Cape Peninsula
University of Technology

DEVELOPMENT OF METHODS FOR PARALLEL COMPUTATION OF THE SOLUTION OF THE PROBLEM FOR OPTIMAL CONTROL

by

LITHA MBANGENI

Thesis submitted in fulfilment of the requirements for the degree

Master of Technology: Electrical Engineering

in the Faculty of Engineering

at the Cape Peninsula University of Technology

Supervisor: Professor. Raynitchka Tzoneva

Co-Supervisor: Mr. Carl Kriger

Bellville
September 2010

DECLARATION

I, Litha Mbangeni, 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.

Signed **Date**

ABSTRACT

Optimal control of fermentation processes is necessary for better behaviour of the process in order to achieve maximum production of product and biomass. The problem for optimal control is a very complex nonlinear, dynamic problem requiring long time for calculation. Application of decomposition-coordinating methods for the solution of this type of problems simplifies the solution if it is implemented in a parallel way in a cluster of computers. Parallel computing can reduce tremendously the time of calculation through process of distribution and parallelization of the computation algorithm. These processes can be achieved in different ways using the characteristics of the problem for optimal control.

Problem for optimal control of a fed-batch, batch and continuous fermentation processes for production of biomass and product are formulated. The problems are based on a criterion for maximum production of biomass at the end of the fermentation process for the fed-batch process, maximum production of metabolite at the end of the fermentation for the batch fermentation process and minimum time for achieving steady state fermentor behavior for the continuous process and on unstructured mass balance biological models incorporating in the kinetic coefficients, the physiochemical variables considered as control inputs. An augmented functional of Lagrange is applied and its decomposition in time domain is used with a new coordinating vector. Parallel computing in a Matlab cluster is used to solve the above optimal control problems. The calculations and tasks allocation to the cluster workers are based on a shared memory architecture. Real-time control implementation of calculation algorithms using a cluster of computers allows quick and simpler solutions to the optimal control problems.

ACKNOWLEDGEMENTS

I wish to thank:

- My parents Mandzolwandle Christopher and Fezeka Felicia Mbangeni, sister Zintle Tabisa Mbangeni, my Aunt Noxolo Mbangeni and Ntlonikazi Victoria Ketani for their support, patience endurance and understanding during the study.
- Professor Raynitchka Tzoneva for her invaluable help in my career. I consider myself fortunate to come across such a noble professor. She have shown a great interest in all phases of this project to ensure its success. But for her keen involvement, this thesis would not have been possible. I am thankful to her for assigning me one of the most important research project in the field of Real-Time Distributed Systems.
- Mr Carl Kriger for his energetic, enthusiastic on assisting with the installation of the software
- The Electrical Engineering Controls Group at the Cape Peninsula University of Technology for their assistance in different areas of the research.

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 Family and daughter Kanyisile Njobe

TABLE OF CONTENTS

CHAPTER EIGHT: DESCRIPTION OF THE MATLAB PARALLEL PROGRAMS FOR OPTIMAL CONTROL PROBLEM OF FERMENTATION PROCESSES

CHAPTER NINE: CONCLUSION AND FUTURE RECOMMENDATIONS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

Terms/Acronyms/Abbreviations	Definition/Explanation
ANN	Artificial Neural Network
AISC	Application Specific Integrated Circuit
bcMPI	Blue Collar MPI
COTS	Commodity off- the-shelf
CVP	Control Vector Parameterization
CP	Complete Parameterization
CPU	Central Processing Unit
ConLab	Control Laboratory
CSTBR	Continuous Stirred Tank Bioreactor
DAQ	Data ecqusition
DCT	Distributed Computing Toolbox
DO	Dissolved oxygen
ES	Enzymes substrate
FPGA	Field-Programmable Gate Array
GA	Genetic Algorithm
GPU	Graphics Processing Unit
GO	Global Optimization
HPCS	High Productivity Computing Systems
MATLAB	Matrix Laboratory
MDCE	Matlab Distributed Computing Engine
MDCS	Matlab Distributed Computing Server
MPI	Message Passing Interface
NLP	Nonlinear Optimization Problem
NUMA	Non-uniform Memory Access
PVM	Parallel Virtual Machine
PCT	Parallel Computing Toolbox
QP	Quadratic programming
SMP	Symmetric Multiprocessor
SQP	Sequential Quadratic Programming
UMA	Uniform Memory Access
Dynamic model	Set of differential equations which Predict Time varying performance of process
Model	Set of equations used to describe a process
ParaM	Element is used to pass values to an embedded(usually an embedded program
Online estimation	Estimation of model parameters during the operation of the system
Parameter	A variable whose different values of the observed outputs
Trajectory	A sequence of values collected at equal time intervals

NOMENCLATURE

Symbols / letters		Definition/Explanation
$x \in R$	$[g/l]$	Biomass concentration
$s \in R$	$[g/l]$	Substrate concentration input in fermentor
p^*	$[g/l]$	Product concentration in limitation for the inhibition
$V \in \mathrm{R}$	$[g/l]$	Volume
D	$[g/l]$	Dilution
$p \in R$	$[g/l]$	Product concentration, in the fermentor
F	$[g/l]$	Flow rate of input substrate
μ_{max}	$[g/l]$	Maximum specific growth rate
K_s		Monod constant or simply saturation constant
K_d	$[h]$	Specific death rate
m_s		Maintenance co-efficient
Y_{XS}		Production co-efficient biomass from substrate
Y_{PX}		Production co-efficient product from substrate
Δt	$[h]$	Sampling period for the discrete model
n		Power coefficient
$\upsilon \in R$		Current variable, denoting state and control variables
K		Number of steps in the optimization horizon
J		Performance index
L		Lagrange functional
$v_{\min}, v_{\max}, v = x, s, p, T$		Bound values of constraints
$\lambda_v \in R, v = x, s, p$		Conjugate variables of Lagrange functional
$\mu_v \in R, v = x, s, p$		Penalty coefficients of augmented Lagrange functional
$e_v \in R, v = x, s, p$		Gradients of augmented Lagrange functional According to conjugate variables
$\alpha_v > 0, v = x, s, p$		Steps in the gradient procedures
l		Index of gradient procedures at first level
ε		Values of errors in calculations
$\delta_v, v = x, s, p$		Interconnection in time domain

CHAPTER ONE
INTRODUCTION

1.1 Awareness of the problem

Fermentation processes are widely used in SA for production of food, beverages, medicines, chemicals. A Fermentation process is classified according to the mode that has been chosen for process operation either as batch, fed-batch or continuous (Agrawal *et al.*, 1989). It is clear that the pursuit to maximize the productivity whilst minimizing the production costs is not the only reason for the control engineers interest in fed-batch fermentation processes. Existing control systems for the fermentation processes in the scientific laboratories and industry are based only on the principles of classic control (Flaus *et al.* 1989). The process behavior is not optimized according to existing environmental conditions and the acting disturbances (Ilkova and Tzonkov, 2004).

The non-stationary process dynamics, slow response, non-linearity, sudden unexplained changes and irregularities make these processes more interesting for researchers. The developed project is based on the latest control technology using PC with Matlab software technology. Another challenge is to integrate the whole system i.e. Data ecqusition(DAQ) and real time control system with the design methods for modeling, parameter estimation and optimal control determination, in order to build a fully automatic, adaptive and robust control system (Neil and Harvey, 1990). The problem that will be considered and solved is concerned with the process behavior optimization based on the existing dependencies between the chemical (environmental) process variables like temperature pH, Dissolved oxygen (DO) and the biological variables (concentration of biomass substrate and product) (Zhang and Lennox, 2003). This problem is characterized with many variables, nonlinear equations, big dimensions which lead to very complex calculations. The time for calculation of the optimal control can be very long and in this way the real-time calculation is not possible.

Distributed and parallel computing is one of the new achievements in computer software and hardware. It can reduce tremendously the time of calculation through distribution and parallelization of the computation algorithm. These processes can be achieved in different ways using the characteristics of the problem for optimal control. Solution of the optimal control through mathematical decomposition is one of the best ways for parallelization of the solution because it is done in such a way that the solution of the problem is a sum from the solution of the sub-problem, this means that the optimality is preserved.

Following the above, the thesis will investigate the new computational distributed and parallel technology capabilities to solve the problem for optimal control of fermentation processes on the basis of the decomposition methods.

