (BASED ON THE SEPARATION FACTOR)

APPENDIX C.1: Program for nonlinear parameter estimation – the direct method using MATLAB fsolve() function

```
% ----------------------------------------------------------------------------------------------------
% Program for parameter estimation of CCIX (Continuous Countercurrent Ion
Exchange)
% using separation factor (Nonlinearn Model).
% The model is based on the six stages UCT (University of Cape Town) model,
% and the current project has 6 stages.
% In this program state is the state variables and state_bar is the measured states.
% The program uses direct method of function fsolve() to solve unknown parameters
% of the separation factor, alpha01.
%
% Author: N. Dube
% Dated: 10 August 2012
% Updated: 21 November 2016
%
%----------------------------------------------------------------------------------------------------
% Original data measured  from UCT (University of Cape Town) project in 1982
% Stage H+ fractional change in liquid concentration using data obtained from UCT
Project volume 4
% stage1 = [0.221 0.577 0.730 0.847 0.920 0.936 0.968 0.974 0.981 0.989 0.988
            0.997 1.000 1.000 1.000 1.000 1.000 1.000];
% stage2 = [0.000 0.140 0.314 0.523 0.656 0.766 0.842 0.886 0.900 0.933 0.958
            0.963 0.982 0.974 0.991 0.994 0.993 0.993];
% stage3 = [0.000 0.04 0.066 0.184 0.295 0.454 0.601 0.690 0.758 0.804 0.877
            0.881 0.951 0.965 0.972 0.981 1.000 0.988];
% stage4 = [0.000 0.000 0.004 0.035 0.082 0.168 0.277 0.361 0.440 0.522 0.698
            0.784 0.899 0.931 0.966 0.966 0.991 0.991];
% stage5 = [0.000 0.000 0.000 0.003 0.020 0.052 0.113 0.167 0.207 0.340 0.474
            0.547 0.780 0.860 0.899 0.905 0.975 0.973];
% stage6 = [0.000 0.000 0.000 0.000 0.000 0.001 0.024 0.033 0.063 0.124 0.167
            0.233 0.482 0.539 0.672 0.779 0.940 0.972];
%
% Obtaining Na+ fractional change in liquid concentration
% This data of Na+ is then plotted to obtain measured states as indicated in the table
of states_bar (see g1 m-file, measured data)
```

```
% stage1 = 1 - stage1;
% stage2 = 1 - stage2;
% stage3 = 1 - stage3;
% stage4 = 1 - stage4;
% stage5 = 1 - stage5;
% stage6 = 1 - stage6;
%
%-----------------------------------------------------------------------------------------------
%
  N = 6;
  K = 18;
  e0 = cputime;
%
   alpha00 = 0.0*[1.000 1.000 1.000 1.000 1.000 1.000];
  alpha00 = alpha00'*ones(1,K);
  options = optimset('Display','iter','MaxIter',10000,'MaxFunEvals',100000)
%
% options = optimset('MaxFunEvals',1000);                    % option to display
algorithm information
  alpha01 = fsolve(@g1,alpha00,options);
%
%---------------------------------------------------------------------------------------------
e1 = cputime;
etime = e1 - e0
%---------------------------------------------------------------------------------------------
 k = 1:K;
 figure(1)
 plot(k,alpha01,':o','Linewidth',2.5,'MarkerSize',5)
text('Position',[2.0 0.5],'String','\alpha_N_a_1','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[4.0 0.1],'String','\alpha_N_a_2','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[5.0 0.0]','String','\alpha_N_a_3','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[7.0 0.5],'String','\alpha_N_a_4','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[10.0 0.7],'String','\alpha_N_a_5','FontName','Times New Roman',
        'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[16.0 1.0]','String','\alpha_N_a_6','FontName','Times New Roman',
       'Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 xlabel('Discrete time [k]','Fontsize',16)
 ylabel('Liquid/resin separation factor [eq/l]','Fontsize',16)
 title('Nonlinear dynamic state of the process','Fontsize',16)
 grid on
 hold off
%
%------------------------------------------Program end-------------------------------------

%-------------------------------------------------------------------------------------------------
% Function for solving a nonlinear model using fsolve() function
%
% Dated: 10 August 2012
%
function falpha1 = g1(alpha01)
        %
        %-------------------------------------------------------------------------------------------
        % Initialization of mass balance equation parameters of the CCIX model based
            on UCT model data
        % Initializing all function coeffients and variables using model data
         %
         N = 6;                          % number of stages of the cation loading column
         h = 32.93;                   % resin holdups (units in litres)
         H = 42.809;                 % resin holdups (units in litres)
         Fl = 2000/60;              % liquid flow rate (units in litres per min)
         d = 2/3;                       % resin/liquid fractional balance (constant)
         T = 1.2*18;                   % liquid upflow time (units in min)
        %
         FR = (h*d)/T;                % resin flow rate (units in litres per min)
          conc_in = 0.436;        % sodium concentration at first stage of the column (eq/l)
                                          – process disturbance
         % conc_in = 0.055;
         % conc_in = 0.110;
         % conc_in = 0.435;
         % conc_in = 0.550;
         % conc_in = 0.755;
         % conc_in = 1.000;
        %-------------------------------------------------------------------------------------------------
        % Introducing measurement vector expanded for the full state trajectory, stage
2 data has been corrected to fit the profile
states_bar = [1.0000 1.0000 0.9960 0.9650 0.9180 0.8320 0.7530 0.690 0.5800...
            0.5100 0.43020 0.3960 0.31010 0.2690 0.20340 0.1940 0.1120 0.0500;
%
            1.0000 0.8600 0.6860 0.4770 0.3440 0.2340 0.1580 0.1140 0.1000...
            0.0670 0.0420 0.0370 0.0180 0.0260 0.0090 0.0060 0.0070 0.0070;
%
            1.0000 1.0000 0.9340 0.8160 0.7050 0.5460 0.3990 0.3100 0.2420...
            0.1960 0.1230 0.1190 0.0490 0.0350 0.0280 0.0190 0.0120 0.0100;
%
            1.0000 1.0000 0.9960 0.9650 0.9180 0.8320 0.7230 0.6390 0.5600...
            0.4780 0.3020 0.2160 0.1010 0.0690 0.0340 0.0340 0.0090 0.0090;
%
            1.0000 1.0000 1.0000 0.9970 0.9800 0.9480 0.8870 0.8330 0.7930...
            0.6600 0.5260 0.4530 0.2200 0.1400 0.1010 0.0950 0.0250 0.0270;
%
            1.0000 1.0000 0.9960 0.9650 0.9180 0.8320 0.7230 0.6390 0.5600...
            0.4780 0.3020 0.2160 0.1010 0.0690 0.0340 0.0340 0.0090 0.0090];
%
%-------------------------------------------------------------------------------------------------
        % Initialize disturbance and control vectors full system trajectories
        %
         K = 18;
         u = FR*ones(1,K);
         states_in = conc_in;
         states = states_bar;                % declaring current state variables based on
measured states
        %
        %--------------------------------------------------------------------------------------------
        %
          dt = 2.7;                              % sampling period in time in (minutes)
        %
        for n = 1:N
          for k = 1:K-1
              %
```

```
             statesnk = states(n,k);
             statesnk1 = states(n,k+1);
             alpha_Na1nk = alpha01(n,k);
             %
             if n == N
                alpha01(n+1,k) = 0;
                states(n+1,k) = 0;
            end
            if n < N
                statesn1k = states(n+1,k);
                alpha_Na1n1k =  alpha01(n+1,k);
            end
            %
            %calculate subfunctions f1, f2 & f3
            %
            num1 = dt*((alpha_Na1nk + (statesnk -
statesnk*alpha_Na1nk))*(alpha_Na1nk + (statesnk - statesnk*alpha_Na1nk)));
            den1 = H*(alpha_Na1nk + (statesnk - statesnk*alpha_Na1nk)) +
h*alpha_Na1nk;
            f1 = num1/den1;
            %
            num2 = dt* 1;
            den2 = alpha_Na1nk + (statesnk - statesnk*alpha_Na1nk);
            f2 = num2/den2;
            %
            if n == N
                f3 = 0;
            else
                num3 = dt*1;
                den3 = alpha_Na1n1k + (statesn1k - statesn1k*alpha_Na1n1k);
                f3 = (num3/den3);
             end
            %
            %-------------------------------------------------------------------------------------------
            %
             if n == 1
                falpha1nk = -statesnk1 + (statesnk + f1*Fl*states_in - f1*Fl*statesnk +
f1*f3*FR*statesnk1 - f1*f2*FR*statesnk);
```

```
               else
                   statesnm1k = states(n-1,k);
                   falpha1nk = -statesnk1 + (statesnk + f1*Fl*statesnm1k - f1*Fl*statesnk +
f1*f3*FR*statesnk1 - f1*f2*FR*statesnk);
               end
              %
              falpha1(n,k) = falpha1nk;
          end
       end
%
%-------------------------------------------Function end--------------------------------------
```

APPENDIX D: PROGRAMS FOR STATE ESTIMATION METHOD USING DESIGN OF THE BILINEAR OBSERVER

APPENDIX D.1: Program for bilinear observer design

```
%----------------------------------------------------------------------------------------------------
% The Program is used for determining the observer determinant as required for
% determination of the observer matrix.
% Proper selection of observer matrix allows error, e(t) ~ 0, as t approaches infinity
%
% Author: NM. Dube
% Dated: 15 December 2011
% Updated: 03 September 2016
%
%----------------------------------------------------------------------------------------------------
% Initialization of mass balance equations of the CCIX model
%
h = 32.93;           % resin hold ups (units in litres)
H = 42.809;          % liquid holdups (units in litres)
Fl = 2000;           % liquid flow rate (units in litres per hour)
N = 6;               % number of stages of the process
d = 2/3;             % resin/liquid fractional balance (constant)
T = 17/60;           % cycle period (units in hours)
%
u = h*d/T;           % resin flow rate (units in litres per hour)
%u = 0.25;           % using normalized value of u;
%
a = [1.0 0.8 0.6 0.4 0.3 0.1];
b = [0.01 0.05 0.10 0.20 0.30 0.35];
%
%----------------------------------------------------------------------------------------------------
% Calculation of model parameters to use in the matrices above, li, mi and mij
%
for i = 1:N
   l(i) = Fl/(H+a(i)*h);
   mi(i) = a(i)/(H+a(i)*h);
end

for i = 1:N-1
   k(i) = (b(i+1) - b(i))/(H+a(i)*h);
   mij(i) = a(i+1)/(H+a(i)*h);
```

```
end
    k(N) = -b(N)/(H+a(N)*h);
%
%------------------------------------------------------------------------------------------------------
%Formation of model matrices A, B, B1, C, W and
%
A(1,1) = -l(1);
A(2,1) =  l(2);
A(2,2) = -l(2);
A(3,2) =  l(3);
A(3,3) = -l(3);
A(4,3) =  l(4);
A(4,4) = -l(4);
A(5,4) =  l(5);
A(5,5) = -l(5);
A(6,5) =  l(6);
A(6,6) = -l(6);
%
B(1,1) = -mi(1);
B(1,2) = mij(1);
B(2,2) = -mi(2);
B(2,3) = mij(2);
B(3,3) = -mi(3);
B(3,4) = mij(3);
B(4,4) = -mi(4);
B(4,5) = mij(4);
B(5,5) = -mi(5);
B(5,6) = mij(5);
B(6,6) = -mi(6);
%
B1 = [k(1) k(2) k(3) k(4) k(5) k(6)]';
%B1 = [3.0 5.0 2.0 1.0 3.0 6.0]';
%
W = [l(1) 0 0 0 0 0]';
%
C = [0 0 0 0 0 1];
%
%--------------------------------------------------------------------------------------------------
```

```
% Initialization of determinant elements
%A
%B
%B1
%
%-----------------------------------------------------------------------------------------------
%
a11 = A(1,1);
a21 = A(2,1);
a22 = A(2,2);
a32 = A(3,2);
a33 = A(3,3);
a43 = A(4,3);
a44 = A(4,3);
a54 = A(5,4);
a55 = A(5,5);
a65 = A(6,5);
a66 = A(6,6);
%
b11 = B(1,1);
b12 = B(1,2);
b22 = B(2,2);
b23 = B(2,3);
b33 = B(3,3);
b34 = B(3,4);
b44 = B(4,4);
b45 = B(4,5);
b55 = B(5,5);
b56 = B(5,6);
b66 = B(6,6);
%
%----------------------------------------------------------------------------------------------
% Initializing all values of observer matrix elements to 1 to obtain characteristic
equation solution.
%
l1 = sym('l1');
l2 = sym('l2');
l3 = sym('l3');
```

```
l4 = sym('l4');
l5 = sym('l5');
l6 = sym('l6');
%
%-----------------------------------------------------------------------------------------------
% Formation of the characteristic equation solution based on determinant of the
  observer state matrix Aobs
%
digits (6)
%
det6 = vpa(l1*a21*a32*a43*a54*a65);
det6 = [0 0 0 0 0 0 det6];
det5 = [0 0 0 0 0 0 0];
det4 = [0 0 0 0 0 0 0];
det3 = [0 0 0 0 0 0 0];
%
det2a = vpa(a21*a66 + a21*l6 - a21*u*b66 + a21*a55 - a21*u*b55 + a21*a44...
        - a21*u*b44 + a21*a33 - a21*u*b33);
%
det2b = vpa(a21*a55*a66 + a21*a55*l6 - a21*a55*u*b66...
                              - a21*u*a66 + a21*u*b55*l6 + a21*u*b55*u*b66...
                              + a21*a65*l5 + a21*a65*u*b56...
                              + a21*a44*a66 + a21*a44*l6 - a21*a44*u*b66...
                              + a21*a44*a55 + a21*a44*u*b55...
                              - a21*u*b44*a55 + a21*u*b55 + a21*u*b45*a54...
                              + a21*a33*a66 + a21*a33*l6 - a21*a33*u*b66...
                              + a21*a33*a55 - a21*a33*u*b55...
                              + a21*a33*a44 - a21*a33*u*b44...
                              - a21*u*b33*a66 - a21*u*b33*l6 + a21*u*b33*u*b66...
                              - a21*u*b33*a55 + a21*u*b33*u*b55...
                              - a21*u*b33*a44 + a21*u*b33*u*b44 + a21*u*b34*a43);
%
det2c = vpa(a21*a44*a55*a66 + a21*a44*a55*l6 - a21*a44*a55*u*b66...
                    - a21*a44*u*b55*a66 - a21*a44*u*b55*l6 +
a21*a44*u*b55*u*b66...
                    + a21*a44*a65*l5 + a21*a44*a65*u*b56...
                    - a21*u*b44*a55*a66 - a21*u*b44*a55*l6 +
a21*u*b44*a55*u*b66...
```

```
                    + a21*u*b44*u*b55*a66 + a21*u*b44*u*b55*l6 -
a21*u*b44*u*b55*u*b66...
                    - a21*u*b44*a65*l5 - a21*u*b44*a65*u*b56...
                    + a21*u*b45*a54*a66 + a21*u*b45*a54*l6 -
a21*u*b45*a54*u*b66...
                    + a21*l4*a54*a65...
                    + a21*a33*a55*a66 + a21*a33*a55*l6 - a21*a33*a55*u*b66...
                    - a21*a33*u*b55*a66 - a21*a33*u*b55*l6 +
a21*a33*u*b55*u*b66...
                    + a21*a33*a65 + a21*a33*a65*u*b56...
                    + a21*a33*a44*a66 + a21*a33*a44*l6 - a21*a33*a44*u*b66...
                    + a21*a33*a44*a55 - a21*a33*a44*u*b55...
                    - a21*a33*u*b44*a66 - a21*a33*u*b44*l6 +
a21*a33*u*b44*u*b66...
                    - a21*a33*u*b44*a55 + a21*a33*u*b44*u*b55 +
a21*a33*u*b45*a54...
                    - a21*u*b33*a44*a66 - a21*u*b33*a55*l6 +
a21*u*b33*a55*u*b66...
                    + a21*u*b33*a65*l5 - a21*u*b33*u*b56...
                    - a21*u*b33*u*a44*a66 - a21*u*b33*a44*l6 +
a21*u*b33*a44*u*b66...
                    - a21*u*b33*a44*a55 + a21*u*a44*u*b55...
                    + a21*u*b33*u*b44*b44*a66 + a21*u*b33*u*b44*l6 -
a21*u*b33*u*b44*u*b66...