1.2 Statement of the problem

The problem for control of fermentation process is very important these days because of population increase and industrial developments. The fermentation process has proven to have positive effects in many fields of domestic life and industry, but the different types of fermentation processes that are batch, fed-batch and continuous still need deeper understanding and improvement. Furthermore these processes have not been studied fully yet as an object of control. It is not clear:

- Which kind of models are convenient, for optimal control calculation
- Which are dependencies between input and outputs,
- Which process variables are the most significant and in which way they affect the quality of the process or its products,
- What the optimal combination of settings for these significant variables is, according to the goals posed on the process quality and the product quantity.
- There is a lack of methods for measuring of important variables, for modeling and parameter estimation and for optimization and control calculation.
- The existing methods for solution of the optimal control problems require complex and time consuming calculations.

It is necessary for better control of the process to overcome the above difficulties by developing adaptive multi-layer strategy of control in which the problem for optimal control will be designed and implemented in integrity with others for controller design and its implementation in order to achieve maximum production of yeast at the end of fermentation processes.

Then the research project problem can be stated in the following way.

To develop methods, algorithms and programme for parallel solution of the problem for optimal control such that the process of calculation is characterized with reduced complexity and minimum time.

The research problem can be divided in the following sub problems:

1.3 Design based problems

1.3.1 Sub-problem 1: Mathematical modeling

The design and implementation of the connection between chemical and biological variables in the existing process kinetic model for three types of fermentation processes-batch, fed-batch and continuous. Unstructured model describing the behavior of the biological variables identifying the dependencies between physiochemical and biological variables will be introduced:

Inputs to be considered as control signals

- Base/acid flow rate
- On/off heating energy
- Flow rate of oxygen
- Flow rate of substrate
- Temperature
- pH
- DO_2 concentration.

State variables

- Concentration of biomass
- Concentration of substrate
- Concentration of product
- Volume of the reactor.

1.3.2 Sub-problem 2: Development of decomposition methods

Development of methods and algorithm for solution of the optimal control problems based on decomposition of the initial problem.

1.3.3 Sub-problem 3: Software development for calculation of the problem on one computer.

Development of Matlab programs for optimal control calculation using sequential mode of calculation.

1.3.4 Sub-problem 4: Simulation

Simulation of the process with the developed programme of optimal control to verify the methods and software.

1.3.5 Sub-problem 5: Software development for calculation of the problem on a cluster of computers

Development of Matlab programs for optimal control calculation using distributed mode of calculation.

1.3.6 Optimization of the process of parallel computation

Experiments with the developed programs and a benchmark program to investigate the influence of the cluster architecture, number of workers, amount of communication messages between the workers on the time of calculation. Different scenarios will be designed and implemented.

1.4 Research Aim and objectives

1.4.1 Research Aim

The research is focused on developing a decomposition method to solve the problem for optimal control of the fermentation processes for production of biomass and product. The problem is based on two types of criteria –maximum production of biomass and product at the end of the fermentation process and maximum production of biomass and product at the end of the unknown optimization period. The mass balance model of the process is used. Methods, algorithm and programme have to be developed for calculation of the optimal trajectories of the chemical and biological variables in Matlab. An algorithm and software for parallel and sequential implementation of the method in Matlab cluster has to be developed.

1.4.2 Objectives

- To develop a unique mathematical model of the fermentation processes by incorporating of physiochemical variables into the kinetic modes. This is achieved by performing a study of various existing models and by studying the complex dependencies between the process variables.
- To develop decomposition methods for optimal control calculation of the three considered types of fermentation processes. This is achieved by performing a study of the existing methods for optimal control on the basis of a functional of Lagrange.
- To develop algorithms and programs in Matlab for sequential calculation of the optimal control.
- To develop algorithms and programs in Matlab for distributed-parallel calculation of the optimal trajectories.

1.5 Hypothesis

The hypothesis is connected with the possibilities for application of the methods and technologies of the modern control to the fermentation processes. It can be formulated as:

- The connection between the chemical and biological variables is possibly to be created using special representation of the kinetic model parameters.
- The created models can describe very well the process behavior and the optimal control problem formulated and solved on their basis will optimize both chemical and biological process variables.
- The parallel computation of the optimal control problems will reduce the time for computation.
- The time for calculation in the cluster can be reduced by proper problem decomposition and reduced communication between workers and with possibilities to achieve shorter time for optimal control problem formulation and solution.

1.6 Delimitation of Research

- The research study is conducted at the Cape Peninsula University of Technology in Department of Electrical Engineering.
- The research will be concentrating on all three types of fermentation namely batch, fed-batch and continuous.
- The considered physiochemical variables are temperature, DO, pH and rpm. Only temperature is used in the developed process models.
- Optimal control problem are solved using two-level computational structures and theory of the variational calculus.
- Maximum number of used workers is 32.

1.7 Motivation of the research

Better results from the fermentation process will be received if the control system succeeds to optimize the environmental conditions for the cells. It is therefore quite important to investigate how the chemical state causes changes in biological state. In order to achieve the full biological potential, the chemical or environment conditions must be maintained optimally. The chemical variables control the enzyme activity of the biological variables and consequently control the biological behavior. Therefore, they are considered as control inputs. This means that if the optimal conditions for the chemical variables are reached according to a given optimization criterion, then the optimal conditions for the biological ones will be also reached. The idea is not new but there has never been any practical industrial application of this method. The ability to control fermentation processes accurately by following desired trajectories is believed to be a key factor in maintaining the product consistency and this believe can be proved by the undertaken research project.

The optimal control problems of fermentation processes have to be solved as these processes are subjected to some disturbances. Taking into account the current development of computer architecture, it appears very useful to obtain optimal control law by iterative methods which are well adapted to parallel computation.

1.8 Assumption

The assumptions are made according to different parts of the project research work, as follows:

- Process under study → it is working according to prescribed technology.
- Model of the processes → the kinetic models of batch, fed-batch and continuous can be used as a basis for description of the connections between the chemical and biological variables.
- Optimal control problem solution → The Lagrange's functional methodology can successfully be used to solve nonlinear problems for optimal control
- Parallel calculation → the time for communication between the workers is less than the time for solution of the task by the workers.
- Cluster memory distribution → allows fast distribution of tasks between the workers.

1.9 Description of the fermentation processes

Fermentation is the biotechnological practice that results in the creation of alcohol or organic acids on the basis of growth of bacteria, moulds or fungi on special nutritional media (Ahmed et al., 1982). It is the process that takes place in tanks called bioreactors or fermentors. Fermentation processes are complex, dynamic autocatalytic processes in which transformation, caused by biological catalysts called enzymes is taking place.

Enzymes are proteins of high molecular mass that act as biological catalysts. They are substrate specific, versatile, and very effective biological catalysts resulting in much higher reaction rates as compared to chemically catalyzed reactions under ambient temperature. Enzymes are classified according to the reaction they catalyze. Enzymes decrease the activation energy (E) of the reaction catalyzed by binding the substrate and forming an enzyme-substrate (ES) complex. Molecular aspects of enzymes-substrate interaction are not yet fully understood, but yet the enzymes require optimal conditions for pH, temperature, ionic strength, etc for their maximum activity. (Michael and Fikret, 1992). This dependence will be used in the thesis to develop mathematical models of the processes for the purpose of optimization of the control and state trajectories.

The technology of the fermentation is as follows:

The nutrients are added and the cells are inoculated into the fermentation vessel, environmental conditions such as pH, temperature, DO_2, etc are kept convenient. The cells start multiplying forming mass cells called biomass, and other substances called secondary metabolites, or products. The research is based on three different types of fermentation:

1.9.1 Fed-batch fermentation

The input to the process is continuously added from the initial stage through the end of the process, but the process output is not removed until the process is finished.

The fed-batch processes start with the cells being grown under the batch movement for some time, usually until close to the end of the exponential growth phase. At this point solution of substrate (nutrients) is fed into the reactor, without the removal of culture fluids. This feed should be balanced enough to keep the growth of the microorganisms at a desired specific growth rate and reducing simultaneously the production of by-products. During the growth phase nutrients are added (Flow rate - F) when needed and acid and base is added to control the level of pH. If the cells are grown in an aerobic condition, using appropriate meters the oxygen flow rate can be measured. Also the rate into which the oxygen is supplied can be controlled. Temperature is one of the variables that need to be controlled precisely. Many variables can influence the operation of the fermentation process. The ones listed above are just the few that can be measured on-line.

1.9.2 Batch fermentation

The input (feed-nutrients) to the process is added before the initial stage of the process and the product (process output) is not removed until the process is complete. The main disadvantage of the batch process is that it is characterized by high down time between batches, comprising the charge and discharge of the fermentor vessel, the cleaning, sterilization and the re-start of the whole process.