                    + a21*u*b33*u*b44*a55 - a21*u*b33*u*b44*b55 -
a21*u*b33*u*b45*a54...
                    + a21*u*b34*a43*a66 + a21*u*b34*a43*l6 -
a21*u*b34*a43*u*b66...
                    + a21*u*b34*a43*a55 - a21*u*b34*a43*u*b55);
%
det2d = vpa(a21*a33*a44*a55*a66 + a21*a33*a44*a55*l6 - a21*a33*a44*u*b66...
            - a21*a33*a44*u*b55*a66 - a21*a33*a44*u*b55*l6 +
a21*a33*a44*u*b55*u*b66...
            + a21*a33*a44*a65*l5 - a21*a33*a44*u*b56...
            - a21*a33*u*b44*a55*a66 - a21*a33*u*b44*a55*l6 +
a21*a33*u*b44*a55*u*b66...
            + a21*a33*u*b44*u*b55*a66 + a21*a33*u*b44*u*b55*l6 -
a21*a33*u*b44*u*b55*u*b66...
```

```
                 - a21*a33*u*b44*a65*l5 - a21*a33*u*b44*a65*u*b56...
                 + a21*a33*u*b45*a54*a66 + a21*a33*u*b45*a54*l6 -
a21*a33*u*b45*a54*u*b66...
                 + a21*a33*l4*a54*a65...
                 - a21*u*b33*a44*a55*a66 - a21*u*b33*a44*a55*l6 +
a21*u*b33*a44*a55*u*b66...
                 + a21*u*b33*a44*u*b55*a66 + a21*u*b33*a44*u*b55*l6 -
a21*u*b33*a44*u*b55*u*b66...
                 - a21*u*b33*a44*a65*l5 - a21*u*b33*a44*a65*u*b66...
                 + a21*u*b33*u*b44*a55*a66 - a21*u*b33*u*b44*a55*l6 -
a21*u*b33*u*b44*a55*u*b66...
                 - a21*u*b33*u*b44*u*b55*a66 - a21*u*b33*b44*u*b55*l6 +
a21*u*b33*u*b44*u*b55*u*b66...
                 + a21*u*b33*u*b44*a65*l5 + a21*u*b33*u*b44*a65*u*b56...
                 - a21*u*b33*u*b45*a54*a66 - a21*u*b33*b45*a54*l6 +
a21*u*b45*a54*u*b66...
                 - a21*u*b33*l4*a54*a65...
                 + a21*u*b34*a43*a55*a66 + a21*u*b34*a43*a55*l6 -
a21*u*b34*a43*a55*u*b66...
                 - a21*u*b34*a43*u*b55*a66 - a21*u*b34*a43*u*b55*l6 +
a21*u*b34*a43*u*b55*u*b66...
                 + a21*u*b34*a43*a65*l5 + a21*u*b34*a43*a65*u*b56 +
a21*l3*a43*a54*a65);
det2 = u*b12*[0 0 0 det2a det2b det2c det2d];
%
det1b = vpa(a66 + l6 - u*b66 + a55 - u*b55 + a44 - u*b44 + a33 - u*b33 + a22 +
u*b22 + a11 - u*b11);
%
det1c = vpa(a55*a66 + a55*l6 - a55*u*b66...
                    - u*b55*a66 - u*b55*l6 + u*b55*u*b66...
                    + a65*l5 + a65*u*b56...
                    + a44*a66 + a44*l6 - a44*u*b66...
                    + a44*a55 + a44*u*b55...
                    - u*b44*a66 - u*b44*l6 + u*b44*u*b66...
                    - u*b44*a55 + u*b44*u*b55 + u*b45*a54...
                    + a33*a66 + a33*l6 - a33*u*b66...
                    + a33*a55 - a33*u*b55 + a33*a44 - a33*u*b44...
                    - u*b33*a66 - u*b33*l6 + u*b33*u*b66...
                    - u*b33*a55 + u*b33*u*b55 - u*b33*a44 + u*b33*u*b44...
                    + u*b34*a43...
                    + a22*a66 + a22*l6 + a22*u*b66...
                    + a22*a55 - a22*u*b55 + a22*u*a44 - a22*u*b44...
                    + a22*a33 - a22*u*b33...
                    - u*b22*a66 - u*b22*l6 + u*b22*u*b66...
                    - u*b22*a55 + u*b22*u*b55 - u*b22*a44 + u*b22*u*b44...
                    - u*b22*a33 + u*b22*u*b33...
                    + u*b23*a32...
                    + a11*a66 + a11*l6 - a11*u*b66...
                    + a11*a55 - a11*u*b55 + a11*a44 - a11*u*b44...
                    + a11*a33 - a11*u*b33 + a11*a22 - a11*u*b22...
                    - u*b11*a66 - u*b11*l6 + u*b11*u*b66...
                    - u*b11*a55 + u*b11*u*b55 - u*b11*a44 + u*b11*u*b44...
                    - u*b11*a33 + u*b11*u*b33 - u*b11*a22 + u*b11*u*b22);
   %
   det1d = vpa(a44*a55*a66 + a44*a55*l6 - a44*a55*u*b66...
           - a44*u*b55*a66 - a44*u*b55*l6 + a44*u*b55*u*b66...
           + a44*a65*l5 + a44*a65*u*b56...
           - u*b44*a55*a66 - u*b44*a55*l6 + u*b44*u*a55*u*b66...
           + u*b44*u*b55*a66 + u*b44*u*b55*l6 - u*b44*u*b55*u*b66...
           - u*b44*a65*l5 - u*b44*a65*u*b56...
           + u*b45*a54*a66 + u*b45*a54*l6 - u*b45*a54*u*b66...
           + l4*a54*a65...
           + a33*a55*a66 + a33*a55*l6 - a33*a55*u*b66...
           - a33*u*b55*a66 - a33*u*b55*l6 + a33*u*b55*u*b66...
           + a33*a65*l5 + a33*a65*u*b56...
           + a33*a44*a66 + a33*a44*l6 - a33*a44*u*b66...
           + a33*a44*a55 - a33*a44*u*b55...
           - a33*u*b44*a66 - a33*u*b44*l6 - a33*u*b44*u*b66...
           - a33*u*b44*a55 + a33*u*b44*u*b55 + a33*u*b45*a54...
           - u*b33*a55*a66 - u*b33*a55*l6 + u*b33*a55*u*b66...
           + u*b33*u*b55*a66 + u*b33*u*b55*l6 - u*b33*u*b55*u*b66...
           - u*b33*a65*l5 - u*b33*a65*u*b56...
           - u*b33*a44*a66 - u*b33*a44*l6 + u*b33*a44*u*b66...
           - u*b33*a44*a55 + u*b33*a44*u*b55...
           + u*b33*u*b44*a44 + u*b33*u*b44*l6 - u*b33*u*b44*u*b66...
           + u*b33*u*b44*a55 - u*b33*u*b44*u*b55...
```

```
          - u*b33*u*b45*a54...
          + u*b34*a43*a66 + u*b34*a43*l6 - u*b34*a43*u*b66...
          + u*b34*a43*a55 - u*b34*a43*u*b55...
          + a22*a55*a66 + a22*a55*l6 - a22*a55*u*b66...
          - a22*u*b55*a66 - a22*u*b55*l6 + a22*u*b55*u*b66...
          + a22*a65*l5 + a22*a65*u*b56...
          + a22*a44*a66 + a22*a44*l6 - a22*a44*u*b66...
          + a22*a44*a55 - a22*a44*u*b55...
          - a22*u*b44*a66 - a22*u*b44*l6 + a22*u*b44*u*b66...
          - a22*u*b44*a55 + a22*u*b44*u*b55...
          + a22*u*b45*a54...
          + a22*a33*a66 + a22*a33*l6 - a22*a33*u*b66...
          + a22*a33*a55 - a22*a33*u*b55 + a22*a33*a44 - a22*a33*u*b44...
          - a22*u*b33*a66 - a22*u*b33*l6 + a22*u*b33*u*b66...
          - a22*u*b33*a55 + a22*u*b33*u*b55 - a22*u*b33*a44 + a22*u*b33*u*b44...
          + a22*u*b34*a43...
          - u*b22*a55*a66 - u*b22*a55*l6 + u*b22*a55*u*b66...
          + u*b22*u*b55*a66 + u*b22*u*b55*l6 - u*b22*u*b55*u*b66...
          - u*b22*a65*l5 - u*b22*a65*u*b56...
          - u*b22*a44*a66 - u*b22*a44*l6 + u*b22*a44*u*b66...
          - u*b22*a44*a55 + u*b22*a44*u*b55...
          + u*b22*u*b44*a66 + u*b22*u*b44*l6 - u*b22*u*b44*u*b66...
          + u*b22*u*b44*a55 - u*b22*u*b44*u*b55...
          - u*b22*u*b45*u*b55...
          - u*b22*a33*a66 - u*b22*a33*l6 + u*b22*a33*u*b66...
          - u*b22*a33*a55 + u*b22*a33*u*b55 - u*b22*a33*a44 + u*b22*a33*u*b44...
          + u*b22*u*b33*a66 + u*b22*u*b33*l6 - u*b22*u*b33*u*b66...
          + u*b22*u*b33*a55 - u*b22*u*b33*u*b55 + u*b22*u*b33*a44 -
u*b22*u*b33*u*b44...
          - u*b22*u*b34*a43...
          + u*b23*a32*a66 + u*b23*a32*l6 - u*b23*a32*u*b66...
          + u*b23*a32*a55 - u*b23*a32*u*b55 + u*b23*a32*a44 - u*b23*a32*u*b44...
          + a11*a55*a66 + a11*a55*l6 - a11*a55*u*b66...
          - a11*u*b55*a66 - a11*u*b55*l6 + a11*u*b55*u*b66...
          + a11*a65*l5 + a11*a65*u*b56...
          + a11*a44*a66 + a11*a44*l6 - a11*a44*u*b66...
          + a11*a44*a55 - a11*a44*u*b55...
          - a11*u*b44*a66 - a11*u*b44*l6 + a11*u*b44*u*b66...
          - a11*u*b44*a55 + a11*u*b44*u*b55...
          + a11*u*b45*a54...
          + a11*a33*a66 + a11*a33*l6 - a11*a33*u*b66...
          + a11*a33*a55 - a11*a33*u*b55 + a11*a33*a44 - a11*a33*u*b44...
          - a11*u*b33*a66 - a11*u*b33*l6 + a11*u*b33*u*b66...
          - a11*u*b33*a55 + a11*u*b33*u*b55 - a11*u*b33*a44 + a11*u*b33*u*b44...
          + a11*u*b34*a43...
          + a11*a22*a66 + a11*a22*l6 - a11*a22*u*b66...
          + a11*a22*a55 - a11*a22*u*b55 + a11*a22*a44 - a11*a22*u*b44...
          + a11*a22*a33 - a11*a22*u*b33...
          - a11*u*b22*a66 - a11*u*b22*l6 + a11*u*b22*u*b66...
          - a11*u*b22*a55 + a11*u*b22*u*b55 - a11*u*b22*a44 + a11*u*b22*u*b44...
          - a11*u*b22*a33 + a11*u*b22*u*b33...
          + a11*u*b23*a32...
          - u*b11*a55*a66 - u*b11*a55*l6 + u*b11*a55*u*b66...
          + u*b11*u*b55*a66 + u*b11*u*b55*l6 - u*b11*u*b55*u*b66...
          - u*b11*a65*l5 - u*b11*a65*u*b56...
          - u*b11*a44*a66 - u*b11*a44*l6 + u*b11*u*a44*u*b66...
          - u*b11*a44*a55 + u*b11*a44*u*b55...
          + u*b11*u*b44*a66 + u*b11*u*b44*l6 - u*b11*u*b44*u*b66...
          + u*b11*u*b44*a55 - u*b11*u*b44*u*b55...
          - u*b11*u*b45*a54...
          - u*b11*a33*a66 - u*b11*a33*l6 + u*b11*a33*u*b66...
          - u*b11*a33*a55 + u*b11*a33*u*b55 - u*b11*a33*a44 + u*b11*a33*u*b44...
          - u*b11*u*b34*a43...
          - u*b11*a22*a66 - u*b11*a22*l6 + u*b11*a22*u*b66...
          - u*b11*a22*a55 + u*b11*a22*u*b55 - u*b11*a22*a44 + u*b11*a22*u*b44...
          + u*b11*a22*a33 + u*b11*a22*u*b33...
          + u*b11*u*b22*a66 + u*b11*u*b22*l6 - u*b11*u*b22*u*b66...
          + u*b11*u*b22*a55 - u*b11*u*b22*u*b55 + u*b11*u*b22*a44 -
u*b11*u*b22*u*b44...
          + u*b11*u*b22*a33 - u*b11*u*b22*u*b33 - u*b11*u*b23*a32);
%
det1e = vpa(a33*a44*a55*a66 + a33*a44*a55*l6 - a33*a44*a55*u*b66...
                    - a33*a44*u*b55*a66 - a33*a44*u*b55*l6 +
a33*a44*u*b55*u*b66...
                    + a33*a44*a65*l5 + a33*a44*a65*u*b56...
                    - a33*u*b44*a55*a66 - a33*u*b44*a55*l6 +
a33*u*b44*a55*u*b66...
                    + a33*u*b44*u*b55*a66 + a33*u*b44*u*b55*l6 -
a33*u*b44*u*b55*u*b66...
                    - a33*u*b44*a65*l5 - a33*u*b44*a65*u*b56...
                    + a33*u*b45*a54*a66 + a33*u*b45*a54*l6 -
a33*u*b45*a54*u*b66...
                    + a33*l4*a54*a65...
                    - u*b33*a44*a55*a66 - u*b33*a44*a55*l6 +
u*b33*a44*a55*u*b66...
                    + u*b33*a44*u*b55*a66 + u*b33*a44*u*b55*l6 -
u*b33*a44*u*b55*u*b66...
                    - u*b33*a44*a65*l5 - u*b33*a44*a65*u*b56...
                    + u*b33*u*b44*a55*a66 + u*b33*u*b44*a55*l6 -
u*b33*u*b44*a55*u*b66...
                    - u*b33*u*b44*u*b55*a66 - u*b33*u*b44*u*b55*l6 +
u*b33*u*b44*u*b55*u*b66...
                    + u*b33*u*b44*a65*l5 + u*b33*u*b44*a65*u*b56...
                    - u*b33*u*b45*a54*a66 - u*b33*u*b45*a54*l6 +
u*b33*u*b45*a54*u*b66...
                    - u*b33*l4*a54*a65...
                    + u*b34*a43*a55*a66 + u*b34*a43*a55*l6 -
u*b34*a43*a55*u*b66...
                    - u*b34*a43*u*b55*a66 - u*b34*a43*u*b55*l6 +
u*b34*a43*u*b55*u*b66...
                    + u*b34*a43*a65*l5 + u*b34*a43*a65*u*b56...
                    + l3*a43*a54*a65...
                    + a22*a44*a55*a66 + a22*a44*a55*l6 - a22*a44*a55*u*b66...
                    - a22*a44*u*b55*a66 - a22*a44*u*b55*l6 +
a22*a44*u*b55*u*b66...
                    + a22*a44*u*a65*l5 + a22*a44*a65*u*b56...
                    - a22*u*b44*a55*a66 - a22*u*b44*a55*l6 +
a22*u*b44*a55*u*b66...
                    + a22*u*b44*u*b55*a66 + a22*u*b44*u*b55*l6 -
a22*u*b44*u*b55*u*b66...
                    - a22*u*b44*a65*l5 - a22*u*b44*a65*u*b56...
                    + a22*u*b45*a54*a66 + a22*u*b45*a54*l6 -
a22*u*b45*a54*u*b66...
```

```
                    + a22*l6*a54*a65...
                    + a22*a33*a55*a66 + a22*a33*a55*l6 - a22*a33*a55*u*b66...
                    - a22*a33*u*b55*a66 - a22*a33*u*b55*l6 +
a22*a33*u*b55*u*b66...
                    + a22*a33*a65*l5 + a22*a33*a65*u*b56...
                    + a22*a33*a44*a55 - a22*a33*a44*u*b55...
                    - a22*a33*u*b44*a66 - a22*a33*u*b44*l6 +
a22*a33*u*b44*u*b66...
                    - a22*a33*u*b44*a55 + a22*a33*u*b44*u*b55...
                    + a22*a33*u*b45*a54...
                    - a22*u*b33*a55*a66 - a22*u*b33*a55*l6 +
a22*u*b33*a55*u*b66...
                    + a22*u*b33*u*b55*a66 + a22*u*b33*u*b55*l6 -
a22*u*b33*u*b55*u*b66...
                    - a22*u*b33*a65*l5 - a22*u*b33*a65*u*b56...
                    - a22*u*b33*a44*a66 - a22*u*b33*a44*l6 +
a22*u*b33*a44*u*b66...
                    - a22*u*b33*a44*a55 + a22*u*b33*a44*u*b55...
                    + a22*u*b33*u*b44*a66 + a22*u*b33*u*b44*l6 -
a22*u*b33*u*b44*u*b66...
                    + a22*u*b33*u*b44*a55 - a22*u*b33*u*b44*u*b55...
                    - a22*u*b33*u*b45*a54...
                    + a22*u*b34*a43*a66 + a22*u*b34*a43*l6 -
a22*u*b34*a43*u*b66...
                    + a22*u*b34*a43*a55 - a22*u*b34*a43*u*b55...
                    - u*b22*a44*a55*a66 - u*b22*a44*a55*l6 +
u*b22*a44*a55*u*b66...
                    + u*b22*a44*u*b55*a66 - u*b22*a44*u*b55*l6 -
u*b22*a44*u*b55*u*b66...
                    - u*b22*a44*a65*l5 - u*b22*a44*a65*u*b56...
                    + u*b22*u*b44*a55*a66 + u*b22*u*b44*a55*l6 -
u*b22*b44*a55*u*b66...
                    - u*b22*u*b44*u*b55*a66 - u*b22*u*b44*u*b55*l6 +
u*b22*u*b44*u*b55*u*b66...
                    + u*b22*u*b44*a65*l5 + u*b22*u*b44*a65*u*b56...
                    - u*b22*u*b45*a54*a66 - u*b22*u*b45*a54*l6 +
u*b22*u*b45*a54*u*b66...
                    - u*b22*l4*a54*a65...
                    - u*b22*a33*a55*a66 - u*b22*a33*a55*l6 +
u*b22*a33*a55*u*b66...
                    + u*b22*a33*u*b55*a66 + u*b22*a33*u*b55*l6 -
u*b22*a33*u*b55*u*b66...
                    - u*b22*a33*a65*l5 - u*b22*a33*a65*u*b56...
                    - u*b22*a33*a44*a66 - u*b22*a33*a44*l6 +
u*b22*a33*a44*u*b66...
                    - u*b22*a33*a44*a55 + u*b22*a33*a44*u*b55...
                    + u*b22*a33*u*b44*a66 + u*b22*a33*u*b44*l6 -
u*b22*a33*u*b44*u*b66...
                    + u*b22*a33*u*b44*a55 - u*b22*a33*u*b44*u*b55...
                    - u*b22*a33*u*b45*a54...
                    + u*b22*u*b33*a55*a66 + u*b22*u*b33*a55*l6 -
u*b22*b33*a55*u*b66...
                    - u*b22*u*b33*u*b55*a66 - u*b22*u*b33*u*b55*l6 +
u*b22*u*b33*u*b55*u*b66...
                    + u*b22*u*b33*a65*l5 + u*b22*u*b33*a65*u*b56...
                    + u*b22*u*b33*a44*a66 + u*b22*u*b33*a44*l6 -
u*b22*u*b33*a44*u*b66...
                    + u*b22*u*b33*a44*a55 - u*b22*u*b33*a44*u*b55...
                    - u*b22*a33*u*b44*a66 + u*b22*a33*u*b44*l6 -
u*b22*a33*u*b44*u*b66...
                    + u*b22*a33*u*b44*a55 - u*b22*a33*u*b44*u*b55...
                    - u*b22*a33*u*b45*a54...
                    + u*b22*u*b33*a55*a66 + u*b22*u*b33*a55*l6 -
u*b22*u*b33*a55*u*b66...
                    - u*b22*u*b33*u*b55*a66 - u*b22*u*b33*u*b55*l6 +
u*b22*u*b33*u*b55*u*b66...
                    + u*b22*u*b33*a65*l5 + u*b22*u*b33*a65*u*b56...
                    + u*b22*u*b33*a44*a66 + u*b22*u*b33*a44*l6 -
u*b22*u*b33*a44*u*b66...
                    + u*b22*u*b33*a44*a55 - u*b22*u*b33*a44*u*b55...
                    - u*b22*u*b33*u*b44*a66 - u*b22*u*b33*u*b44*l6 +
u*b22*u*b33*u*b44*u*b66...
                    + u*b22*u*b33*u*b44*a55 - u*b22*u*b33*u*b44*u*b55...
                    + u*b22*u*b33*u*b45*a54...
                    - u*b22*u*b34*a43*a66 - u*b22*u*b34*a43*l6 +
u*b22*u*b34*a43*u*b66...
```

```
                        - u*b22*u*b34*a43*a55 + u*b22*u*b34*a43*u*b55...
                        + u*b23*a32*a55*a66 + u*b23*a32*a55*l6 -
u*b23*a32*a55*u*b66...
                        - u*b23*a32*u*b55*a66 - u*b23*a32*u*b55*l6 +
u*b23*a32*u*b55*u*b66...
                        + u*b23*a32*a65*l5 + u*b23*a32*a65*u*b56...
                        + u*b23*a32*a44*a66 + u*b23*a32*a44*l6 -
u*b23*a32*a44*u*b66...
                        + u*b23*a32*a44*a55 - u*b23*a32*a44*u*b55...
                        - u*b23*a32*u*b44*a66 - u*b23*a32*u*b44*l6 +
u*b23*a32*u*b44*u*b66...
                        + u*b23*a32*u*b44*a55 - u*b23*a32*u*b44*u*b55...
                        + u*b23*a32*u*b45*a54...
                        + a11*a44*a55*a66 + a11*a44*a55*l6 - a11*a44*a55*u*b66...
                        - a11*a44*u*b55*a66 - a11*a44*u*b55*l6 +
a11*a44*u*b55*u*b66...
                        + a11*a44*a65*l5 + a11*a44*a65*u*b56...
                        + a11*u*b44*a55*a66 + a11*u*b44*a55*l6 -
a11*u*b44*a55*u*b66...
                        + a11*u*b44*u*b55*a66 + a11*u*b44*u*b55*l6 -
a11*u*b44*u*b55*u*b66...
                        - a11*u*b44*a65*l5 - a11*u*b44*a65*u*b56...
                        + a11*u*b45*a54*a66 + a11*u*b45*a54*l6 -
a11*u*b45*a54*u*b66...
                        + a11*l4*a54*a65...
                        + a11*a33*a55*a66 + a11*a33*a55*l6 - a11*a33*a55*u*b66...
                        - a11*a33*u*b55*a66 - a11*a33*u*b55*l6 +
a11*a33*u*b55*u*b66...
                        + a11*a33*a65*l5 + a11*a33*a65*u*b56...
                        + a11*a33*a44*a66 + a11*a33*a44*l6 - a11*a33*a44*u*b66...
                        + a11*a33*a44*a55 - a11*a33*a44*u*b55...
                        - a11*a33*u*b44*a66 - a11*a33*u*b44*l6 +
a11*a33*u*b44*u*b66...
                        - a11*a33*u*b44*a55 + a11*a33*u*b44*u*b55...
                        + a11*a33*u*b45*a54...
                        - a11*u*b33*a55*a66 - a11*u*b33*a55*l6 +
a11*u*b33*a55*u*b66...
```

```
                         + a11*u*b33*u*b55*a66 + a11*u*b33*u*b55*l6 -
a11*u*b33*u*b55*u*b66...
                         - a11*u*b33*a65*l5 - a11*u*b33*a65*u*b56...
                         - a11*u*b33*a44*a66 - a11*u*b33*a44*l6 +
a11*u*b33*a44*u*b66...
                         - a11*u*b33*a44*a55 + a11*u*b33*a44*u*b55...
                         + a11*u*b33*u*b44*a66 + a11*u*b33*u*b44*l6 -
a11*u*b33*u*b44...
                         + a11*u*b33*u*b44*a55 - a11*u*b33*u*b44*u*b55...
                         - a11*u*b33*u*b45*a54...
                         + a11*u*b34*a43*a66 + a11*u*b34*a43*l6 -
a11*u*b34*a43*u*b66...
                         + a11*u*b34*a43*a55 - a11*u*b34*a43*u*b55...
                         + a11*a22*a55*a66 + a11*a22*a55*l6 - a11*a22*a55*u*b66...
                         - a11*a22*u*b55*a66 - a11*a22*u*b55*l6 +
a11*a22*u*b55*u*b66...
                         + a11*a22*a65*l5 + a11*a22*a65*u*b56...
                         + a11*a22*a44*a66 + a11*a22*a44*l6 - a11*a22*a44*u*b66...
                         + a11*a22*a44*a55 - a11*a22*a44*u*b55...
                         - a11*a22*u*b44*a66 - a11*a22*u*b44*l6 +
a11*a22*u*b44*u*b66...
                         + a11*a22*a33*a55 - a11*a22*a33*u*b55...
                         + a11*a22*a33*a44 - a11*a22*a33*u*b44...
                         - a11*a22*u*b33*a66 - a11*a22*u*b33*l6 +
a11*a22*u*b33*u*b66...
                         - a11*a22*u*b33*a55 + a11*a22*u*b33*u*b55...
                         - a11*a22*u*b33*a44 + a11*a22*u*b33*u*b44...
                         + a11*a22*u*b34*a43...
                         - a11*u*b22*a55*a66 - a11*u*b22*a55*l6 +
a11*u*b22*a55*u*b66...
                         + a11*u*b22*u*b55*a66 + a11*u*b22*u*b55*l6 -
a11*u*b22*u*b55*u*b66...
                         - a11*u*b22*a65*l5 - a11*u*b22*a65*u*b56...
                         - a11*u*b22*a44*a66 - a11*u*b22*a44*l6 +
a11*u*b22*a44*u*b66...
                         - a11*u*b22*a44*a55 + a11*u*b22*a44*u*b55...
                         + a11*u*b22*u*b44*a66 + a11*u*b22*u*b44*l6 -
a11*u*b22*u*b44*u*b66...
```

```
                    + a11*u*b22*u*b44*a55 - a11*u*b22*u*b55...
                    - a11*u*b22*u*b45*a54...
                    - a11*u*b22*a33*a66 - a11*u*b22*a33*l6 +
a11*u*b22*a33*u*b66...
                    - a11*u*b22*a33*a55 + a11*u*b22*a33*u*b55...
                    - a11*b22*a33*a44 + a11*u*b22*a33*u*b44...
                    + a11*u*b22*u*b33*a66 + a11*u*b22*u*b33*l6 -
a11*u*b22*u*b33*u*b66...
                    + a11*u*b22*u*b33*a55 - a11*u*b22*u*b33*u*b55...
                    + a11*u*b22*u*b33*a44 - a11*u*b22*u*b33*u*b44...
                    - a11*u*b22*u*b34*a43...
                    + a11*u*b23*a32*a66 + a11*u*b23*a32*l6 -
a11*u*b23*a32*u*b66...
                    + a11*u*b23*a32*a66*a55 - a11*u*b23*a32*u*b55...
                    + a11*u*b23*a32*a44 - a11*u*b23*a32*u*b44...
                    - u*b11*a44*a55*a66 - u*b11*a44*a55*l6 +
u*b11*a44*a55*u*b66...
                    + u*b11*a44*u*b55*a66 + u*b11*a44*u*b55*l6 -
u*b11*a44*u*b55*u*b66...
                    - u*b11*a44*a65*l5 - u*b11*a44*a65*u*b56...
                    + u*b11*u*b44*a55*a66 + u*b11*u*b44*a55*l6 -
u*b11*u*b44*a55*u*b66...
                    - u*b11*u*b44*u*b55*a66 - u*b11*u*b44*u*b55*l6 +
u*b11*u*b44*u*b55*u*b66...
                    + u*b11*u*b44*a65*l5 + u*b11*u*b44*a65*u*b56...
                    - u*b11*u*b45*a54*a66 - u*b11*u*b45*a54*l6 +
u*b11*u*b45*a54*u*b66...
                    - u*b11*l4*a54*a65...
                    - u*b11*a33*a55*a66 - u*b11*a33*a55*l6 +
u*b11*a33*a55*u*b66...
                    + u*b11*a33*u*b55*a66 + u*b11*a33*u*b55*l6 -
u*b11*a33*u*b55*u*b66...
                    - u*b11*a33*a65*l5 - u*b11*a33*a65*u*b56...
                    - u*b11*a33*a44*a66 - u*b11*a33*a44*l6 +
u*b11*a33*a44*u*b66...
                    - u*b11*a33*a44*a55 + u*b11*a33*a44*u*b55...
                    + u*b11*a33*u*b44*a66 + u*b11*a33*u*b44*l6 -
u*b11*a33*u*b44*u*b66...
```

```
                    - u*b11*a33*u*b44*a55 - u*b11*a33*u*b44*u*b55...
                    + u*b11*a33*u*b45*a54...
                    - u*b11*u*b33*a55*a66 + u*b11*u*b33*a55*l6 -
u*b11*u*b33*a55*u*b66...
                    - u*b11*u*b33*u*b55*a66 - u*b11*u*b33*u*b55*l6 +
u*b11*u*b33*u*b55*u*b66...
                    + u*b11*u*b33*a65*l5 + u*b11*u*b33*a65*u*b56...
                    + u*b11*u*b33*a44*a66 + u*b11*u*b33*a44*l6 -
u*b11*u*b33*a44*u*b66...
                    + u*b11*u*b33*a44*a55 - u*b11*u*b33*a44*u*b55...
                    - u*b11*u*b33*u*b44*a66 - u*b11*u*b33*u*b44*l6 +
u*b11*u*b33*u*b44*u*b66...
                    - u*b11*u*b33*u*b44*a55 + u*b11*u*b33*u*b44*u*b55...
                    + u*b11*u*b33*u*b45*a54...
                    + u*b11*u*b34*a43*a66 - u*b11*u*b34*a43*l6 +
u*b11*u*b34*a43*u*b66...
                    - u*b11*u*b34*a43*a55 + u*b11*u*b34*a43*u*b55...
                    - u*b11*a22*a55*a66 - u*b11*a22*a55*l6 +
u*b11*a22*a55*u*b66...
                    + u*b11*a22*u*b55*a66 - u*b11*a22*u*b55*l6 -
u*b11*a22*u*b55*u*b66...
                    - u*b11*a22*a65*l5 - u*b11*a22*a65*u*b56...
                    - u*b11*a22*a44*a66 - u*b11*a22*a44*l6 +
u*b11*a22*a44*u*b66...
                    - u*b11*a22*a44*a55 + u*b11*a22*a44*u*b55...
                    + u*b11*a22*u*b44*a66 + u*b11*a22*u*b44*l6 -
u*b11*a22*a44*u*b66...
                    + u*b11*a22*u*b44*a55 - u*b11*a22*u*b44*u*b55...
                    - u*b11*a22*u*b45*a54...
                    - u*b11*a22*a33*a66 - u*b11*a22*a33*l6 +
u*b11*a22*a33*u*b66...
                    - u*b11*a22*a33*a55 + u*b11*a22*a33*u*b55...
                    - u*b11*a22*a33*a44 + u*b11*a22*a33*u*b44...
                    + u*b11*a22*u*b33*a66 + u*b11*a22*u*b33*l6 -
u*b11*a22*u*b33*u*b66...
                    + u*b11*a22*u*b33*a55 - u*b11*a22*u*b33*u*b55...
                    + u*b11*a22*u*b33*a44 - u*b11*a22*u*b33*u*b44...
                    - u*b11*a22*u*b34*a43...
                    + u*b11*u*b22*a55*a66 + u*b11*u*b22*a55*l6 -
u*b11*u*b22*a55*u*b66...
                    - u*b11*u*b22*u*b55*a66 - u*b11*u*b22*u*b55*l6 +
u*b11*u*b22*u*b55*u*b66...
                    + u*b11*u*b22*a65*l5 + u*b11*u*b22*a65*u*b56...
                    + u*b11*u*b22*a44*a66 + u*b11*u*b22*a44*l6 -
u*b11*u*b22*a44*u*b66...
                    + u*b11*u*b22*a44*a55 - u*b11*u*b22*a44*u*b55...
                    - u*b11*u*b22*u*b44*a66 - u*b11*u*b22*u*b44*l6 +
u*b11*u*b22*u*b44*u*b66...
                    - u*b11*u*b22*u*b44*a55 + u*b11*u*b22*u*b44*u*b55...