1.9.3 Continuous fermentation

The input to the process is continuously added and the process output is continuously removed making it more economic compared to the other two type mentioned above.

1.10 Control system layers

The optimal control problem in the thesis will be solved in Matlab as part of the optimal control strategy for the process. In future the Matlab cluster could be connected to the PC and used for faster solution of the optimization problem. The problem for calculation of the optimal control is considered as a part of a real time optimal and adaptive control strategy which could be implemented for every fermentor, Figure 1.1.

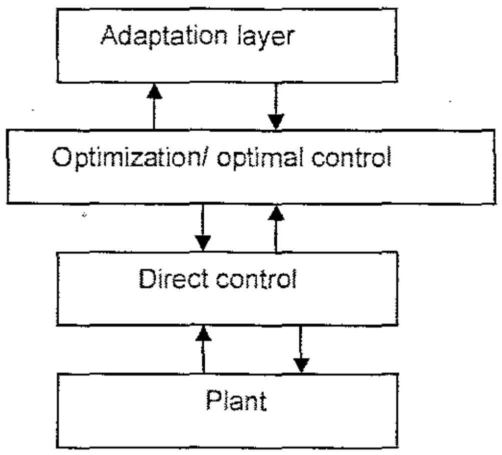


Figure 1.1: Three layers of control system

The problem solved on each layer is as follows:

- Adaptation - model identification, (model parameters estimation)
- Optimization - optimization based on the model parameters from adaptation layer and design of the direct controller parameters. The study will consider and solve optimal control problems.
- Direct control - realization and implementation of the closed loop linear and non-linear control according to optimal trajectories of the process variables obtained from the optimization layer.

1.11 Parallel optimal control calculation

Computational necessities have been one of the main concerns for the optimal control of large-scale interconnected systems. The advent of parallel computers fostered a considerable amount of effort to solve these problems by using parallel algorithms to speed up computation time. The approach that is used in the thesis is based on the idea of decomposition and coordination, where a large optimal control problem is decomposed into a number of sub-problems, and appropriate coordinating variables are introduced. This forms a two-level structure, where the low level consists of many smaller optimal control sub-problems and the high level is a parameter optimization problem.

With the coordinating variable given, low level sub-problems are decoupled and solved in parallel. In the thesis coupling variables and Lagrange multipliers are selected as coordinating variables. The global optimality is then ensured by iteratively updating coordinating variables at the high level. Successful methods for solving long-horizon problems have been developed along the line of time decomposition, where the original problem is decomposed along the time axis (Chang et al, 1989) and (Tang et al 1991) and are further discussed in chapter three of the thesis.

1.12 Thesis Outline

Chapter 1 describes the need of the study discussed in the thesis and highlights comparison between the different types of fermentation processes i.e. batch, fed-batch and continuous processes for the production of yeast. The aim and the objectives of the dissertation are stated and explained.

Chapter 2 describes fermentation process, firstly a brief introduction in microorganism's types, fermentation process phases, and fermentation technology based on three different types of fermentation as the research in the thesis is focusing on them. Models that describe the system in all possible situations with a high accuracy are developed based on the mass balance principle of fermentation processes for three fermentation processes mentioned above.

Chapter 3 describes the importance of fermentation process control technology. It also highlights some important adaptive optimization techniques as they are used as optimization methods needed in order to successfully obtain the optimal operating policies of the processes.

Chapter 4 describes the formulation of the problem for optimal control of a batch fermentation process. Decomposition method for solving discrete time formulation of the problems based on varational calculus using an augmented Lagrange functional and its decomposition in time domain is developed. The algorithm of the method that is used to solve the optimal control problem is described.

Chapter 5 describes the formulation of the problem for optimal control of fed-batch fermentation process. Decomposition method is firstly developed for batch processes to solve the above problems on the basis of an augmented Lagrange functional and new coordinating vector for its time domain decomposition is applied and is developed. The algorithm of the method for solving the optimal control problem is developed.

Chapter 6 describes the problem for minimum time control of continuous fermentation. Decomposition method for solving discrete time formulation of the problems based on varational calculus using an augmented Lagrange functional and its decomposition in time domain is developed. The algorithm of the method that is used to solve the optimal control problem is described.

Chapter 7 is firstly elaborating the reason why there is a need in applying parallel computing in chemical processes. It also explains the concept of parallel computing and gives a brief history of parallel Matlab environment. Matlab Parallel Computing Toolbox and Matlab Distributed Computing Server and its concepts are introduced.

Chapter 8 describes the programme of optimization in Matlab Distributed Computing Engine and Parallel Computing Toolbox. The Results from calculations with the Matlab sequential program and the parallel calculations also are described. The results from the sequential and parallel computing are compared.

Chapter 9 presents the conclusion stressing the developments in the dissertation as well as the future work on the topic and the possible application of the developed work

1.13 Conclusion

This chapter explains the necessity of the research. Aim and the objectives of the thesis are stated and explained. The selected development and simulation software used is discussed. Three different types of fermentation processes i.e. batch, fed-batch and continuous are highlighted. Description of modeling approaches as applicable to the thesis is discussed in chapter two. Also the derivation of the developed fermentation models using mass balance equations and rate laws is discussed and presented in chapter two.

CHAPTER TWO
DESCRIPTION AND PROCESS MODELING OF FERMENTATION PROCESSES

2.1 Introduction

This chapter describes growth stages of the processes in nutrient solution with microorganism's cultivation under physiological conditions. Three different types of fermentation processes i.e. batch, fed-batch and continuous are discussed and compared according to their characteristics. The description of modeling approaches as applicable to the thesis and the comparison between different types is highlighted. The derivation of the developed fermentation models using mass balance equations and rate laws is discussed and presented.

In this chapter section 2.1 recapitulates the content of the chapter. Section 2.2 describes fermentation process, firstly a brief introduction in microorganism's types, fermentation process phases, and fermentation technology based on three different types of fermentation as the research in the thesis is focusing on them. Modeling of fermentation processes has always been somewhat challenge. However, it is neither necessary nor desirable to construct comprehensive mechanistic process models that can describe the system in all possible situations with a high accuracy as presented in section 2.3. In section 2.4 three modeling approaches are highlighted. In section 2.5 describes kinetic models of fermentation processes. In section 2.6 mass balance principle of fermentation processes is presented. Lastly conclusion is given in section 2.7.

2.2 Fermentation process

2.2.1 Microorganism types

Fermentation is the oldest of all biotechnological processes. The term is resulting from the Latin verb *fevere*, to boil—the outer shell of fruit extracts or malted grain acted upon by yeast, during the production of alcohol. Microbiologists consider fermentation as 'any process for production of product by means of mass culture of micro-organisms' and Biochemists consider fermentation as 'an energy generating process in which organic compounds act both as electron donors and acceptors(Pumphrey and Julien, 1996). Fermentation process plays a key role in the pharmaceutical, biotechnology and food industries. Several species belonging to the following category of micro-organisms are used in fermentation processes namely

- Prokaryotic
  Comprise unicellular organisms able of self duplication or of directing their own replication. They do not possess a true nucleus or a nuclear membrane.
- Eukaryotic
  They have nuclear enclosed within a distinct nuclear membrane.

Classification of microorganisms (Pumphrey and Julien, 1996) is shown below on Figure2.1.

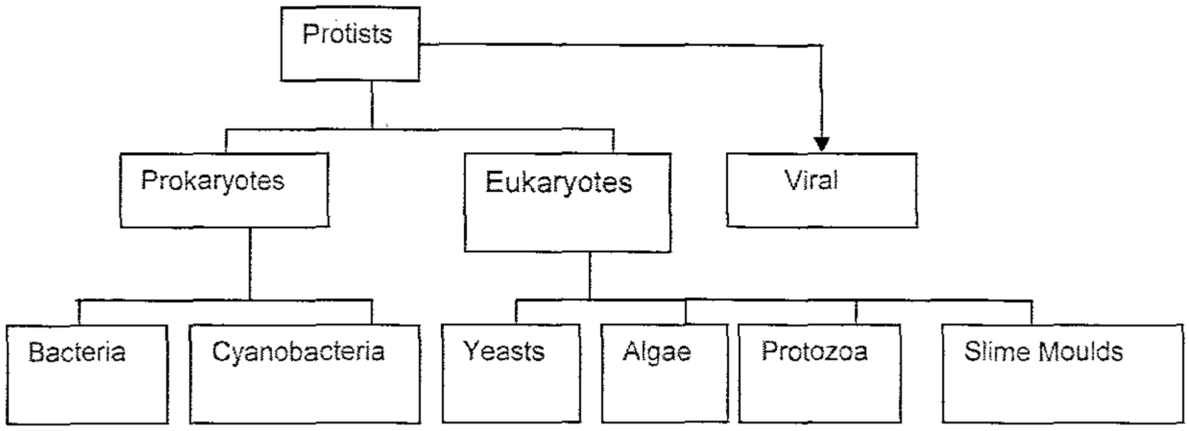


Figure 2.1: General Classification of microorganisms

The non-cellular protists do not undertake self-reproduction; instead they direct their reproduction within another cell term the host. Cyan bacteria are classified another group of microorganisms, although they are often considered to be included with other bacteria. Fungi are divided into lower fungi, slime moulds and higher fungi which comprise yeasts. Yeasts are free-living, single cells which they closely resemble fungi. Protozoa are living, organisms which although not generally engaged for biotechnology processes have been included for comprehensiveness. Viruses can be regarded as intracellular parasites.