                    + u*b11*u*b22*u*b45*a54...
                    + u*b11*u*b22*a33*a66 + u*b11*u*b22*a33*l6 -
u*b11*u*b22*a33*u*b66...
                    + u*b11*u*b22*a33*a55 - u*b11*u*b22*a33*u*b55...
                    + u*b11*u*b22*a33*a44 - u*b11*u*b22*a33*u*b44...
                    - u*b11*u*b22*u*b33*a66 - u*b11*b22*u*b33*l6 +
u*b11*u*b22*u*b33*u*b66...
                    - u*b11*u*b22*u*b33*a55 + u*b11*u*b22*u*b33*u*b55...
                    - u*b11*u*b22*u*b33*a44 + u*b11*u*b22*u*b33*u*b44...
                    + u*b11*u*b22*u*b34*a43...
                    - u*b11*u*b23*a32*a66 - u*b11*u*b23*a32*l6 +
u*b11*u*b23*a32*u*b66...
                    - u*b11*u*b23*a32*a55 + u*b11*u*b23*a32*u*b55...
                    - u*b11*u*b23*a32*a44 + u*b11*u*b23*a32*u*b44);
 %
 det1f = vpa(a22*a33*a44*a55*a66 + a22*a33*a44*a55*l6 -
a22*a33*a44*a55*u*b66...
         - a22*a33*a44*u*b55*a66 - a22*a33*a44*u*b55 +
a22*a33*a44*u*b55*u*b66...
         + a22*a33*a44*a65*l5 + a22*a33*a44*a65*u*b56...
         - a22*a33*u*b44*a55*a66 - a22*a33*u*b44*a55*l6 +
a22*a33*u*b44*a55*u*b66...
         + a22*a33*u*b44*u*b55*a66 + a22*a33*u*b44*u*b55*l6 -
a22*a33*u*b44*u*b55*u*b66...
         - a22*a33*u*b44*a65*l5 - a22*a33*u*b44*a65*u*b56...
         + a22*a33*u*b45*a54*a66 + a22*a33*u*b45*a54*l6 -
a22*a33*u*b45*a54*u*b66...
```

```
            + a22*a33*l4*a54*a65...
            - a22*u*b33*a44*a55*a66 - a22*u*b33*a44*a55*l6 +
a22*u*b33*a44*a55*u*b66...
            + a22*u*b33*a44*u*b55*a66 + a22*u*b33*a44*u*b55*l6 -
a22*u*b33*a44*u*b55*u*b66...
            - a22*u*b33*a44*a65*l5 - a22*u*b33*a44*a65*u*b56...
            + a22*u*b33*u*b44*u*a55*a66 + a22*u*b33*u*b44*a55*l6 -
a22*u*b33*u*b44*a55*u*b66...
            - a22*u*b33*u*b44*u*b55*a66 - a22*u*b33*u*b44*u*b55*l6 +
a22*u*b33*u*b44*u*b55*u*b66...
            + a22*u*b33*u*b44*a65 + a22*u*b33*u*b44*a65*u*b56...
            - a22*u*b33*u*b45*a54 - a22*u*b33*u*b45*a54*l6 +
a22*u*b33*u*b45*a54*u*b66...
            - a22*u*b33*l4*a54*a65...
            + a22*u*b34*a43*a55*a66 + a22*u*b34*a43*a55*l6 -
a22*u*b34*a43*a55*u*b66...
            - a22*u*b34*a43*u*b55*a66 - a22*u*b34*a43*u*b55*l6 +
a22*u*b34*a43*u*b55*u*b66...
            + a22*u*b34*a43*a65*l5 + a22*u*b34*a43*a65*u*b56...
            + a22*l3*a43*a54*a65...
            - u*b22*a33*a44*a55*a66 - u*b22*a33*a44*a55*l6 +
u*b22*a33*a44*a55*u*b66...
            + u*b22*a33*a44*u*b55*a66 + u*b22*a33*a44*u*b55*l6 -
u*b22*a33*a44*u*b55*u*b66...
            - u*b22*a33*a44*a65*l5 - u*b22*a33*a44*a65*u*b56...
            + u*b22*a33*u*b44*a55*a66 + u*b22*a33*u*b44*a55*l6 -
u*b22*a33*u*b44*a55*u*b66...
            - u*b22*a33*u*b44*u*b55*a66 - u*b22*a33*u*b44*u*b55*l6 +
u*b22*a33*u*b44*u*b55*u*b66...
            + u*b22*a33*u*b44*a65*l5 + u*b22*a33*u*b44*a65*u*b56...
            - u*b22*a33*u*b45*a54*a66 - u*b22*a33*u*b45*a54*l6 +
u*b22*a33*u*b45*a54*u*b66...
            - u*b22*a33*l4*a54*a65...
            + u*b22*u*b33*a44*a55*a66 + u*b22*u*b33*a44*a55*l6 -
u*b22*u*b33*a44*a55*u*b66...
            + u*b22*u*b33*a44*u*b55*a66 - u*b22*u*b33*a44*u*b55*l6 +
u*b22*u*b33*a44*u*b55*u*b66...
            + u*b22*u*b33*a44*a65*l5 + u*b22*u*b33*a44*a65*u*b56...
```

```
        - u*b22*u*b33*u*b44*a55*a66 - u*b22*u*b33*u*b44*a55*l6 +
u*b22*u*b33*u*b44*a55*u*b66...
        + u*b22*u*b33*u*b44*u*b55*a66 + u*b22*u*b33*u*b44*u*b55*l6 -
u*b22*u*b33*u*b44*u*b55*u*b66...
        - u*b22*u*b33*u*b44*a65*l5 - u*b22*u*b33*u*b44*a65*u*b56...
        + u*b22*u*b33*u*b45*a54*a66 + u*b22*u*b33*u*b45*a54*l6 -
u*b22*u*b33*u*b45*a54*u*b66...
        + u*b22*u*b33*l4*a54*a65...
        - u*b22*u*b34*a43*a55*a66 - u*b22*u*b34*a43*a55*l6 +
u*b22*u*b34*a43*a55*u*b66...
        + u*b22*u*b34*a43*u*b55*a66 + u*b22*u*b34*a43*u*b55*l6 -
u*b22*u*b34*a43*u*b55*u*b66...
        - u*b22*u*b34*a43*a65*l5 - u*b22*u*b34*a43*a65*u*b56...
        - u*b22*l3*a43*a54*a65...
        + u*b23*a32*a44*a55*a66 + u*b23*a32*a44*a55*l6 -
u*b23*a32*a44*a55*u*b66...
        - u*b23*a32*a44*u*b55*a66 - u*b23*a32*a44*u*b55*l6 +
u*b23*a32*a44*u*b55*u*b66...
        + u*b23*a32*a44*a65*l5 + u*b23*a32*a44*a65*u*b56...
        - u*b23*a32*u*b44*a55 - u*b23*a32*u*b44*a55*l6 +
u*b23*a32*u*b44*a55*u*b66...
        + u*b23*a32*u*b44*u*b55*a66 + u*b23*a32*u*b44*u*b55*l6 -
u*b23*a32*u*b44*u*b55*u*b66...
        - u*b23*a32*u*b44*a65*l5 - u*b23*a32*u*b44*a65*u*b56...
        + u*b23*a32*u*b45*a54*a66 + u*b23*a32*u*b45*a54*l6 -
u*b23*a32*u*b45*a54*u*b66...
        + u*b23*a32*l4*a54*a65...
        + l2*a32*a43*a54*a65...
        + a11*a33*a44*a55*a66 + a11*a33*a44*a55*l6 - a11*a33*a44*a55*u*b66...
        - a11*a33*a44*u*b55*a66 - a11*a33*a44*u*b55*l6 +
a11*a33*a44*u*b55*u*b66...
        + a11*a33*a44*a65*l5 + a11*a33*a44*a65*u*b56...
        - a11*a33*u*b44*a55*a66 - a11*a33*u*b44*a55*l6 +
a11*a33*u*b44*a55*u*b66...
        + a11*a33*u*b44*u*b55*a66 + a11*a33*u*b44*u*b55*l6 -
a11*a33*u*b44*u*b55*u*b66...
        - a11*a33*u*b44*a65*l5 - a11*a33*u*b44*a65*u*b56...
        + a11*a33*l4*a54*a65...
```

```
          - a11*u*b33*a44*a55*a66 - a11*u*b33*a44*a55*l6 +
a11*u*b33*a44*a55*u*b66...
          + a11*u*b33*a44*u*b55*a66 + a11*u*b33*a44*u*b55*l6 -
a11*u*b33*a44*u*b55*u*b66...
          - a11*u*b33*a44*a65*l5 - a11*u*b33*a44*a65*u*b56...
          + a11*u*b33*u*b44*a55*a66 + a11*u*b33*u*b44*a55*l6 -
a11*u*b33*u*b44*a55*u*b66...
          - a11*u*b33*u*b44*u*b55*a66 - a11*u*b33*u*b44*u*b55*l6 +
a11*u*b33*u*b44*u*b55*u*b66...
          + a11*u*b33*u*b44*a65*l5 + a11*u*b33*u*b44*a65*u*b56...
          - a11*u*b33*u*b45*a54*a66 - a11*u*b33*u*b45*a54*l6 +
a11*u*b33*u*b45*a54*u*b66...
          - a11*u*b33*l4*a54*a65...
          + a11*u*b34*a43*a55*a66 + a11*u*b34*a43*a55*l6 -
a11*u*b34*a43*a55*u*b66...
          - a11*u*b34*a43*u*b55*a66 - a11*u*b34*a43*u*b55*l6 +
a11*u*b34*a43*u*b55*u*b66...
          + a11*u*b34*a43*a65*l5 + a11*u*b34*a43*a65*u*b56...
          + a11*l3*a43*a54*a65...
          + a11*a22*a44*a55*a66 + a11*a22*a44*a55*l6 - a11*a22*a44*a55*u*b66...
          - a11*a22*a44*u*b55*a66 - a11*a22*a44*u*b55*l6 +
a11*a22*a44*u*b55*u*b66...
          + a11*a22*a44*a65*l5 + a11*a22*a44*a65*u*b56...
          - a11*a22*u*b44*a55*a66 - a11*a22*u*b44*a55*l6 +
a11*a22*u*b44*a55*u*b66...
          + a11*a22*u*b44*u*b55*a66 + a11*a22*u*b44*u*b55*l6 -
a11*a22*u*b44*u*b55*u*b66...
          - a11*a22*u*b44*a65*l5 - a11*a22*u*b44*a65*u*b56...
          + a11*a22*l4*a54*a65...
          + a11*a22*a33*a55*a66 + a11*a22*a33*a55*l6 - a11*a22*a33*a55*u*b66...
          - a11*a22*a33*u*b55*a66 - a11*a22*a33*u*b55*l6 +
a11*a22*a33*u*b55*u*b66...
          + a11*a22*a33*a65 + a11*a22*a33*a65*u*b56...
          + a11*a22*a33*a44*a55 + a11*a22*a33*a44*l6 -
a11*a22*a33*u*b44*u*b66...
          + a11*a22*a33*a44*a55 - a11*a22*a33*a44*u*b55...
          - a11*a22*a33*u*b44*a66 - a11*a22*a33*u*b44*l6 +
a11*a22*a33*u*b44*u*b66...
```

```
        - a11*a22*a33*u*b44*a55 + a11*a22*a33*u*b44*u*b55...
        + a11*a22*a33*u*b45*a54...
        - a11*a22*u*b33*a55*a66 - a11*a22*u*b33*a55*l6 +
a11*a22*u*b33*a55*u*b66...
        + a11*a22*u*b33*u*b55*a66 + a11*a22*u*b33*u*b55*l6 -
a11*a22*u*b33*u*b55*u*b66...
        - a11*a22*u*b33*a65*l5 - a11*a22*u*b33*a65*u*b56...
        - a11*a22*u*b33*a44*a66 - a11*a22*u*b33*a44*l6 +
a11*a22*u*b33*a44*u*b66...
        - a11*a22*u*b33*a44*a55 + a11*a22*u*b33*a44*u*b55...
        + a11*a22*u*b33*u*b44*a66 + a11*a22*u*b33*u*b44*l6 -
a11*a22*u*b33*u*b44*u*b66...
        + a11*a22*u*b33*u*b44*a55 - a11*a22*u*b33*u*b44*u*b55...
        - a11*a22*u*b33*u*b45*a54...
        + a11*a22*u*b34*a43*a66 + a11*a22*u*b34*a43*l6 -
a11*a22*u*b34*a43*u*b66...
        + a11*a22*u*b34*a43*a55 - a11*a22*u*b34*a43*u*b55...
        - a11*u*b22*a44*a55*a66 - a11*u*b22*a44*a55*l6 +
a11*u*b22*a44*a55*u*b66...
        + a11*u*b22*a44*u*b55*a66 + a11*u*b22*a44*u*b55*l6 -
a11*u*b22*a44*u*b55*u*b66...
        - a11*u*b22*a44*a65*l5 - a11*u*b22*a44*a65*u*b56...
        + a11*u*b22*u*b44*a55*a66 + a11*u*b22*u*b44*a55*l6 -
a11*u*b22*u*b44*a55*u*b66...
        - a11*u*b22*u*b44*u*b55*a66 - a11*u*b22*u*b44*u*b55*l6 +
a11*u*b22*u*b44*u*b55*u*b66...
        + a11*u*b22*u*b44*a65*l5 + a11*u*b22*u*b44*a65*u*b56...
        - a11*u*b22*u*b45*a54*a66 - a11*u*b22*a54*l6 +
a11*u*b22*u*b45*a54*u*b66...
        - a11*u*b22*l4*a54*a65...
        - a11*u*b22*a33*a55*a66 - a11*u*b22*a33*a55*l6 +
a11*u*b22*a33*a55*u*b66...
        + a11*u*b22*a33*u*b55*a66 + a11*u*b22*a33*u*b55*l6 -
a11*u*b22*u*b55*u*b66...
        - a11*u*b22*a33*a65*l5 - a11*u*b22*a33*a65*u*b56...
        - a11*u*b22*a33*a44*a66 - a11*u*b22*a33*a44*l6 +
a11*u*b22*a33*a44*u*b66...
        - a11*u*b22*a33*a44*a55 + a11*u*b22*a33*a44*u*b55...
```

```
          + a11*u*b22*a33*u*b44*a66 + a11*u*b22*a33*u*b44*l6 -
a11*u*b22*a33*u*b44*u*b66...
          + a11*u*b22*a33*u*b44*a55 - a11*u*b22*a33*u*b44*u*b55...
          - a11*u*b22*a33*u*b45*a54...
          + a11*u*b22*u*b33*a55*a66 + a11*u*b22*u*b33*a55*l6 -
a11*u*b22*u*b33*a55*u*b66...
          - a11*u*b22*u*b33*u*b55*a66 - a11*u*b22*u*b33*u*b55*l6 +
a11*u*b22*u*b33*u*b55*u*b66...
          + a11*u*b22*u*b33*a65*l5 + a11*u*b22*u*b33*u*b56...
          + a11*u*b22*u*b33*a44*a66 + a11*u*b22*u*b33*a44*l6 -
a11*u*b22*u*b33*a44*u*b66...
          + a11*u*b22*u*b33*a44*a55 - a11*u*b22*u*b33*a44*u*b55...
          - a11*u*b22*u*b33*u*b44*a66 - a11*u*b22*u*b33*u*b44*l6 +
a11*u*b22*u*b33*u*b44*u*b66...
          - a11*u*b22*u*b33*u*b44*a55 + a11*u*b22*u*b33*u*b44*u*b55...
          + a11*u*b22*u*b33*u*b45*a54...
          - a11*u*b22*u*b34*a43*a66 - a11*u*b22*u*b34*a43*l6 +
a11*u*b22*u*b34*a43*u*b66...
          - a11*u*b22*u*b34*a43*a55 + a11*u*b22*u*b34*a43*u*b55...
          + a11*u*b23*a32*a55*a66 + a11*u*b23*a32*a55*l6 -
a11*u*b23*a32*a55*u*b66...
          - a11*u*b23*a32*u*b55*a66 - a11*u*b23*a32*u*b55*l6 +
a11*u*b23*a32*u*b55*u*b66...
          + a11*u*b23*a32*a65*l5 + a11*u*b23*a32*a65*u*b56...
          + a11*u*b23*a32*a44*a66 + a11*u*b23*a32*a44*l6 -
a11*u*b23*a32*a44*u*b66...
          + a11*u*b23*a32*a44*a55 - a11*u*b23*a32*a44*u*b55...
          - a11*u*b23*a32*u*b44*a66 - a11*u*b23*a32*u*b44*l6 +
a11*u*b23*a32*u*b44*u*b66...
          - a11*u*b23*a32*u*b44*a55 + a11*u*b23*a32*u*b44*u*b55...
          + a11*u*b23*a32*u*b45*a54...
          - u*b11*a33*a44*a55*a66 - u*b11*a33*a44*a55*l6 +
a11*a33*a44*a55*u*b66...
          + u*b11*a33*a44*u*b55*a66 + u*b11*a33*a44*u*b55*l6 -
u*b11*a33*a44*u*b55*u*b66...
          - u*b11*a33*a44*a65*l5 - u*b11*a33*a44*a65*u*b56...
          + u*b11*a33*u*b44*a55*a66 + u*b11*a33*u*b44*a55*l6 -
u*b11*a33*u*b44*a55*u*b66...
```

```
            - u*b11*a33*u*b44*u*b55*a66 - u*b11*a33*u*b44*u*b55*l6 +
u*b11*a33*u*b44*u*b55*u*b66...
            + u*b11*a33*u*b44*a65*l5 + u*b11*a33*u*b44*a65*u*b56...
            - u*b11*a33*u*b45*a54*a66 - u*b11*a33*u*b45*a54*l6 +
u*b11*a33*u*b45*a54*u*b66...
            - u*b11*a33*l4*a54*a65...
            + u*b11*u*b33*a44*a55*a66 + u*b11*u*b33*a44*a55*l6 -
u*b11*u*b33*a44*a55*u*b66...
            - u*b11*u*b33*a44*u*b55*a66 - u*b11*u*b33*a44*u*b55*l6 +
u*b11*u*b33*a44*u*b55*u*b66...
            + u*b11*u*b33*a44*a65*l5 + u*b11*u*b33*a44*a65*u*b56...
            - u*b11*u*b33*u*b44*a55*a66 - u*b11*u*b33*u*b44*a55*l6 +
u*b11*u*b33*u*b44*a55*u*b66...
            + u*b11*u*b33*u*b44*u*b55*a66 + u*b11*u*b33*u*b44*u*b55*l6 -
u*b11*u*b33*u*b44*u*b55*u*b66...
            - u*b11*u*b33*u*b44*a65*l5 - u*b11*u*b33*u*b44*a65*u*b56...
            + u*b11*u*b33*u*b45*a54*a66 + u*b11*u*b33*u*b45*a54*l6 -
u*b11*u*b33*u*b45*a54*u*b66...
            + u*b11*u*b33*l4*a54*a65...
            - u*b11*u*b34*a43*a55*a66 - u*b11*u*b34*a43*a55*l6 +
u*b11*u*b34*a43*a55*u*b66...
            + u*b11*u*b34*a43*u*b55*a66 + u*b11*u*b34*a43*u*b55*l6 -
u*b11*u*b34*a43*u*b55*u*b66...
            - u*b11*u*b34*a43*a65*l5 - u*b11*u*b34*a43*a65*u*b56...
            - u*b11*l3*a43*a54*a65...
            - u*b11*a22*a44*a55*a66 - u*b11*a22*a44*a55*l6 +
u*b11*a22*a44*a55*u*b66...
            - u*b11*a22*a44*a65*l5 - u*b11*a22*a44*a65*u*b56...
            + u*b11*a22*u*b44*a55*a66 + u*b11*a22*u*b44*a55*l6 -
u*b11*a22*u*b44*a55*u*b66...
            - u*b11*a22*u*b44*u*b55*a66 - u*b11*a22*u*b44*u*b55*l6 +
u*b11*a22*u*b44*u*b55*u*b66...
            + u*b11*a22*u*b44*a65*l5 + u*b11*a22*u*b44*a65*u*b56...
            - u*b11*a22*u*b45*a54*a66 - u*b11*a22*u*b45*a54*l6 +
u*b11*a22*u*b45*a54*u*b66...
            - u*b11*a22*l4*a54*a65...
            - u*b11*a22*a33*a55*a66 - u*b11*a22*a33*a55*l6 +
u*b11*a22*a33*a55*u*b66...
```

```
        + u*b11*a22*a33*u*b55*a66 + u*b11*a22*a33*u*b55*l6 -
u*b11*a22*a33*u*b55*u*b66...
        - u*b11*a22*a33*a65*l5 - u*b11*a22*a33*a65*u*b56...
        - u*b11*a22*a33*a44*a66 - u*b11*a22*a33*a44*l6 +
u*b11*a22*a33*a44*u*b66...
        - u*b11*a22*a33*a44*a55 + u*b11*a22*a33*a44*u*b55...
        + u*b11*a22*a33*u*b44*a66 + u*b11*a22*a33*u*b44*l6 -
u*b11*a22*a33*u*b44*u*b66...
        + u*b11*a22*a33*u*b44*a55 - u*b11*a22*a33*u*b44*u*b55...
        - u*b11*a22*a33*u*b45*a54...
        + u*b11*a22*u*b33*a55*a66 + u*b11*a22*u*b33*a55*l6 -
u*b11*a22*u*b33*a55*u*b66...
        - u*b11*a22*u*b33*u*b55*a66 - u*b11*a22*u*b33*u*b55*l6 +
u*b11*a22*u*b33*u*b55*u*b66...
        + u*b11*a22*u*b33*a65*l5 + u*b11*a22*u*b33*a65*u*b56...
        + u*b11*a22*u*b33*a44*a66 + u*b11*a22*u*b33*a44*l6 -
u*b11*a22*u*b33*a44*u*b66...
        + u*b11*a22*u*b33*a44*a55 - u*b11*a22*u*b33*a44*u*b55...
        - u*b11*a22*u*b33*u*b44*a66 - u*b11*a22*u*b33*u*b44*l6 +
u*b11*a22*u*b33*u*b44*u*b66...
        - u*b11*a22*u*b33*u*b44*a55 + u*b11*a22*u*b33*u*b44*u*b55...
        + u*b11*a22*u*b33*u*b45*a54...
        - u*b11*a22*u*b34*a43*a66 - u*b11*a22*u*b34*a43*l6 +
u*b11*a22*u*b34*a43*u*b66...
        - u*b11*a22*u*b34*a43*a55 + u*b11*a22*u*b34*a43*u*b55...
        + u*b11*u*b22*a44*a55*a66 + u*b11*u*b22*a44*a55*l6 -
u*b11*u*b22*a44*a55*u*b66...
        - u*b11*u*b22*a44*u*b55*a66 - u*b11*u*b22*a44*u*b55*l6 +
u*b11*u*b22*a44*u*b55*u*b66...
        + u*b11*u*b22*a44*a65*l5 + u*b11*u*b22*a44*a65*u*b56...
        - u*b11*u*b22*u*b44*a55*a66 - u*b11*u*b22*u*b44*a55*l6 +
u*b11*u*b22*u*b44*a55*u*b66...
        + u*b11*u*b22*u*b44*u*b55*a66 + u*b11*u*b22*u*b44*u*b55*l6 -
u*b11*u*b44*u*b55*u*b66...
        - u*b11*u*b22*u*b44*a65*l5 - u*b11*u*b22*u*b44*a65*u*b56...
        + u*b11*u*b22*u*b45*a54*a66 + u*b11*u*b22*u*b45*a54*l6 -
u*b11*u*b22*u*b45*a54*u*b66...
        + u*b11*u*b22*l4*a54*a65...
        + u*b11*u*b22*a33*a55*a66 + u*b11*u*b22*a33*a55*l6 -
u*b11*u*b22*a33*a55*u*b66...
        - u*b11*u*b22*a33*u*b55*a66 - u*b11*u*b22*a33*u*b55*l6 +
u*b11*u*b22*a33*u*b55*u*b66...
        + u*b11*u*b22*a33*a65*l5 + u*b11*u*b22*a33*a65*u*b56...
        + u*b11*u*b22*a33*a44*a66 + u*b11*u*b22*a33*a44*l6 -
u*b11*u*b22*a33*a44*u*b66...
        + u*b11*u*b22*a33*a44*a55 - u*b11*u*b22*a33*a44*u*b55...
        - u*b11*u*b22*a33*u*b44*a66 - u*b11*u*b22*a33*u*b44*l6 +
u*b11*u*b22*a33*u*b44*u*b66...
        - u*b11*u*b22*a33*u*b44*a55 + u*b11*u*b22*a33*u*b44*u*b55...
        + u*b11*u*b22*a33*u*b45*a54...
        - u*b11*u*b22*u*b33*a55*a66 - u*b11*u*b22*u*b33*a55*l6 +
u*b11*u*b22*u*b33*a55*u*b66...
        + u*b11*u*b22*u*b33*u*b55*a66 + u*b11*u*b22*u*b33*u*b55*l6 -
u*b11*u*b22*u*b33*a55*u*b66...
        + u*b11*u*b22*u*b33*u*b55*a66 + u*b11*u*b22*u*b33*u*b55*l6 -
u*b11*u*b22*u*b33*u*b55*u*b66...
        - u*b11*u*b22*u*b33*a65*l5 - u*b11*u*b22*u*b33*a65*u*b56...
        - u*b11*u*b22*u*b33*a44*a66 - u*b11*u*b22*u*b33*a44*l6 +
u*b11*u*b22*u*b33*a44*u*b66...
        - u*b11*u*b22*u*b33*a44*a55 + u*b11*u*b22*u*b33*a44*u*b55...
        + u*b11*u*b22*u*b33*u*b44*a66 + u*b11*u*b22*u*b33*u*b44*l6 -
u*b11*u*b22*u*b33*u*b44*u*b66...
        + u*b11*u*b22*u*b33*u*b44*a55 - u*b11*u*b22*u*b33*u*b44*u*b55...
        - u*b11*u*b22*u*b33*u*b45*a54...
        + u*b11*u*b22*u*b34*a43*a66 + u*b11*u*b22*u*b34*a43*l6 -
u*b11*u*b22*u*b34*a43*u*b66...
        + u*b11*u*b22*u*b34*a43*a55 - u*b11*u*b22*u*b34*a43*u*b55...
        - u*b11*u*b23*a32*a55*a66 - u*b11*u*b23*a32*a55*l6 +
u*b11*u*b23*a32*a55*u*b66...
        + u*b11*u*b23*a32*u*b55*a66 + u*b11*u*b23*a32*u*b55*l6 -
u*b11*u*b23*a32*u*b55*u*b66...
        - u*b11*u*b23*a32*a65*l5 - u*b11*u*b23*a32*a65*u*b56...
        - u*b11*u*b23*a32*a44*a66 - u*b11*u*b23*a32*a44*l6 +
u*b11*u*b23*a32*a44*u*b66...
        - u*b11*u*b23*a32*a44*a55 + u*b11*u*b23*a32*a44*u*b55...
```

```
          + u*b11*u*b23*a32*u*b44*a66 + u*b11*u*b23*a32*u*b44*l6 -
u*b11*u*b23*a32*u*b44*u*b66...
          + u*b11*u*b23*a32*u*b44*a55 - u*b11*u*b23*a32*u*b44*u*b55...
          - u*b11*u*b23*a32*u*b45*a54);
%
det1g = vpa(a11*a22*a33*a44*a55*a66 + a11*a22*a33*a44*a55*l6 -
a11*a22*a33*a44*a55*u*b66...
                    - a11*a22*a33*a44*u*b55*a66 - a11*a22*a33*a33*u*b55*l6 +
a11*a22*a33*a44*u*b55*u*b66...
                    + a11*a22*a33*a44*a65*l5 + a11*a22*a33*a44*a65*u*b56...
                    + a11*a22*a33*u*b45*a54*a66 + a11*a22*a33*u*b45*a54*l6 -
a11*a22*a33*u*b45*a54*u*b66...
                    + a11*a22*a33*l4*a54*a65...
                    - a11*a22*u*b33*a44*a55*a66 - a11*a22*u*b33*a44*a55*l6 +
a11*a22*u*b33*a44*a55*u*b66...
                    + a11*a22*u*b33*a44*u*b55*a66 + a11*a22*u*b33*a44*u*b55*l6 -
a11*a22*u*b33*a44*u*b55*u*b66...
                    + a11*a22*u*b33*u*b44*a65*l5 +
a11*a22*u*b33*u*b44*a65*u*b56...
                    - a11*a22*u*b33*u*b45*a54*a66 - a11*a22*u*b33*u*b45*a54*l6 +
a11*a22*u*b33*u*b45*a54*u*b66...
                    - a11*a22*u*b33*l4*a54*a65...
                    + a11*a22*u*b34*a43*a55*a66 + a11*a22*u*b34*a43*a55*l6 -
a11*a22*u*b34*a43*a55*u*b66...
                    - a11*a22*u*b34*a43*u*b55*a66 - a11*a22*u*b34*a43*u*b55*l6 -
a11*a22*u*b34*a43*u*b55*u*b66...
                    + a11*a22*u*b34*a43*a65*l5 + a11*a22*u*b34*a43*a65*u*b56...
                    + a11*a22*l3*a43*a54*a65...
                    - a11*u*b22*a33*a44*a55*a66 - a11*u*b22*a33*a44*a55*l6 +
a11*u*b22*a33*a44*a55*u*b66...
                    + a11*u*b22*a33*a44*u*b55*a66 + a11*u*b22*a33*a44*u*b55*l6 -
a11*u*b22*a33*a44*u*b55*u*b66...
                    - a11*u*b22*a33*a44*a65*l5 - a11*u*b22*a33*a44*a65*u*b56...
                    + a11*u*b22*a33*u*b44*a55*a66 + a11*u*b22*a33*u*b44*a55*l6 -
a11*u*b22*a33*u*b44*a55*u*b66...
                    - a11*u*b22*a33*u*b44*u*b55*a66 -
a11*u*b22*a33*u*b44*u*b55*l6 + a11*u*b22*a33*u*b44*u*b55*u*b66...
```

```
                 + a11*u*b22*a33*u*b44*a65*l5 +
a11*u*b22*a33*u*b44*a65*u*b56...
                 - a11*u*b22*a33*u*b45*a54*a66 - a11*u*b22*a33*u*b45*a54*l6 +
a11*u*b22*a33*u*b45*a54*u*b66...
                 - a11*u*b22*a33*l4*a54*a65...
                 + a11*u*b22*u*b33*a44*a55*a66 + a11*u*b22*u*b33*a44*a55*l6 -
a11*u*b22*u*b33*a44*a55*u*b66...
                 - a11*u*b22*u*b33*a44*u*b55*a66 -
a11*u*b22*u*b33*a44*u*b55*l6 + a11*u*b22*u*b33*a44*u*b55*u*b66...
                 + a11*u*b22*u*b33*a44*a65*l5 +
a11*u*b22*u*b33*a44*a65*u*b56...
                 - a11*u*b22*u*b33*u*b44*a55*a66 -
a11*u*b22*u*b33*u*b44*a55*l6 + a11*u*b22*u*b33*u*b44*a55*u*b66...
                 + a11*u*b22*u*b33*u*b44*u*b55*a66 +
a11*u*b22*u*b33*u*b44*u*b55*l6 - a11*u*b22*u*b33*u*b44*u*b55*u*b66...
                 - a11*u*b22*u*b33*u*b44*a65*l5 -
a11*u*b22*u*b33*u*b44*a65*u*b56...
                 + a11*u*b22*u*b33*u*b45*a54*a66 +
a11*u*b22*u*b33*u*b45*a54*l6 - a11*u*b22*u*b33*u*b45*a54*u*b66...
                 + a11*u*b22*u*b33*l4*a54*a65...
                 - a11*u*b22*u*b34*a43*a55*a66 - a11*u*b22*u*b34*a43*a55*l6 +
a11*u*b22*u*b34*a43*a55*u*b55*u*b66...
                 + a11*u*b22*u*b34*a43*u*b55*a66 +
a11*u*b22*u*b34*a43*u*b55*l6 - a11*u*b22*u*b34*a43*u*b55*u*b66...
                 - a11*u*b22*u*b34*a43*a65*l5 -
a11*u*b33*u*b34*a43*a65*u*b56...
                 - a11*u*b22*l3*a43*a54*a65...
                 + a11*u*b23*a32*a44*a55*a66 + a11*u*b23*a32*a44*a55*l6 -
a11*u*b23*a32*a44*a55*u*b66...
                 - a11*u*b23*a32*a44*u*b55*a66 - a11*u*b23*a32*a44*u*b55*l6 +
a11*u*b23*a32*a44*u*b55*u*b66...