Growth of organisms involve multifaceted energy based processes. The rate of growth of micro-organisms is needy upon more than a few culture conditions, which should supply for the energy required for various chemical reactions. The speed of production of micro-organisms and hence the fusion of a range of chemical compounds under artificial culture, requires organism specific chemical compound as the reproduction medium. The kinds and relative concentrations of ingredients of the medium, temperature, pH, dissolved oxygen, flow rate, volume etc pressure microbial growth.

2.2.2 Fermentation process phases

After the inoculation of a nutrient solution with microorganisms and cultivation under physiological conditions, four phases of growth can be observed.

- Lag phase

Physiochemical equilibrium between microorganisms and the environment inoculation with very little growth.

- Log phase

At the end of the lag phase cells have modified to the new environment of growth. Growth of the cell mass is now described quantitatively as a repetition of cell number per unit time for bacteria and yeasts. Plotting the number of cells, biomass against time on a semi logarithmic graph, a straight line will results. Although the cells adjust the medium through uptake of substrate and emission of metabolic products, the growth of cells remain steady at this phase and is independent on substrate concentration as long as surplus substrate is there.

- Stationary phase

As soon poisonous substances have been created, growth slows down. The biomass increases steadily, although the composition of cells may alter. Various metabolite created in this phase are often of biological interests.

- Death phase

In this stage the energy reserves of the cells are worn out. A straight line may be obtained when a semi logarithmic plot is made of survivors versus time, indicating that the cells are dying at an exponential rate. The length of time between the stationary phase and death phase depend on the microorganism and the process used. Fermentation is episodic at the end of the log phase or before death phase begins. All four growth phases are observed on Figure 2.2

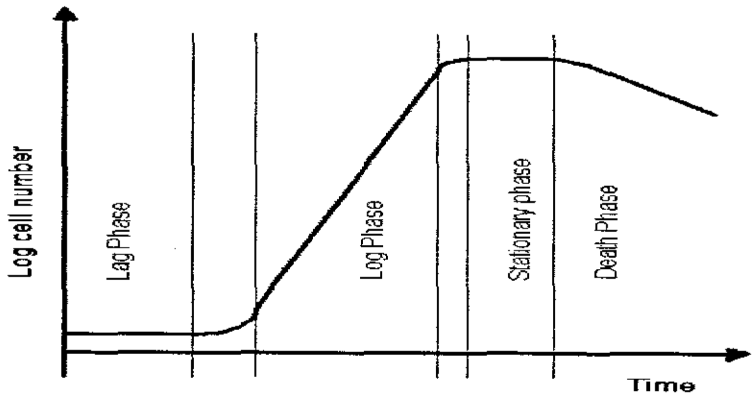


Figure 2.2: Growth curve of a bacterial culture

In industry fermentation is a process that takes place in tanks called bioreactors or fermentors. A typical industrial fermentation and typical process measurements that are available are shown on Figure 2.3.

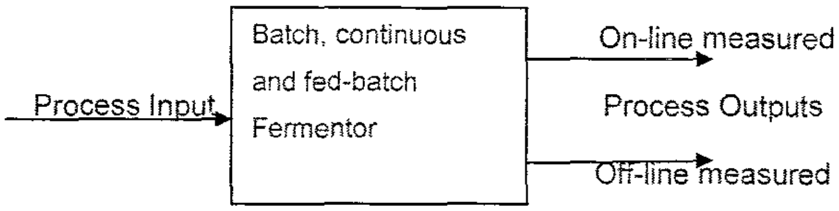


Figure 2.3: Typical industrial fermentation

It is important to distinguish between those process variables measurements that can be made off-line and on-line. Process inputs are substrate flow-rate, Heat on or off, etc. On-line environmental control are, temperature, pH, etc), and off-line environmental control are, biomass concentration, product concentration and substrate concentration. Off-line measurements are made by process operators taking a sample of broth and awaiting results from laboratory before making corrections to the process inputs. The fermentation process can be operated in either batch, fed-batch or continuous fermentors.

2.2.3 Comparison of fermentation processes

Fed-batch bioreactors have a number of well known advantages over batch or continuous fermentors (Banga et al., 2002). For example, fed-batch can be the best (or even only) alternative due to its effectiveness in overcoming undesired effects like substrate inhibition and catabolite repression. Besides, it provides better control of deviations in the organism's growth pattern and production of high cell densities can be possible due to extension of process time, which is useful for the production of substances associated with growth. An illustrative example of the importance of the fed-batch culture is the large scale production of monoclonal antibodies, with product revenues in USA of several billions of USD (Bibila and Robison, 1994).

2.3 Modeling of fermentation processes

Fermentation processes are dynamic, non-linear and non stationary, in nature, thus leading to many difficulties in modeling. Modeling of biological processes has always been somewhat challenge. However, it is neither necessary nor desirable to construct comprehensive mechanistic process models that can describe the system in all possible situations with a high accuracy (Galvanauskas et al., 1998). In order to optimize a real biotechnical production process, the model must be regarded as a vehicle to reach more easily to the final aim. The model must describe those aspects of the process that significantly affect the process performance. Modeling must be leaning at the optimization task to be solved and must not uncritically sample all the information available about the microbial system used. From the theoretical point of view, the model describes the possible paths of the process through state-space representation. The system is non-linear and time varying, state equation are written as

$$x(t) = a(x(t), u(t), t) \tag{2.1}$$

Where $x(t)$ is the state vector and $u(t)$ is the control vector. The easiest modeling concept presumes ideal mixing, which is rarely, if ever achieved. The asepticity necessity create measurements from the fermentor non-trivial. Cellular process, mass transfer and control aspects make the modeling appears more devastating, and thus it should cautiously be considered, what actually needs to be modeled given exact problem. Different techniques have been developed in recent years for different modeling needs. Some focus on the reactor, some on reaction kinetics (Eungdamrong and Iyengar, 2004), and some on system biology: metabolomics, proteomics (Goesman et al., 2003; Klein et al., 2002; Lemerle et al., 2005). Lack of specific sensors makes certain variables unreachable by on-line measurements. The difficulties experienced by control engineers when studying these types of processes, result from the fact that the process model is based on nonlinear algebraic relations and differential equations. These difficulties make the biotechnological process an excellent field for application of advanced control (Dahhou et al., 1991a).

2.4 Modeling Approaches

To be able to control an industrial process it is required to know the relationship between the unprocessed material, process settings and end-product results (Jorgensen and Naes, 2004). Measured input parameters can be mathematically modeled to end product attributes, without any prior knowledge or hypothesis of their contribution to those end-product attributes. The use of predictive modeling methodologies can be used to build predictive models for target responses (Estrada-Flores et al., 2006). There are three modeling approaches that were studied in the manufacturing of dairy products (Peter Roupas, 2008) namely

- White box models
- Black box models
- Grey box models

2.4.1 White box models

White box modeling approaches are described as mechanistic, first principle, hypothetico-deductive, phenomenological, or state-space modeling. These approaches are based on the knowledge and fundamental theoretical principles of the process or factor being modeled and universal equations can be applied to build the model. Such models are applicable over any range of conditions or which model assumptions hold. Values of the input data are independent of measured and the accuracy of the model and the validity of its assumptions and underlying theory, are tested against the real-world measurements of the modeled process (Estrada-Flores et al., 2006).

2.4.2 Black-box models

Black box modeling approaches are empirical, inductive or input/output modeling and are useful when the mechanisms underlying a systems behavior are unknown or poorly understood, or where known mechanisms are too complex to be modeled efficiently from first principle. Examples of this approach include multiple regression models or Artificial Neural networks (ANN) models that require no theoretical basis. Black box models are developed from actual world information (Estrada-Flores et al., 2006) and have been proven to be particularly helpful where serious parameters are identified and measured on-line (O'Callaghan and Cunningham, 2005).