                 + a11*u*b23*a32*a44*a65*l5 + a11*u*b23*a32*a44*a65*u*b56...
                 - a11*u*b23*a32*u*b44*a55*a66 - a11*u*b23*a32*u*b44*a55*l6 +
a11*u*b23*a32*u*b44*a55*u*b66...
                 + a11*u*b23*a32*u*b44*u*b55*a66 +
a11*u*b23*a32*u*b44*u*b55*l6 - a11*u*b23*a32*u*b44*u*b55*u*b66...
                 - a11*u*b23*a32*u*b44*a65*l5 -
a11*u*b23*a32*u*b44*a65*u*b56...
```

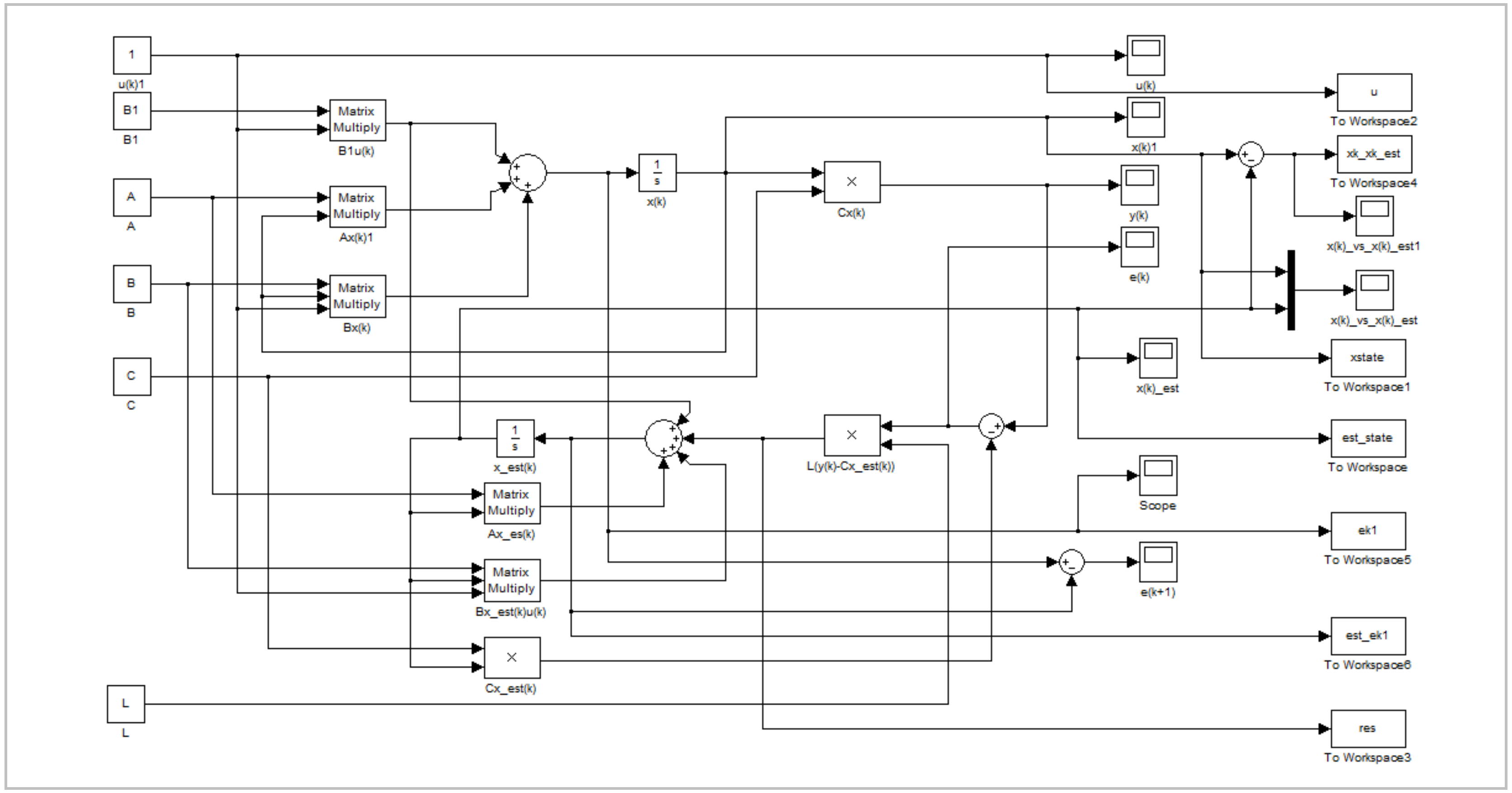
```
                    + a11*u*b23*a32*u*b45*a54*a66 + a11*u*b23*a32*u*b45*a54*l6 -
a11*u*b23*a32*u*b45*a54*u*b66...
                    + a11*u*b23*a32*l4*a54*a65...
                    + a11*l2*a32*a43*a54*a65...
                    - u*b11*a22*a33*a44*a55*a66 - u*b11*a22*a33*a44*a55*l6 +
u*b11*a22*a33*a44*a55*u*b66...
                    + u*b11*a22*a33*a44*u*b55*a66 + u*b11*a22*a33*a44*u*b55*l6 -
u*b11*a22*a33*a44*u*b55*u*b66...
                    - u*b11*a22*a33*a44*a65*l5 - u*b11*a22*a33*a44*a65*u*b56...
                    + u*b11*a22*a33*u*b44*a55*a66 + u*b11*a22*a33*u*b44*a55*l6 -
u*b11*a22*a33*u*b44*a55*u*b66...
                    - u*b11*a22*a33*u*b44*u*b55*a66 -
u*b11*a22*a33*u*b44*u*b55*l6 + u*b11*a22*a33*u*b44*u*b55*u*b66...
                    + u*b11*a22*a33*u*b44*a65*l5 +
u*b11*a22*a33*u*b44*a65*u*b56...
                    - u*b11*a22*a33*u*b45*a54*a66 - u*b11*a22*a33*u*b45*a54*l6 +
u*b11*a22*a33*u*b45*a54*u*b66...
                    - u*b11*a22*a33*l4*a54*a65...
                    + u*b11*a22*u*b33*a44*a55*a66 + u*b11*a22*u*b33*a44*a55*l6 -
u*b11*a22*u*b33*a44*a55*u*b66...
                    - u*b11*a22*u*b33*a44*u*b55*a66 -
u*b11*a22*u*b33*a44*u*b55*l5 + u*b11*a22*u*b33*a44*u*b55*u*b66...
                    + u*b11*a22*u*b33*a44*a65*l5 +
u*b11*a22*u*b33*a44*a65*u*b56...
                    - u*b11*a22*u*b33*u*b44*a55*a66 -
u*b11*a22*u*b33*u*b44*a55*l6 + u*b11*a22*u*b33*u*b44*a55*u*b66...
                    + u*b11*a22*u*b33*u*b44*u*b55*a66 +
u*b11*a22*u*b33*u*b44*u*b55*l6 - u*b11*a22*u*b33*u*b44*u*b55*u*b66...
                    - u*b11*a22*u*b33*u*b44*a65*l5 -
u*b11*a22*u*b33*u*b44*a65*u*b56...
                    + u*b11*a22*u*b33*u*b45*a54*a66 +
u*b11*a22*u*b33*u*b45*a54*l6 - u*b11*a22*u*b33*u*b45*a54*u*b66...
                    + u*b11*a22*u*b33*l4*a54*a65...
                    - u*b11*a22*u*b34*a43*a55*a66 - u*b11*a22*u*b34*a43*a55*l6 +
u*b11*a22*u*b34*a43*a55*u*b66...
                    + u*b11*a22*u*b34*a43*u*b55*a66 +
u*b11*a22*u*b34*a43*u*b55*l6 - u*b11*a22*u*b34*a43*u*b55*u*b66...
```

```
                 - u*b11*a22*u*b34*a43*a65*l5 -
u*b11*a22*u*b34*a43*a65*u*b56...
                 - u*b11*a22*l3*a43*a54*a65...
                 + u*b11*u*b22*a33*a44*a55*a66 + u*b11*u*b22*a33*a44*a55*l6 -
u*b11*u*b22*a33*a44*a55*u*b66...
                 - u*b11*u*b22*a33*a44*u*b55*a66 -
u*b11*u*b22*a33*a44*u*b55*l6 + u*b11*u*b22*a33*a44*u*b55*u*b66...
                 + u*b11*u*b22*a33*a44*a65*l5 +
u*b11*u*b22*a33*a44*a66*u*b56...
                 - u*b11*u*b22*a33*u*b44*a55*a66 -
u*b11*u*b22*a33*u*b44*a55*l6 + u*b11*u*b22*a33*u*b44*a55*u*b66...
                 + u*b11*u*b22*a33*u*b44*u*b55*a66 +
u*b11*u*b22*a33*u*b44*u*b55*l6 - u*b11*u*b22*a33*u*b44*u*b55*u*b66...
                 - u*b11*u*b22*a33*u*b44*a65*l5 -
u*b11*u*b22*a33*u*b44*a65*u*b56...
                 + u*b11*u*b22*a33*u*b45*a54*a66 +
u*b11*u*b22*a33*u*b45*a54*l6 - u*b11*u*b22*a33*u*b45*a54*u*b66...
                 + u*b11*u*b22*a33*l4*a54*a65...
                 - u*b11*u*b22*u*b33*a44*a55*a66 -
u*b11*u*b22*u*b33*a44*a55*l6 + u*b11*u*b22*u*b33*a44*a55*u*b66...
                 + u*b11*u*b22*u*b33*a44*u*b55*a66 +
u*b11*u*b22*u*b33*a44*u*b55*l6 - u*b11*u*b22*u*b33*a44*u*b55*u*b66...
                 - u*b11*u*b22*u*b33*a44*a65*l5 -
u*b11*u*b22*u*b33*a44*a65*u*b56...
                 + u*b11*u*b22*u*b33*u*b44*a55*a66 +
u*b11*u*b22*u*b33*u*b44*a55*l6 - u*b11*u*b22*u*b33*u*b44*a55*u*b66...
                 - u*b11*u*b22*u*b33*u*b44*u*b55*a66 -
u*b11*u*b22*u*b33*u*b44*u*b55*l6 + u*b11*u*b22*u*b33*u*b44*u*b55*u*b66...
                 + u*b11*u*b22*u*b33*u*b44*a65*l5 +
u*b11*u*b22*u*b33*u*b44*a65*u*b56...
                 - u*b11*u*b22*u*b33*u*b45*a54*a66 -
u*b11*u*b22*u*b33*u*b45*a54*l6 + u*b11*u*b22*u*b33*u*b45*a54*u*b66...
                 - u*b11*u*b22*u*b33*l4*a54*a65...
                 + u*b11*u*b22*u*b34*a43*a55*a66 +
u*b11*u*b22*u*b34*a43*a55*l6 - u*b11*u*b22*u*b34*a43*a55*u*b66...
                 - u*b11*u*b22*u*b34*a43*u*b55*a66 -
u*b11*u*b22*u*b34*a43*u*b55*l6 + u*b11*u*b22*u*b34*a43*u*b55*u*b66...
```

```
                        + u*b11*u*b22*u*b34*a43*a65*l5 +
u*b11*u*b22*u*b34*a43*a65*u*b56...
                        + u*b11*u*b22*l3*a43*a54*a65...
                        - u*b11*u*b23*a32*a44*a55*a66 - u*b11*u*b23*a32*a44*a55*l6 +
u*b11*u*b23*a32*a44*a55*u*b66...
                        + u*b11*u*b23*a32*a44*u*b55*a66 +
u*b11*u*b23*a32*a44*u*b55*l6 - u*b11*u*b23*a32*a44*u*b55*u*b66...
                        - u*b11*u*b23*a32*a44*a65*l5 -
u*b11*u*b23*a32*a44*a65*u*b56...
                        + u*b11*u*b23*a32*u*b44*a55*a66 +
u*b11*u*b23*a32*u*b44*a55*l6 - u*b11*u*b23*a32*u*b44*a55*u*b66...
                        - u*b11*u*b23*a32*u*b44*u*b55*a66 -
u*b11*u*b23*a32*u*b44*u*b55*l6 + u*b11*u*b23*a32*u*b44*u*b55*u*b66...
                        - u*b11*u*b23*a32*u*b45*a54*a66 -
u*b11*u*b23*a32*u*b45*a54*l6 + u*b11*u*b23*a32*u*b45*a54*u*b66...
                        - u*b11*u*b23*a32*l4*a54*a65...
                        - u*b11*l2*a32*a43*a54*a65);
%
det1 = [1 det1b det1c det1d det1e det1f det1g];
%
det = det1 + det2 + det3 + det4 + det5 + det6;
%
%----------------------------------------------------------------------------------------------------
% Determining the desired observer (based on observer characteristic equation and
placement of poles at required location(s))
%
syms s
a = simplify((s+5)^6)
a = expand(a)
%
%------------------------------------------------------------------------------------------
% Solving the observer matrix elements and
% Comparison of the two characteristic equations and calculation of the observer
matrix
% (see if it is possible to automatically extract l's if roots are automatically changed)
%
l6 = (30 + 130.876);
l5 = (375 + 74153.1 + 85.8693*l6)/43.3821;
```

```
l4 = (2500 - 31907.1 - 1200.41*l6 + 2097.57*l5)/1646.75;
l3 = (9375 + 0.737398e7 - 28092.9*l6 + 0.345555e7*l5 + 91740.3*l4)/58832.6;
l2 = (18750 - 0.161011e9 + 0.322378e7*l6 - 0.384588e7*l5 - 0.399188e7*l4 +
     0.314216e7*l3)/0.188063e7;
l1 = (15625 + 0.132894e10 - 0.313572e8*l6 + 0.367093e8*l5 + 0.386912e8*l4 -
     0.432444e8*l3 + 0.477368e8*l2)/0.543903e8;
%
%------------------------------------------------------------------------------------------------
% Final determination of the observer gain matrix L
%
L = [l1 l2 l3 l4 l5 l6]'
%
%--------------------------------------------Program end-------------------------------------
```

APPENDIX D.2: SIMULINK program for the design of the bilinear observer



APPENDIX D.3: MATLAB program for extracting values of the design of the observer back to workspace

```
%---------------------------------------------------------------------------------------------------
%The Program is a supplementary program for extracting Simulink values
%so as to display them on Workspace (MATLAB Front-End)
%
% Author: NM. Dube
% Dated: 12 June 2014
% Updated: 03 September 2016

%---------------------------------------------------------------------------------------------------
Q = 6;
P = 3000;             % number of data points generated by Simulink for the vectors x-
state and est-state these are double arrays
                      % (according to Workspace data/ variable details)
% collecting data points for the state vector as generated by Simulink to display on
   Workspace
for i = 1:1:100
    p1(i) = xstate(1,i);
    p2(i) = xstate(2,i);
    p3(i) = xstate(3,i);
    p4(i) = xstate(4,i);
    p5(i) = xstate(5,i);
    p6(i) = xstate(6,i);
% collecting data points for the dynamic error vector as generated by Simulink to
   display on Workspace
    e1(i) = ek1(1,i);
    e2(i) = ek1(2,i);
    e3(i) = ek1(3,i);
    e4(i) = ek1(4,i);
    e5(i) = ek1(5,i);
    e6(i) = ek1(6,i);
end
% collecting data points for the estimated vector as generated by Simulink to display
   on Workspace
for j = 1:1:100
```

```
    xp1(j) = est_state(1,j);
    xp2(j) = est_state(2,j);
    xp3(j) = est_state(3,j);
    xp4(j) = est_state(4,j);
    xp5(j) = est_state(5,j);
    xp6(j) = est_state(6,j);
% collecting data points for the estimated vector as generated by Simulink to display
    on Workspace
    est_e1(j) = est_ek1(1,j);
    est_e2(j) = est_ek1(2,j);
    est_e3(j) = est_ek1(3,j);
    est_e4(j) = est_ek1(4,j);
    est_e5(j) = est_ek1(5,j);
    est_e6(j) = est_ek1(6,j);
end
% calculating new error and dynamic error vectors using data collected in the above
    steps
for k = 1:1:100
    error(:,k) = est_state(:,k) - xstate(:,k);
    error_ek1(:,k) = ek1(:,k) - est_ek1(:,k);
end
%
% Plotting observer performance in different colours
figure(1)
plot(p1,'-o','Linewidth',3.0,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
plot(p2,'-<m','Linewidth',3.0,'MarkerSize',7,'MarkerFaceColor',[1,0.4,0.6])
hold on
plot(p3,'-sy','Linewidth',3.0,'MarkerSize',7,'MarkerFaceColor',[1 1 0])
hold on
plot(p4,'-dk','Linewidth',3.0,'MarkerSize',7,'MarkerFaceColor',[0 0 0])
hold on
plot(p5,'-*g','Linewidth',3.0,'MarkerSize',8,'MarkerFaceColor',[0 1 0])
hold on
plot(p6,'-^r','Linewidth',3.0,'MarkerSize',7,'MarkerFaceColor',[1 0 0])
hold on
```

```
%
plot(xp1,':o','Linewidth',3.0,'MarkerSize',7,'MarkerFaceColor',[0.0 0.0 1.0])
hold on
plot(xp2,':<m','Linewidth',3.0,'MarkerSize',7,'MarkerFaceColor',[1,0.4,0.6])
hold on
plot(xp3,':sy','Linewidth',3.0,'MarkerSize',7,'MarkerFaceColor',[1 1 0])
hold on
plot(xp4,':dk','Linewidth',3.0,'MarkerSize',7,'MarkerFaceColor',[0 0 0])
hold on
plot(xp5,':g*','Linewidth',3.0,'MarkerSize',8,'MarkerFaceColor',[0 1 0])
hold on
plot(xp6,':^r','Linewidth',3.0,'MarkerSize',7,'MarkerFaceColor',[1 0 0])
hold on
title ('Process states vs estimated states dynamic behaviour','Fontsize',16)
xlabel('Number of cycles [number]','Fontsize',16)
ylabel('Measured and estimated states','Fontsize',16)
grid on
hold off
%
% Plotting the state error difference trajectories
figure(2)
k = 1:100;
hold on
plot(error(1,k),'o','Linewidth',3.0)
plot(error(2,k),'om','Linewidth',3.0)
plot(error(3,k),'oy','Linewidth',3.0)
plot(error(4,k),'ok','Linewidth',3.0)
plot(error(5,k),'or','Linewidth',3.0)
plot(error(6,k),'og','Linewidth',3.0)
title('Error between measured and estimated states','Fontsize',16)
xlabel('Sampling period [k]','Fontsize',16)
ylabel('Error between measured & estimated states','Fontsize',16)
grid on
hold off
%
figure(3)
```

```
k = 1:100;
hold on
plot(error_ek1(1,k),'o','Linewidth',3.0)
plot(error_ek1(2,k),'om','Linewidth',3.0)
plot(error_ek1(3,k),'oy','Linewidth',3.0)
plot(error_ek1(4,k),'ok','Linewidth',3.0)
plot(error_ek1(5,k),'or','Linewidth',3.0)
plot(error_ek1(6,k),'og','Linewidth',3.0)
title('The rate of change of the error','Fontsize',16)
xlabel('Sampling period [k]','Fontsize',16)
ylabel('Error rate','Fontsize',16)
grid on
hold off
%
%----------------------------------Program end---------------------------------------------
```

APPENDIX E: Program for state estimation using the design of the Kalman filter

```
% ---------------------------------------------------------------------------------------
% Program for state estimation of a bilinear system using a Kalman filter for
% a CCIX (continuous countercurrent ion exchange).
% The CCIX process in this application is used for water desalination.
% The model is based on the six stages UCT (University of Cape Town) model.
%
% In this program x_curr represents estimated state variables and x_bar is the
  measured states.
%
% Written by: N. Dube
% Dated: 10 January 2013
% Updated: 30 September 2017
%
%-----------------------------------------------------------------------------------------
% Original data measured  from UCT (University of Cape Town) project in 1982
% Stage H+ fractional change in liquid concentration using data obtained from UCT
Project volume 4
 stage1 = [0.221 0.577 0.730 0.847 0.920 0.936 0.968 0.974 0.981 0.989 0.988
              0.997 1.000 1.000 1.000 1.000 1.000 1.000];
 stage2 = [0.000 0.140 0.314 0.523 0.656 0.766 0.842 0.886 0.900 0.933 0.958
              0.963 0.982 0.974 0.991 0.994 0.993 0.993];
 stage3 = [0.000 0.04 0.066 0.184 0.295 0.454 0.601 0.690 0.758 0.804 0.877 0.881
              0.951 0.965 0.972 0.981 1.000 0.988];
 stage4 = [0.000 0.000 0.004 0.035 0.082 0.168 0.277 0.361 0.440 0.522 0.698
              0.784 0.899 0.931 0.966 0.966 0.991 0.991];
 stage5 = [0.000 0.000 0.000 0.003 0.020 0.052 0.113 0.167 0.207 0.340 0.474
              0.547 0.780 0.860 0.899 0.905 0.975 0.973];
 stage6 = [0.000 0.000 0.000 0.000 0.000 0.001 0.024 0.033 0.063 0.124 0.167
              0.233 0.482 0.539 0.672 0.779 0.940 0.972];
%
% Obtaining Na+ fractional change in liquid concentration
% This data of Na+ is then plotted to obtain measured states as indicated in the table
of states_bar
 stage1 = 1 – stage1;
 stage2 = 1 – stage2;
 stage3 = 1 – stage3;
 stage4 = 1 – stage4;
 stage5 = 1 – stage5;
 stage6 = 1 – stage6;
%
%-----------------------------------------------------------------------------------------------
% Initialization of mass balance equation parameters of the CCIX model based on
UCT model data
% Initializing all function coeffients and variables using model data
%
  h = 32.93;         % resin holdups (units in litres)
  H = 42.809;        % liquid holdups (units in litres)
  Fl = 2000/60;      % liquid flow rate (units in litres per min)
  %
  N = 6;             % number of stages of the cation loading colum
  d = 2/3;           % resin/liquid fractional balance (constant)
  T = 18;            % liquid upflow time (units in min)
%
   FR = 1.0;        % resin flow rate (units in litres per min) original value FR = (h*d)/T;
 conc_in = 1.0;   % sodium concentration at first stage of the column (eq/l) - process
                    disturbance
%---------------------------------------------------------------------------------------------
% Initialization of state space equations of the CCIX model
% Original coefficients values
%
    a = [1.2 1.4 1.6 1.8 2.0 2.2];
    b = [0.08 0.09 0.10 0.11 0.12 0.13];
%
%-----------------------------------------------------------------------------------------------
% Determination of matrices for state space model x_dot = Ax + Bu and y = Cx
% Declaration of parameters from the model equations as determined by,
% dx/dt = Ax(t)+B1x(t)u(t)+Bu(t)+W(t)w(t)
% parameters are l(i), m(i), m(i(j)) and k(i)
%
for i = 1:N
```

```
    l(i) = Fl/(H+a(i)*h);
    mi(i) = a(i)/(H+a(i)*h);
end

for i = 1:N-1
    mij(i) = a(i+1)/(H+a(i)*h);
    k(i) = (b(i+1) - b(i))/(H+a(i)*h);
end
    k(N) = -b(N)/(H+a(N)*h);
%
%-----------------------------------------------------------------------------------------------
% Initialization of model parameters A, B, B1 W and C using calculated parameters
li, mi and mij
 A(1,1) = -l(1);
 A(2,1) = l(2);
 A(2,2) = -l(2);
 A(3,2) = l(3);
 A(3,3) = -l(3);
 A(4,3) = l(4);
 A(4,4) = -l(4);
 A(5,4) = l(5);
 A(5,5) = -l(5);
 A(6,5) = l(6);
 A(6,6) = -l(6);
%
 B(1,1) = -mi(1);
 B(1,2) = mij(1);
 B(2,2) = -mi(2);
 B(2,3) = mij(2);
 B(3,3) = -mi(3);
 B(3,4) = mij(3);
 B(4,4) = -mi(4);
 B(4,5) = mij(4);
 B(5,5) = -mi(5);
 B(5,6) = mij(5);
 B(6,6) = -mi(6);
```

```
%
 B1 = [k(1) k(2) k(3) k(4) k(5) k(6)]';
 W = [-l(1) 0 0 0 0 0]';
 C = [0 0 0 0 0 1];
%
%-----------------------------------------------------------------------------------------
% Measurements data as obtained from UCT project experiments,
%
 x_bar = [1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 0.999 0.999 0.999 0.998...
        0.998 0.997 0.996 0.993 0.988 0.986 0.983 0.978 0.974 0.967...
        0.962 0.956 0.950 0.942 0.930 0.918 0.907 0.897 0.886 0.876...
        0.867 0.857 0.847 0.837 0.828 0.822 0.815 0.807 0.800 0.793...
        0.769 0.745 0.721 0.697 0.674 0.648 0.625 0.602 0.578 0.553...
        0.528 0.515 0.502 0.488 0.475 0.462 0.439 0.397 0.355 0.313...
        0.299 0.284 0.278 0.266 0.251 0.247 0.233 0.221 0.211 0.205...
        0.200 0.198 0.189 0.176 0.166 0.142 0.130 0.122 0.114 0.108...
        0.102 0.091 0.080 0.071 0.064 0.050 0.032 0.021 0.018 0.011;
    %
        1.000 1.000 1.000 1.000 1.000 0.988 0.964 0.938 0.914 0.887...
        0.863 0.832 0.800 0.770 0.738 0.707 0.674 0.635 0.598 0.561...
        0.524 0.485 0.458 0.435 0.410 0.386 0.364 0.340 0.320 0.300...
        0.280 0.261 0.241 0.225 0.212 0.197 0.184 0.170 0.157 0.149...
        0.142 0.134 0.126 0.117 0.114 0.110 0.107 0.105 0.103 0.100...
        0.094 0.088 0.083 0.076 0.070 0.065 0.061 0.056 0.053 0.047...
        0.043 0.042 0.041 0.039 0.038 0.037 0.036 0.034 0.029 0.025...
        0.023 0.018 0.018 0.022 0.023 0.024 0.025 0.025 0.023 0.019...
        0.016 0.014 0.010 0.008 0.008 0.007 0.007 0.006 0.005 0.005...
        0.006 0.006 0.007 0.007 0.006 0.006 0.007 0.007 0.007 0.007;
    %
        1.000 1.000 1.000 1.000 1.000 0.997 0.989 0.983 0.975 0.967...
        0.961 0.956 0.952 0.946 0.942 0.937 0.927 0.906 0.885 0.864...
        0.842 0.820 0.800 0.781 0.760 0.741 0.720 0.698 0.670 0.641...
        0.613 0.584 0.555 0.528 0.503 0.475 0.448 0.424 0.397 0.382...
        0.365 0.348 0.335 0.317 0.304 0.292 0.278 0.266 0.255 0.243...
        0.234 0.226 0.217 0.209 0.201 0.190 0.177 0.164 0.151 0.138...
        0.124 0.123 0.122 0.121 0.120 0.119 0.115 0.103 0.089 0.077...
        0.065 0.053 0.047 0.045 0.043 0.039 0.037 0.035 0.034 0.033...
        0.032 0.029 0.028 0.026 0.025 0.024 0.023 0.021 0.018 0.015...
        0.013 0.008 0.008 0.006 0.002 0.005 0.006 0.008 0.001 0.002;
    %
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 0.999 0.999 0.998 0.997 0.996 0.995 0.988 0.984 0.977...
        0.973 0.966 0.958 0.950 0.942 0.934 0.925 0.915 0.898 0.884...
        0.868 0.854 0.837 0.818 0.798 0.779 0.760 0.740 0.722 0.706...
        0.682 0.676 0.661 0.646 0.632 0.617 0.603 0.598 0.574 0.560...
        0.546 0.530 0.516 0.501 0.486 0.464 0.433 0.400 0.368 0.337...
        0.277 0.268 0.257 0.249 0.242 0.227 0.209 0.188 0.167 0.147...
        0.126 0.105 0.096 0.092 0.085 0.079 0.074 0.067 0.062 0.055...
        0.049 0.043 0.036 0.034 0.034 0.034 0.034 0.034 0.033 0.028...
        0.025 0.020 0.015 0.012 0.008 0.008 0.008 0.008 0.008 0.008;
    %
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 0.994...
        0.992 0.981 0.970 0.970 0.968 0.964 0.960 0.955 0.951 0.948...
        0.945 0.939 0.924 0.915 0.894 0.882 0.872 0.865 0.857 0.850...
        0.842 0.834 0.823 0.810 0.798 0.787 0.775 0.750 0.722 0.690...
        0.678 0.646 0.626 0.590 0.582 0.573 0.566 0.532 0.517 0.493...
        0.467 0.452 0.423 0.418 0.401 0.374 0.331 0.290 0.282 0.256...
        0.232 0.228 0.208 0.195 0.180 0.166 0.152 0.138 0.133 0.125...
        0.117 0.110 0.104 0.100 0.088 0.077 0.066 0.055 0.044 0.032...
        0.028 0.020 0.014 0.011 0.009 0.007 0.005 0.002 0.002 0.002;
    %
        1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000...
        1.000 1.000 1.000 1.000 1.000 1.000 0.999 0.999 0.999 0.998...
        0.998 0.997 0.996 0.993 0.988 0.986 0.983 0.978 0.974 0.967...
        0.962 0.956 0.950 0.942 0.930 0.918 0.907 0.897 0.886 0.876...
        0.867 0.857 0.847 0.837 0.828 0.822 0.815 0.807 0.800 0.793...
        0.769 0.745 0.721 0.697 0.674 0.648 0.625 0.602 0.578 0.553...
        0.528 0.515 0.502 0.488 0.475 0.462 0.439 0.397 0.355 0.313...
        0.299 0.284 0.278 0.266 0.251 0.247 0.233 0.221 0.211 0.205...
        0.200 0.198 0.189 0.176 0.166 0.142 0.130 0.122 0.114 0.108...
```

```
        0.102 0.091 0.080 0.071 0.064 0.050 0.032 0.021 0.018 0.011];
%
%-----------------------------------------------------------------------------------------
% Initializing full trajectories for disturbance, input and control vectors
% Initializing the state vector (x0), initial state at the input  and the full trajectory of
state variable
%
  K = 100;                          % total number of sampling points
  xf = conc_in;                       % declaring input concentration as a distubance
  FR = FR*(ones(1,K));      % control input declaration
%
%-----------------------------------------------------------------------------------------
% Plotting the real states of the system
figure(1)
plot(k,x_bar,':o','Linewidth',2.5,'MarkerSize',6)
title('Real system behabiour','Fontsize',16)
xlabel('Discerete time[k]','Fontsize',16)
ylabel('Real states of the system  [meq/l]','Fontsize',16)
grid on
text('Position',[15.0 0.23],'String','x_1','FontName','Times New
     Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[25.0 0.40],'String','x_2','FontName','Times New
     Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[35.0 0.52],'String','x_3','FontName','Times New
     Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[43.0 0.64],'String','x_4','FontName','Times New
     Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[48.0 0.76],'String','x_5','FontName','Times New
     Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
text('Position',[51.0 0.80],'String','x_6','FontName','Times New
     Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%
%-----------------------------------------------------------------------------------------
% Generating random system and measurement noises,
  for i = 1:K
     y(i) = x_bar(6,i);                % declaring the system output vector
  end
%
% v(k) values
  v = 0.1*[0.8138 0.3660 0.7837 0.3020 0.5285 0.9384 0.5108 0.2094 0.7971
0.5887...
       0.0826 0.9439 0.5616 0.0106 0.0780 0.1694 2.0193 0.8469 0.0754 0.0205...
       0.0976 0.0882 0.0727 0.2225 0.0648 0.3569 0.5222 0.4881 0.2899 0.8239...
       0.2754 0.3677 0.4354 0.9323 0.2641 0.5971 0.0110 0.7416 2.0540 0.1016...
       0.0866 0.3303 0.0478 0.8024 0.2693 0.0961 0.8892 0.2134 0.2365 0.5276...
       0.5592 0.9470 0.4634 0.9255 0.2098 0.6307 0.6227 0.8634 0.7727 0.2212...
       0.3024 0.0229 0.2350 0.4300 0.9217 0.5790 0.0671 0.0188 0.2151 0.4629...
       0.3422 0.2880 0.2577 0.0219 0.3543 0.2250 0.4794 0.2197 0.5329 0.1290...
       0.1497 0.3331 0.9176 0.3175 0.8653 0.1700 0.8714 0.2612 0.8580 0.8134...
       0.0915 0.6576 0.5723 0.0319 0.9263 0.6234 0.4390 0.0708 0.6126 0.6460];
%
% w(k) values
 w = 0.1*[0.2803 0.0867 0.0332 0.5675 0.5385 0.8121 0.2635 0.1602 0.4682
0.4661...