2.4.3 Grey-box model

A grey box, or hybrid modeling approach combines the use of first principle based white-box models and database black box models, and is particularly useful when there is a lack of fundamental theory to describe the system, or process being modeled, when there is a limited experimental data for validation, or when there is a need to decrease the complexity of the model. This model approach has been used to create mathematical models to predict the shelf life of chilled products or the thermal behavior of imperfectly mixed fluids or to create Artificial Neural Networks (ANNs) and dynamic differential equations for control-related applications (Estrada-Flores et al., 2006). It has an advantage over white box in that it provides a way around missing information in a complex system and performance from a control point of view (O'Callaghan and Cunningham, 2005). It has been reported that such grey-box solution have not perform the separate white and black models in terms of predictive precision for explicit applications (Jones and Brown, 2007). The grey box modeling gives opportunities to illustrate in a better way the performance of the fermentation processes.

The capability to predict the performance of fermentation systems enhances the possibility of optimizing their performance. Mathematical expressions of model systems symbolize a tool for this and the most modern advances in computer hardware and software have made the approach more helpful than previous basic attempts. The current information of biochemical microbial pathways and the experience in optimization of chemical fermentors combined with very powerful and easily reached computers, loaded with easy to use software and mathematical routines, are changing the way processes are being developed and operated. This possibility is investigated in the thesis by development of mathematical models and decomposition methods for calculation of optimal control of all three types of the fermentation processes using parallel computation in a cluster of computers.

2.5 Kinetic models

Kinetic modeling expresses the verbally or mathematically expressed correlation between rates and reactant/product concentrations, that when inserted into the mass balances, permits a prediction of the degree of conversion of substrates and the yield of individual product at other operating conditions(Nielsen and Villadsen, 1992). Kinetic model is normally a black box model and can be shown diagrammatically by the structure on Figure 2.4.

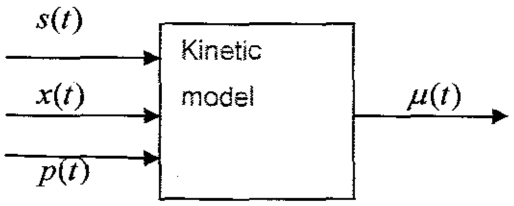


Figure 2.4: Kinetic model structure

Where s(t) is substrate concentration, p(t) is a product concentration, x(t) is a biomass concentration and µ(t) is the growth rate of microorganisms. If the rate expressions are correctly set up, it may be possible to express the course of an entire fermentation experiment based on the initial values for the components of state vector, e.g., concentration of substrate. The basis of modeling is to express functional relationship between the forward reaction rates of the reactions considered in the model and the concentration of the substrates, metabolic products and biomass:

$$\mu_i = f_i(s, p, x) \tag{2.2}$$

Where μ_i is the forward reaction rate. A good understanding of the kinetics in terms of mathematical modeling is an important step in developing a successful control strategy. It is often possible to accurately represent the growth of microorganism, which is an unusually complex phenomenon by relatively simple empirical models. One of the most important factors influencing the growth of *Sachoromyces cerevisiae* is the effect of specific inhibitors. During ethanol fermentation, in general both product and substrate inhibitions can be considered as significant. An extensive literature review on alcohol tolerance of the microorganisms has been presented by (Casey and Ingledew, 1986). (D'Amore and Stewart, 1987), (Jones, 1989), (Pamment, 1989) and (Warren et al, 1992) have also presented a review of literature on ethanol inhibition. Ethanol inhibition has been modeled in general by its effect on two parameters:

- The specific growth rate of the microorganisms
- The specific product formation rate

Both models can be well thought-out as important in the case of batch fermentations since these models can predict ethanol production in the lack of growth of the microorganism

Kinetic models are investigationally derived mathematical formulas that fit the measured data reasonably well. The easiest kinetic modeling practice is the application of an unstructured kinetic model. The kinetics can be linear or nonlinear, single-phase or multiple phase (Eungdamrong and Iyengar, 2004), (Mitchel et al., 2004). Linear kinetic models take account of constant rate and first order kinetics. Nonlinear kinetic models comprise exponential, logistics, second order other defined functions (Eungdamrong and Iyengar, 2004), (Maurer and Rittmann, 2004) and (Mitchel et al., 2004).

The mainly accepted defined kinetic model function has certainly been the Monod model, which contains two parameters that define the relation of growth and substrate utilization. The Monod model was the foremost unstructured mechanistic model. It can be considered as the mainly basic model to describe the growth of a given microorganism in the absence of any specific inhibition effects. This model is based on the hypothesis that the growth is restricted by a single substrate concentration and is given by:

$$\mu = \frac{\mu_{\max} s}{K_s + s} \tag{2.3}$$

Where K_s is the Monod constant, μ_{max} is the maximum specific growth rate and s is the substrate concentration [*g/l*]. The importance of K_s is that when substrate concentration is numerically equal to K_s growth rate is exactly half of the maximum growth rate. In many systems the product inhibits the rate of growth. The equation that is used to represent the product inhibition on the growth rate is (Dochan et al., 1990

$$r_g = \left(1 - \frac{p}{p^*}\right)^n \left(\frac{\mu_{\max} xs}{K_s + s}\right) \tag{2.4}$$

Where r_g is the cell growth rate, x is the cell concentration, p^* is the product concentration at which all metabolism cease and n is the empirical constant for *Saccharomyces cerevicerae*. The investigations on the thesis are based on the kinetic model (2.4). Maximum specific growth rate is expressed as function of ethanol concentration. In other words the overall effect of ethanol inhibition on growth rate can be accounted by adjusting growth rate in above. Contois and Moser (Kiviharju et al., 2007) presented modifications to the Monod model. Contois theory was base on increase inhibition by biomass itself, which has later been doubted (NieLsen and Villadsen, 1994). Other different formulas were presented by Teisser and Blackman (Kiviharju et al., 2007). The logistics expression has also been successfully used in the growth estimations. The Monod model has also been extended to substrate and product inhibition (NieLsen and Villadsen, 1994). These, however have more parameters and require more computing power to fit appropriately. Kinetic models have been applied to the lag and death phases and in particle interactions (Mitchel et al., 2004). Temperature and PH can change the activities of many enzymes and influence the growth kinetics (Maurer and Rittmann, 2004), (Mitchel et al., 2004) and (NieLsen and Villadsen, 1994). The above mentioned models are all empirical expressions, it is futile to debate which one is the best, simply choose the one that best fits the system. Some of the kinetic models are given below:

Tessier model

$$\mu = \mu_{\max}(1-(e)^{-s/Ks}) \tag{2.5}$$

Where s is the concentration of substrate

Moser model

$$\mu = \frac{\mu_{\max} s^n}{K_s + s^n} \tag{2.6}$$

Where n is the empirical constant for *Saccharomyces cerevicerae*

Contois model

$$\mu = \frac{\mu_{\max} s}{K_s x + s} \tag{2.7}$$

Blackman model

$$\mu = \begin{cases} \mu_{\max}\frac{s}{2K_s}; s \le 2K_s \\ \mu_{\max} \qquad s \ge 2K_s \end{cases} \tag{2.8}$$

2.6 Mass balance principle of fermentation processes

In modern approaches to fermentation control, a reasonably accurate mathematical model of the reaction and reactor environment is required. Using process models, progress beyond environmental control of bioreactors into the realm of direct biological control can be achieved. Development of fermentation models is aided by information from measurements taken during the process operation. Models are mathematical relationship between variables. For biological processes, specifying the model structure can be very difficult because of the complexity of cellular processes and the large number of environmental factors which affect the cell culture.