      0.7524 0.3665 0.0390 0.6696 0.7095 0.3193 0.0014 0.2600 0.6123 0.8596...
      0.0014 0.2600 0.6123 0.8596 0.5303 0.4197 0.4600 0.5149 0.2736 0.3570...
      0.8432 0.0984 0.1960 0.2775 0.0657 0.6452 0.5168 0.6050 0.1869 0.2458...
      0.3628 0.8714 0.2749 0.8225 0.3675 0.1581 0.1916 0.3287 0.1151 0.8831...
      0.5229 0.3143 0.0643 0.6621 0.4764 0.4518 0.5462 0.9402 0.8961 0.1994...
      0.6241 0.4416 0.0390 0.1396 0.0945 0.5786 0.5754 0.8234 0.2556 0.0549...
      0.1419 0.9391 0.0004 0.7252 0.4241 0.5413 0.9409 0.4511 0.1052 0.1522...
      0.3257 0.0289 0.0321 0.1035 0.0637 0.4469 0.4002 0.5299 0.7142 0.6959...
      0.3999 0.4878 0.7129 0.2904 0.6365 0.4831 0.0704 0.7258 0.7033 0.5189;
  %
      0.3911 0.3367 0.4430 0.5237 0.2742 0.4974 0.1455 0.6746 0.9415 0.1039...
      0.3242 0.6917 0.4335 0.7753 0.7818 0.5363 0.4617 0.3405 0.6010 0.8876...
      0.0503 0.5239 0.1385 0.6332 0.3308 0.2974 0.7899 0.9531 0.1194 0.1391...
      0.0543 0.1359 0.6961 0.6494 0.4659 0.1812 0.1157 0.5081 0.3161 0.5777...
      0.0492 0.7345 0.6268 0.1688 0.2789 0.4803 0.5028 0.7536 0.6620 0.0765...
      0.9674 0.6593 0.2951 0.4722 0.1787 0.8921 0.2441 0.3722 0.2630 0.9319...
      0.2196 0.1305 0.0837 0.1473 0.0506 0.2447 0.7006 0.1453 0.1017 0.0826...
      0.1522 0.6781 0.2726 0.4282 0.1314 0.1403 0.4357 0.8278 0.5526 0.4303...
```

```
    0.0951 0.3050 0.3434 0.0819 0.3515 0.4531 0.5266 0.4590 0.6127 0.3091...
    0.4233 0.3034 0.5620 0.5215 0.8382 0.6330 0.6480 0.2589 0.4953 0.8838;
%
    0.0972 0.4766 0.1289 0.2467 0.4869 0.6560 0.2915 0.9546 0.5147 0.9437...
    0.0388 0.9302 0.5882 0.9604 0.2440 0.4599 0.7266 0.3418 0.3258 0.6148...
    0.9970 0.9932 0.9687 0.6732 0.4717 0.1550 0.4823 0.7175 0.0807 0.0228...
    0.6053 0.5007 0.2416 0.8493 0.0482 0.1626 0.7202 0.0857 0.5112 0.5765...
    0.8378 0.0015 0.0050 0.0980 0.5672 0.2962 0.1220 0.4644 0.3511 0.2086...
    0.4856 0.9539 0.9783 0.1515 0.4512 0.3042 0.2545 0.4827 0.2441 0.2122...
    0.7252 0.3059 0.2236 0.0336 0.0116 0.3378 0.9135 0.3613 0.2333 0.7115...
    0.3563 0.1423 0.6004 0.6239 0.5911 0.4620 0.2812 0.2315 0.0115 0.7964...
    0.9487 0.9000 0.1928 0.4345 0.4558 0.3943 0.4074 0.3878 0.6771 0.0638...
    0.2962 0.9351 0.9333 0.7164 0.0986 0.9532 0.0483 0.2793 0.8657 0.2234;
%
    0.3431 0.2915 0.5661 0.2071 0.8291 0.2149 0.0070 0.4825 1.5198 0.1751...
    0.6815 0.4305 0.9821 0.0254 0.7143 0.3321 0.3414 0.0266 0.8453 0.8291...
    0.3378 0.1035 0.6191 0.4387 0.0137 0.7207 0.1809 0.6044 0.6292 0.6048...
    0.6486 0.9367 0.3417 0.2047 0.5591 0.8385 0.1724 0.1062 1.1373 0.3045...
    0.1276 0.1354 0.3320 0.8728 0.6691 0.6930 0.4037 0.0567 0.3686 0.5212...
    0.7960 0.5885 0.6633 0.0481 0.0668 0.7187 0.5919 0.5170 0.2650 0.0194...
    0.3463 0.3354 0.4814 0.0285 0.9089 0.2464 0.0800 0.6268 0.0508 0.5786...
    0.6188 0.6858 0.1828 0.2249 0.7342 0.9713 0.4935 0.4656 0.5151 0.1722...
    0.9178 0.0391 0.4458 0.2793 0.5221 0.1624 0.4175 0.4793 0.0673 0.8457...
    0.5220 0.5136 0.5696 0.0641 0.4593 0.2312 0.5567 0.9927 0.5858 0.8588;
%
    0.0479 0.5867 0.0963 0.8945 0.7807 0.5681 0.8342 0.3187 0.7923 0.8837...
    0.5444 0.1434 0.7044 0.7105 0.9456 0.5552 0.9653 0.4349 0.3384 0.8158...
    0.9602 0.8903 0.9416 0.8945 0.1970 0.3667 0.6156 0.6511 0.5357 0.5879...
    0.9617 0.3938 0.9216 0.4586 2.2428 0.8875 0.8164 0.5809 0.3350 0.6837...
    0.1834 0.3671 0.3307 0.5121 0.8897 0.6548 0.5808 0.4858 0.3842 0.1800...
    0.0436 3.3466 0.3453 0.2725 0.0607 0.7225 0.8948 0.2058 0.6792 0.1552...
    0.8135 0.4817 0.2882 0.0546 0.3499 0.5955 0.1340 0.1896 0.6326 0.3786...
    0.2599 0.6570 0.1265 0.4208 0.6346 0.2690 0.2499 0.0210 0.8727 0.9295...
    0.7202 0.3758 0.7409 0.1489 0.9068 0.1736 0.2933 0.5159 0.3986 0.5638...
    0.3598 0.9712 0.9685 0.0737 0.1259 0.0313 0.0728 0.2660 0.6031 0.7628;
%
```

```
     0.3700 0.1890 0.3952 0.4279 0.3248 0.8431 0.0974 0.0425 0.0947 0.9046...
     0.3824 0.1351 0.3894 0.2058 0.9172 0.6317 0.6981 0.6810 0.3001 0.6461...
     0.8561 0.4312 0.8194 0.3253 0.2557 0.9680 0.6752 0.2321 0.1298 0.5518...
     0.7139 0.5059 0.5380 0.1109 0.4209 0.2458 0.7202 0.3663 0.4160 0.0302...
     0.9688 0.5358 0.1493 0.5666 0.4539 0.5277 0.1192 0.3004 0.5533 0.8696...
     0.3079 0.4611 0.9794 0.7611 0.4052 0.2746 0.1572 0.0644 0.3737 0.4373...
     0.0221 0.5859 0.4737 0.7803 0.8222 0.3872 0.7674 0.6924 0.8030 0.1741...
     0.2703 0.0814 0.0514 0.4320 0.6748 0.0813 0.5350 0.3564 0.6596 0.4447...
     0.8282 0.8625 0.5585 0.1935 0.4838 0.6050 0.6791 0.2848 0.8247 0.9685...
     0.0137 0.0555 0.0266 0.2215 0.6780 0.3025 0.2967 0.3165 0.6228 0.3034];
%
  zout = (y + v);
%
%---------------------------------------------------------------------------------------------
% Initializing the noise covariance matrices (must be inside iteration, otherwise the
become constant
%
  I = eye(6);                              % identity matrix of 6x6 dimensions
%
%-------------------------------------------------------------------------------------------------
figure(2)
i = 1:K;
subplot(2,1,1)
%at the end of this iteration states should be xER^(NxK)
plot(k,wk,':o','Linewidth',2.5,'MarkerSize',6)
xlabel('Discerete time[k]','Fontsize',13)
ylabel('System noise dynamics[meq/l]','Fontsize',13)
title('System noise dynamic behabiour','Fontsize',14)
grid on
% %-------------------------------------------------------------------------------------------------
subplot(2,1,2)
i = 1:K;
plot(k,vk,':o','Linewidth',2.5,'MarkerSize',5)
xlabel('Discrete time [k]','Fontsize',13)
ylabel('Measurement noise dynamics[meq/l]','Fontsize',13)
title('Measurement noise of the process','Fontsize',14)
```

```
grid on
%---------------------------------------------------------------------------------------------------
% initialization of state estimate
%
x_curr = [5.0 8.0 5.0 5.0 10.0 5.0]';
%
% Initializing a priori state equation variables
% Initializing parameters for start of the calculation
  x_curr_out = x_curr(6,1)
  x_bar_out = x_bar(6,1)
%
  ekk = abs(x_bar_out - x_curr_out)
%
  Pkk = cov(ekk*ekk');                    % covariance P(k,k) at P(0,0)
  if Pkk == 0
       Pkk = 1.0;
   end
  Pkk = Pkk
 %
%--------------------------------------------------------------------------------------------------------
% Initialize the evaluation criteria (calculation time and least square error)
  e0 = cputime;
  Jk1 = 0;
  x_estkk(:,1) = x_curr(:,1);
  x_currk(:,1) = x_curr(:,1);
%
  k = 1;                       % initialize the stopping procedure maximum number
  while k < K
%-----------------------------------------------------------------------------
%  Initial values that need to be inside the iteration index (not outside, values must
      be calculated for every moment not all moments
%  Noises have to be moving as well, if calculated outside the circle, they become
      constants-which is incorrect
  %
    vk = v(k+1);
    Vk = cov(vk*vk');           % calculating Vk = E{v(k)*v(k)^T}
    wk = w(:,k);
    Wk = cov(wk*wk');           % calculating Wk = E{w(k)*w(k)^T}
 %
Vk1 = 0.1405;
 %
   Wk1 = [Wk(1,1) 0 0 0 0 0;...
          0 Wk(2,2) 0 0 0 0;...
          0 0 Wk(3,3) 0 0 0;...
          0 0 0 Wk(4,4) 0 0;...
          0 0 0 0 Wk(5,5) 0;...
          0 0 0 0 0 Wk(6,6)];
%-----------------------------------------------------------------------------------------------------
% Solving the Time Update (Prediction) equation starting with initial values of
    estimated states (x^(k,k)
%
%   k = k ;                                % test condition for current k moment
    zout_k = zout(k);                     %
    FRk = FR(k);                          % control input for each k measurement
    x_estk1k = A*x_estkk + B*x_estkk*FRk + B1*FRk + W*xf;  % calcualte state
                                                   estimate x(k+1,k) for current k moment
%
%-----------------------------------------------------------------------------------------------------
% Determination of the a priori covariance matrix P(k+1,k)
% calculate prediction error covariance matrix P(k+1,k)
   Pk1k = A*Pkk*A' + B*FRk*Pkk*A' + A*FRk*Pkk*B' + B*FRk*Pkk*FRk*B' + Wk1;
%
%-----------------------------------------------------------------------------------------------------
% Measurement Update equation solution; the feedback equation
K_k1 = Pk1k*C'*(inv(C*Pk1k*C' + Vk1));              % calculate the Kalman filter gain
                                                     K*(k+1)
%
x_estk1k1 = A*x_estk1k + K_k1*(zout_k - C*x_estk1k);   % estimated state at
                                                          moment (k+1,k+1)
%
% calculation of the covariance a posteriori matrix
```

```
   Pk1k1 = (I-K_k1*C)*Pk1k;                     % calculate correction error covariance
                                                    matrix P(k+1,k+1)
%-----------------------------------------------------------------------------------
% Updating Time Update equations for the next calculation
%
    xk1k1_curr(:,k) = x_estk1k1;                 % collection of estimates data points
                                                    x^(k+1,k+1) for plotting
    x_currk(:,k) = x_estk1k1;                    % set current estimate to be the next
                                                  estimate
    x_estkk = x_currk(:,k);
  %
    Pkk = Pk1k1;                                 % set P(k,k)for next Time Update calculation to
                                                  current P(k+1,k+1)
  %-----------------------------------------------------------------------------------------------
  % Calculating the error difference between measured and estimated states
  %
    error(:,k) = x_bar(:,k)-x_estk1k1;      % error difference between real & estimated
                                             states
    Gk(:,k) = K_k1;                   % collection of filter gain values for the full trajectory
  %
    Jk = (error(:,k)'*error(:,k));          % calculate the current least square error
    Jk1 = Jk1 + Jk;                         % calculate the improved error
    k = k+1;                                % increase integer count for the next calculation
   end
  e1 = cputime;                             % end-point for processing time
  %
%------------------------------------------------------------------------------------------------
% Display of optimalilty test values (the criterion)
  error = abs(error);                 % final error
  et = e1 – e0                        % overall processing time
  Jk1                                 % final least square error
  FRk = FRk
%
%------------------------------------------------------------------------------------------------
% Plotting  state estimates for the full trajectory
%  %
```

```
 i = 1:K–1;
 figure(3)
 plot(i,xk1k1_curr,':o','Linewidth',2.5,'MarkerSize',5)
 title('Estimated states based on Kalman filter gain','Fontsize',15)
 xlabel('Discrete time[k]','Fontsize',15)
 ylabel('State estimates [meq/l]','Fontsize',16)
 grid on
 text('Position',[10.0 -0.02],'String','x^~_1','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[15.0 0.020],'String','x^~_2','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[20.0 0.025],'String','x^~_3','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[25.0 0.025],'String','x^~_4','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[30.0 0.005],'String','x^~_5','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[50.0 0.05],'String','x^~_6','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%

 figure(4)
 k = 1:K–1;
 plot(k,Gk,'o','Linewidth',2.5,'MarkerSize',5)
 title('Kalman filter gain values over full trajectory','Fontsize',16)
 xlabel('Discerete time[k]','Fontsize',16)
 ylabel('Kalman filter gains for all stages','Fontsize',16)
 grid on
%
%-----------------------------------------------------------------------------------------------------
% % Plotting error difference between measured and estimated states
%
 figure(5)
 j = 1:K–1;
 plot(j,error,'o','Linewidth',2.5)
 xlabel('Discrete time [k]','Fontsize',16)
 ylabel('Error difference [meq/l]','Fontsize',16)
 title('Error difference between measured & estimated states','Fontsize',16)
 grid on
 text('Position',[15.0 0.35],'String','e_x_1','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[25.0 0.45],'String','e_x_2','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[35.0 0.5],'String','e_x_3','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[45.0 0.65],'String','e_x_4','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[55.0 0.70],'String','e_x_5','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
 text('Position',[65.0 0.82],'String','e_x_6','FontName','Times New
        Roman','Fontsize',22,'FontWeight','Bold','FontAngle','Italic')
%
%----------------------------------------------------------------------------------------
% Single plots each plotted on a single graph
figure(5)
i = 1:K
plot(i,x_kk(:,i))
grid on
plot(i,xk1k1_curr(6,i))
title('Standard Kalman Filter State Estimation','FontSize',9)
ylabel('x^6 [meq/l]','FontSize',9)
xlabel('time[s]','FontSize',9)
%
plot(i,xk1k1_curr(5,i))
title('Standard Kalman Filter State Estimation','FontSize',9)
ylabel('x^5 [meq/l]','FontSize',9)
xlabel('time[s]','FontSize',9)
%
plot(i,xk1k1_curr(4,i))
title('Standard Kalman Filter State Estimation','FontSize',9)
ylabel('x^4 [meq/l]','FontSize',9)
xlabel('time[s]','FontSize',9)
```

```
%
plot(i,xk1k1_curr(3,i))
title('Standard Kalman Filter State Estimation','FontSize',9)
ylabel('x^3 [meq/l]','FontSize',9)
xlabel('time[s]','FontSize',9)
%
plot(i,xk1k1_curr(2,i))
title('Standard Kalman Filter State Estimation','FontSize',9)
ylabel('x^2 [meq/l]','FontSize',9)
xlabel('time[s]','FontSize',9)
%
plot(i,xk1k1_curr(1,i))
title('Standard Kalman Filter State Estimation','FontSize',9)
ylabel('x^1 [meq/l]','FontSize',9)
xlabel('time[s]','FontSize',9)
%
%------------------------------------Program end---------------------------------------
```