Process models vary in form but have the unifying feature, that they predict outputs from set of inputs. Another important factor affecting reactor performance is the mode of operation such as batch, fed-batch and continuous. Choice of operating strategy has an effect on substrate conversion, product concentration, susceptibility for contamination and process reliability. The dynamic behavior of the growth of one population of microorganisms on a single limiting substrate in a stirred tank reactor is most often expressed by equations (Dochan and Bastin, 1990) which are obtained from straight forward mass balances:

$$\boxed{\begin{array}{l}\text{The rate of}\\\text{mass in}\\\text{through}\\\text{system}\\\text{boundaries.}\end{array}} + \boxed{\begin{array}{l}\text{The rate of}\\\text{mass out}\\\text{through}\\\text{system}\\\text{boundaries}\end{array}} + \boxed{\begin{array}{l}\text{The rate of}\\\text{Mass}\\\text{generated}\\\text{within}\\\text{system.}\end{array}} - \boxed{\begin{array}{l}\text{The rate of}\\\text{mass}\\\text{consumed}\\\text{within}\\\text{system.}\end{array}} = \boxed{\begin{array}{l}\text{The rate of}\\\text{Mass}\\\text{accumulated}\\\text{within}\\\text{system.}\end{array}} \quad (2.9)$$

Mathematically (2.9) can be written by the following equation

$$\frac{dM(t)}{dt} = M_i(t) - M_o(t) + R_G(t) - R_C(t), M(0) = M_\phi \text{ where } t \text{ is the time} \qquad (2.10)$$

M = mass of component in the vessel

M_i = mass flow rate of component entering the vessel

M_o = mass flow rate of component leaving the vessel

R_G = mass rate of generation of component by reaction

R_C = mass rate of consumption by reaction

M_ϕ = Initial mass of the component in the vessel

The mass balance equation can be written for a separate component, or for the whole process. This equation is used to develop the unstructured models of the different types of fermentation processes. Three equations form the corresponding model of the fermentation processes. They describe the mass balances for the biomass, substrate and the product. Equations are derived under the following assumptions

- Stirred tank reactor
- The process is assumed to be in complete mixed conditions; this implies that the composition of the medium is homogeneous in the reactor.

- The biomass growth and the substrate consumption are propositional to the biomass concentration and are described below for every type of fermentation process.

2.6.1 Models for Batch fermentation process

Batch processes operate in a closed system; substrate is added at the beginning of the process and product is removed only at the end of the process. It is assumed that the reactor has no leaks and evaporation meaning the volume of the vessel V is constant.

- **Derivation of the biomass equation**

Assuming well mixed batch fermenter for cell culture or biomass concentration x and applying mass balance equation (2.10), it can be written

The reactor has no input and output which means that

$$M_i = M_o = 0 \tag{2.11}$$

Mass of cells in the reactor is equal to the product of the concentration of the biomass and volume.

$$M = xV \tag{2.12}$$

Mass rate of cell growth R_G is equal to r_xV

$$R_G = r_xV \tag{2.13}$$

Where r_x is equal to μx, r_x is the volumetric rate of growth, μ is the specific growth rate of the microorganisms, which can be expressed by Monod kinetic law in (2.3).

If cell death takes place in the reactor, the mass rate of consumption of biomass is given by

$$R_C = r_dV \tag{2.14}$$

Where $r_d = k_dx$ is the death of microorganisms and k_d is the death constant. Then following equation (2.10) where

$$M = xV \tag{2.15}$$

$$\frac{d(xV)}{dt} = \mu xV - k_dxV, \quad x(0) = x_\phi \tag{2.16}$$

Taking derivatives of equation (2.16)

$$\frac{xd(V)}{dt}+\frac{Vd(x)}{dt} = xV(\mu - k_d), x(0) = x_\phi \tag{2.17}$$

Since the volume of the reactor is constant, $\frac{dV}{dt}=0$ and the mass balance equation for the biomass is

$$\frac{d(x)}{dt} = x(\mu - k_d), \quad x(0) = x_\phi \tag{2.18}$$

- **Derivation of the mass balance equation for the limiting substrate**

The reactor has no input and output which means that mass flow rate entering the reactor is equal to the mass flow rate leaving the reactor and equal to zero, which is given by equation (2.11). The mass of substrate in the reactor is

$$M = sV \tag{2.19}$$

Where s is the substrate concentration and V the volume of the Reactor

No substrate generation in the reactor $R_G = 0$. The substrate is consumed by the microorganisms in the reactor

$$R_C = r_s V \tag{2.20}$$

Where r_s is the rate of consumption, which depends on the specific growth rate, the yield coefficient Y_{XS} and the coefficient of maintenance of microorganisms m_s and biomass concentration x

$$r_s = -\left(\frac{\mu}{Y_{XS}} + m_s\right)x \tag{2.21}$$

Then

$$R_C = -\left(\frac{\mu}{Y_{XS}} + m_s\right)xV \tag{2.22}$$

Applying equation (2.10) the substrate mass balance equation describes only substrate consumption

$$\frac{d(sV)}{dt} = -\left(\frac{\mu}{Y_{XS}} + m_s\right)xV \tag{2.23}$$

Taking derivatives of equation (2.23)

$$\frac{d(s)}{dt} = -\left(\frac{\mu_{\max} s}{Y_{XS}(K_s + s)} + m_s\right)x, \qquad s(0) = s_\phi \tag{2.24}$$

- **Derivation of mass balance equation for the product concentration**

The reactor has no input and output which means that mass flow rate entering the reactor is equal to the mass flow rate leaving the reactor and equal to zero, which is given by equation (2.11).

The mass of product in the reactor is

$$M = pV \tag{2.25}$$

No product consumption in the reactor

$$R_C = 0$$

Generation of the product in the reactor is

$$R_G = r_p V \tag{2.26}$$

Where r_p is equal to $q_p x$, r_p is the volumetric rate of product formation, q_p is the specific rate of product formation.

Applying equation (2.10) for the product mass balance equation

$$\frac{d(pV)}{dt} = q_p xV \tag{2.27}$$

The specific rate of product formation can be expressed as the sum of the maintenance coefficient m_p and the yield coefficient for the production of product by biomass Y_{PX}

$$q_p = (Y_{px}\mu + m_p) \tag{2.28}$$

Then the product equation is

$$\frac{d(p)}{dt} = \left(\frac{Y_{px}\mu_{\max} s}{(K + s)} + m_p\right)x, \qquad p(0) = p_\phi \tag{2.29}$$

On the basis of the above derivation, the mass balance model of the batch fermentation process is

$$\dot{x} = \left(\frac{\mu_{\max} s}{(K_s + s)} - k_d \right) x, \qquad x(0) = x_\phi \tag{2.30}$$

$$\dot{s} = -\left(\frac{\mu_{\max} s}{Y_{XS}(K_s + s)} + m_s \right) x, \quad s(0) = s_\phi \tag{2.31}$$

$$\dot{p} = \left(\frac{Y_{px}\mu_{\max} s}{(K_s + s)} + m_p \right) x, \quad p(0) = p_\phi \tag{2.32}$$

2.6.2 Models for Fed-batch fermentation process

The fed-batch process is characterized with substrate input, no output and changeable volume V because of the input flow. The volume is limited by the vessel volume of the reactor and this means that the input flow can be used with limitations.

- **Derivation of the biomass mass balance equation**

The output does not exist and Mo= 0, the input exists and is given by the flow rate F_i. The mass of the biomass coming into the reactor is given by

$$M_i = Fx_i \tag{2.33}$$

Where x_i is the concentration of biomass into the inflow. The mass of biomass in the reactor is

$$M = xV \tag{2.34}$$

The consumption of biomass in the reactor is given by the death of microorganisms

$$R_C = r_d V = k_d x V \tag{2.35}$$

The generation of microorganisms is given by

$$R_G = r_x V = \mu x V \tag{2.36}$$

Substitution of the above equations (2.33), (2.34), (2.35) and (2.36) into the general mass balance equation (2.10) gives

On the basis of the above derivation, the mass balance model of the batch fermentation process is

$$\dot{x}=\left(\frac{\mu_{\max} s}{(K_s+s)}-k_d\right)x, \qquad x(0)=x_\phi \tag{2.30}$$

$$\dot{s}=-\left(\frac{\mu_{\max} s}{Y_{XS}(K_s+s)}+m_s\right)x, \quad s(0)=s_\phi \tag{2.31}$$

$$\dot{p}=\left(\frac{Y_{px}\mu_{\max} s}{(K_s+s)}+m_p\right)x, \quad p(0)=p_\phi \tag{2.32}$$

2.6.2 Models for Fed-batch fermentation process

The fed-batch process is characterized with substrate input, no output and changeable volume V because of the input flow. The volume is limited by the vessel volume of the reactor and this means that the input flow can be used with limitations.

- **Derivation of the biomass mass balance equation**

The output does not exist and Mo= 0, the input exists and is given by the flow rate F_i. The mass of the biomass coming into the reactor is given by

$$M_i = Fx_i \tag{2.33}$$

Where x_i is the concentration of biomass into the inflow. The mass of biomass in the reactor is

$$M = xV \tag{2.34}$$

The consumption of biomass in the reactor is given by the death of microorganisms

$$R_C = r_d V = k_d x V \tag{2.35}$$

The generation of microorganisms is given by

$$R_G = r_x V = \mu x V \tag{2.36}$$

Substitution of the above equations (2.33), (2.34), (2.35) and (2.36) into the general mass balance equation (2.10) gives

$$\frac{d(xV)}{dt} = Fx_i + \mu xV - k_d xV, \qquad x(0) = x_\phi \tag{2.37}$$

By differentiation, equation (2.37) becomes

$$\frac{xd(V)}{dt} + \frac{Vd(x)}{dt} = Fx_i + xV(\mu - k_d), \qquad x(0) = x_\phi \tag{2.38}$$

The derivative of the volume is equal to the input flow rate

$$\frac{dV}{dt} = F \tag{2.39}$$

Dividing the obtained equation by the volume V it is obtained that

$$\frac{d(x)}{dt} = \frac{Fx_i}{V} + x(\mu - k_d - \frac{F}{V})$$

But dilution rate $D = \frac{F}{V}$ (2.40)

Then equation (2.38) becomes

$$\frac{d(x)}{dt} = Dx_i + x(\mu - k_d - D), \; x(0) = x_\phi \tag{2.41}$$

- **Derivation of the limiting substrate mass balance equation**

The fermenter has no output and the output mass flow rate is given by the equation

$M_o = 0$.

The substrate is not generated in the fermenter and $R_G = 0$. The substrate consumption is given by

$$R_C = r_s V = -\left(\frac{\mu}{Y_{xs}} + m_s\right) xV \tag{2.42}$$

The mass of the substrate in the fermenter

$$M = sV \tag{2.43}$$

The rate of mass inflow into the fermentor is

$$M_i = Fs_i \tag{2.44}$$

Where s_i is the concentration of the input substrate

Applying the general mass balance equation (2.10), the mass balance equation for limiting substrate is.

$$\frac{d(sV)}{dt} = Fs_i - \left(\frac{\mu}{Y_{XS}} + m_s\right)xV, \quad s(0) = s_\phi \tag{2.45}$$

After differentiation equation (2.45) becomes

$$\frac{sd(V)}{dt} + \frac{Vd(s)}{dt} = Fs_i - \left(\frac{\mu}{Y_{XS}} + m_s\right)xV, \; s(0) = s_\phi$$

Dividing by the volume V then the mass balance equation for limiting substrate is

$$\frac{d(s)}{dt} = D(s_i - s) - \left(\frac{\mu}{Y_{XS}} + m_s\right)x, \; s(0) = s_\phi \tag{2.46}$$

- Derivation of the mass balance equation for the product

The input mass rate is given

$$M_i = Fp_i \tag{2.47}$$

Where p_i is the concentration of the input product and the fermenter has no output M_o = 0.

The rate of mass of the product generation is

$$R_G = r_g V = Y_{PX}\mu xV \tag{2.48}$$

The mass of the product in the fermenter is

$$M = pV \tag{2.49}$$

After substitution in the general mass balance equation and after some algebraic transformations the obtained mass balance equation for the product is

$$\frac{d(p)}{dt} = -D(p - p_i) + Y_{PX}\mu x, \; p(0) = p_\phi \tag{2.50}$$

On the basis of the above derivation, the mass balance model of the fed-batch fermentation process is

$$\frac{d(x)}{dt} = \left[\frac{\mu_{\max} s}{(K_s+s)} - k_d - D)\right]x, \; x(0) = x_\phi \tag{2.51}$$

$$\frac{d(s)}{dt} = D(s_i - s) - \left(\frac{\mu_{\max} s}{Y_{XS}(K_s+s)} + m_s\right)x, \; s(0) = s_\phi \tag{2.52}$$

$$\frac{d(p)}{dt} = -D(p - p_i) + \left(Y_{PX}\frac{\mu_{\max} s}{(K_s+s)}x\right), \; p(0) = p_\phi \tag{2.53}$$

$$\frac{dV}{dt} = F, \; V(0) = V_\phi \tag{2.54}$$

Normally it is accepted that the input concentration of the biomass and the product are zero, $p_i = 0$, $x_i = 0$ and the input concentration of the substrate is bigger than the concentration of substrate in the fermenter $s_i > s$.

2.6.3 Models for Continuous fermentation process

In continuous process model nutrients are added continuously and also product is continuously removed. The rate of fed flow is assumed to be equal to the rate of the out flow such that there will be no change in volume. Models are derived using both dynamic and steady state conditions.

2.6.3.1 Dynamic-state balances are:

- **Derivation of the equation for biomass mass balance**

The rate of biomass inflow to the fermenter is

$$M_i = Fx_i \tag{2.55}$$

The rate of the output mass is

$$M_o = Fx \tag{2.56}$$

Where the output concentration of the biomass is equal to the biomass concentration in the fermentor. The generation of biomass concentration in the fermenter is given by equation (2.36). The consumption of biomass is given by the death of microorganisms presented by equation (2.35). The mass into the fermenter is given by equation (2.34). After substitution in the general mass balance equation (2.10) the following is obtained

$$\frac{d(xV)}{dt} = Fx_i - Fx + \mu xV - k_d xV \tag{2.57}$$

After differentiation

$$\frac{xd(V)}{dt} \frac{Vd(x)}{dt} = Fx_i - Fx + \mu xV - k_d xV, \quad x(0) = x_\phi$$

Since the volume of the fermenter is constant $\frac{dV}{dt} = 0$ and the dilution rate is $D = \frac{F}{V}$. It is obtained

$$\frac{d(x)}{dt} = Dx_i + x(\mu - k_d - D), \quad x(0) = x_\phi \tag{2.58}$$

- **Derivation of the mass balance equation for the substrate**

The mass of the substrate in the fermenter is given by equation (2.43). The rate of the input mass of the substrate is given by equation (2.44). The output rate of mass of substrate is

$$M_o = Fs \tag{2.59}$$

The generation of substrate is zero $R_G = 0$. The consumption of substrate is given by equation (2.42). After substitution in the general mass balance equation (2.10) and differentiation this is obtained

$$\frac{sd(V)}{dt} + \frac{Vd(s)}{dt} = Fs_i - Fs - \left(\frac{\mu(t)}{Y_{XS}} + m_s\right)xV, \quad s(0) = s_\phi \tag{2.60}$$

$\frac{dV}{dt} = 0$ then

$$\frac{d(s)}{dt} = D(s_i - s) - \left(\frac{\mu}{Y_{XS}} + m_s\right)x, s(0) = s_\phi \tag{2.61}$$

Derivation of the mass balance equation for product

The mass rate of the input product is given by equation (2.47). The mass of the product in the fermenter is given by equation (2.49) and the consumption of the product in the fermenter is zero $R_C = 0$. The mass rate of the output product is

$$M_o = Fp \tag{2.62}$$

The generation of the product in the fermenter is

$$R_G = r_p V = q_p x V \tag{2.63}$$

After substitution in the general mass balance equation it is obtained

$$\frac{d(pV)}{dt} = Fp_i - Fp + q_p x V \tag{2.64}$$

After some algebraic manipulation the following is obtained

$$\frac{d(p)}{dt} = D(p_i - p) + (Y_{PX}\mu + m_p)x, \quad p(0) = p_\phi \tag{2.65}$$

The model for continuous fermentation process is

$$\frac{d(x)}{dt} = Dx_i + \left(\frac{\mu_{\max} s}{(K_s + s)} - k_d - D\right)x, \quad x(0) = x_\phi \tag{2.66}$$

$$\frac{d(s)}{dt} = D(s_i - s) - \left(\frac{\mu_{\max} s}{Y_{XS}(K_s + s)} + m_s\right)x, \quad s(0) = s_\phi \tag{2.67}$$

$$\frac{d(p)}{dt} = D(p_i - p) + \left(Y_{PX}\frac{\mu_{\max} s}{(K_s + s)} + m_p\right)x, \quad p(0) = p_\phi \tag{2.68}$$

Normally no concentrations of biomass and product in the input flow, $x_i = 0$ and $p_i = 0$ in the above equations.

2.6.3.2 Steady –state equations description

The steady state equations are obtained after considering steady-state mode of operation in which the derivatives of the biomass, substrate and product variables are zero. Then the equations are

$$\left[\frac{\mu_{\max} s}{(K_s + s)} - k_d - D\right]x = 0 \tag{2.69}$$

$$D(s_i - s) - \left(\frac{\mu_{\max} s}{Y_{XS}(K_s + s)} + m_s \right) x = 0 \tag{2.70}$$

$$Dp + \left(Y_{PX} \frac{\mu_{\max} s}{(K_s + s)} + m_p \right) x = 0 \tag{2.71}$$

2.6.4 Physiochemical variables incorporation into the biological variables equation

In industry

- The processes fed-batch and continuous are controlled by input flow rate (F) and keeping constant T, pH, DO_2 at some previously determined values.
- Batch processes keeping T, pH, DO_2 etc at some previously determined values.

This is far from reality because of disturbances influence and microorganisms requirements. The complex relationship between physiochemical and biological variables and sudden unexplained changes in the process are reflected in the values of the equation kinetic parameters μ_{max}, K_s, Y_{XS} and Y_{PX} which can be considered as functions of the physiochemical variables. Then the physiochemical variables can be considered as control inputs for the process. The optimal value of these variables will determine the optimal value of the biomass and product concentration. The production of yeast and alcohol is dependent in a strong way on the temperature. This is why the thesis developments are based on considerations that the process kinetics coefficients depend on the temperature in the following way:

$$\mu_{\max} = a_1 + a_2 T + a_3 T^2 \tag{2.72}$$

$$K_s = b_1 + b_2 T + b_3 T^2 \tag{2.73}$$

$$Y_{XS} = c_1 + c_2 T + c_3 T^2 \tag{2.74}$$

$$Y_{PX} = d_1 + d_2 T + d_3 T^2 \tag{2.75}$$

Input variables for the above models are the temperature T and the input flow rate F, output variables are the biological variables x, s, p, where p is the product of ethanol giving inhibitory effect on the cell growth. The coefficients values of $a_1 \rightarrow a_3$, $b_1 \rightarrow b_3$, $c_1 \rightarrow c_3$ and $d_1 \rightarrow d_3$ are not known and also they cannot be determined theoretically. As a result of the latter, these coefficients are determined on the basis of experimental data and application of the least mean square (LSM) approach for the parameter estimation.

For inhibition of the cell growth by the ethanol production, the Monod equation can be written as:

$$\mu = (1 - \frac{p}{p^*})^n \frac{\mu_{\max} s}{K_s + s} \qquad (2.76)$$

Once the values of μ_{max}, K_s, Y_{XS} and Y_{PX} have been determined, the model is used to estimate cell biomass, substrate and product concentration as a function of time. The main problem with the fermentation process is that model parameters are difficult to be calculated because they change with time.

2.6.5 Discrete models of the fermentation processes

For the purposes of optimal control calculation and computer application a discrete version of the mass balance equation is derived on the basis of consideration of the first derivative as follows

$$\frac{dz}{dt} = \frac{z(k+1) - z(k)}{\Delta t} \qquad (2.77)$$

Where **z** is representing x, s, p and V and Δt is the sampling period. Discrete models are obtained after substituting equation (2.77) in the mass balance equations.

- **For batch fermentation process**

$$\frac{x(k+1) - x(k)}{\Delta t} = \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - k_d x(k), x(0) = x_\phi \qquad (2.78)$$

$$\frac{s(k+1) - s(k)}{\Delta t} = -\frac{\mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - m_s x(k), s(0) = s_\phi \qquad (2.79)$$

$$\frac{p(k+1) - p(k)}{\Delta t} = \mu_{\max} Y_{PX}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right], p(0) = 0 \qquad (2.80)$$

The above equations can be written in the following way:

$$x(k+1) = x(k) + \Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right] - \Delta t k_d x(k), x(0) = x_\phi \tag{2.81}$$

$$s(k+1) = s(k) - \frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right] - \Delta t m_s x(k), s(0) = s_\phi \tag{2.82}$$

$$p(k+1) = p(k) + \Delta t\mu_{\max} Y_{PX}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right], p(0) = 0 \tag{2.83}$$

- **For fed-batch fermentation process**

$$\frac{x(k+1)-x(k)}{\Delta t} = \mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right] - k_d x(k) - \frac{F(k)x(k)}{V(k)}, x(0) = x_\phi \tag{2.84}$$

$$\frac{s(k+1)-s(k)}{\Delta t} = -\frac{\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right] - mx(k) + \frac{F(k)}{V(k)}[s_i - s(k)], s(0) = s_\phi \tag{2.85}$$

$$\frac{p(k+1)-p(k)}{\Delta t} = \mu_{\max} Y_{PX}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right] - \Delta t\frac{F(k)p(k)}{V(k)}, p(0) = 0 \tag{2.86}$$

$$\frac{V(k+1)-V(k)}{\Delta t} = F(k), V(0) = V_\phi, V(K) = V_{\max} \tag{2.87}$$

The above equations can be written in the following way:

$$x(k+1) = x(k) + \Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right] - \Delta t k_d x(k) - \Delta t\frac{F(k)x(k)}{V(k)}, x(0) = x_\phi \tag{2.88}$$

$$s(k+1) = s(k) - \frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right] - \Delta t m x(k) + \Delta t\frac{F(k)}{V(k)}[s_i - s(k)], s(0) = s_\phi \tag{2.89}$$

$$p(k+1) = p(k) + \Delta t\mu_{\max} Y_{PX}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right] - \Delta t\frac{F(k)p(k)}{V(k)}, p(0) = 0 \tag{2.90}$$

$$V(k+1) = V(k) + \Delta t F(k), V(0) = V_{\phi}, V(K) = V_{\max} \quad (2.91)$$

$$D(k) = \frac{F(k)}{V(k)} \quad (2.92)$$

- **For continuous fermentation process**

$$\frac{x(k+1) - x(k)}{\Delta t} = \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - k_d x(k) - D(k)x(k), x(0) = x_{\phi} \quad (2.93)$$

$$\frac{s(k+1) - s(k)}{\Delta t} = -\frac{\mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - m_s x(k) + D(k)[s_i - s(k)], s(0) = s_{\phi} \quad (2.94)$$

$$\frac{p(k+1) - p(k)}{\Delta t} = \mu_{\max} Y_{PX}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - \Delta t D(k)p(k), p(0) = 0 \quad (2.95)$$

The above equations can be written in the following way:

$$x(k+1) = x(k) + \Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - \Delta t k_d x(k) - \Delta t D(k)x(k), x(0) = x_{\phi} \quad (2.96)$$

$$s(k+1) = s(k) - \frac{\Delta t \mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - \Delta t m x(k) + \Delta t D(k)[s_i - s(k)], s(0) = s_{\phi} \quad (2.97)$$

$$p(k+1) = p(k) + \Delta t \mu_{\max} Y_{PX}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - \Delta t D(k)p(k), p(0) = 0 \quad (2.98)$$

$$D(k) = \frac{F(k)}{V(k)} \quad (2.99)$$

The discrete time models are used for formulation and the solution of the problems for optimal control of the fermentation processes. The models (2.81-2.83), (2.88-2.92) and (2.96-2.99) are used to solve the problem for optimal control of fed-batch, batch and continuos fermentation process.

2.7 Conclusion

Simple models of growth kinetics have evolve little over the past years, and methodical analysis of growth profiles in fermentation systems have not been undertaken. The fact that all fermentation variables cannot be identified is causing the process to be very complicated to model. This can be accredited to the investigational difficulties faced in growth kinetic studies. Further, to be useful in fermentor models, such equations should describe how growth is affected by variations in key environmental parameters such as temperature but current models do this in a very simple fashion and are based on limited experimental data. Experimental work is necessary to describe how growth and death depend on these factors over wide ranges, and how temporal variations in these factors affect the physiological state of the microorganism.

Mathematical models for batch, fed-batch and continuous fermentation processes are developed using the mass balance equations and rate laws. The influence of the environmental for the process temperature over the process of growth and death is proposed to be considered by representation of the kinetic coefficient as function of the temperature. The developed mass balance equations are used further in chapter 4, 5 and 6 for determination of the process optimal control. Overview of the existing approaches for fermentation process control is provided in chapter 3 which describes the importance of computer control, optimization and decomposition of bioprocess optimal control.

CHAPTER THREE
CONTROL, OPTIMIZATION AND DECOMPOSITION OF BIOPROCESS OPTIMAL CONTROL PROBLEM

3.1 Introduction

This chapter describes the importance of fermentation process, precisely looking at principal features, advantages and limitations of a computer control which has been used to solve fermentation problems. Also the optimal and hierarchical control techniques which are particularly effective and efficient in the solution of some fermentation problems and which can be implemented using a computer. A review on the different techniques which have been applied on the optimization of fermentation processes, which are becoming rather popular. A review on the decomposition method that will be used on solving optimal control problem of the processes.

In section 3.2 describes the importance of fermentation process control. Section 3.3 and section 3.4 presents batch, fed-batch and continuous control technology. Section 3.5 describes some important adaptive optimization techniques as they are used as optimization methods needed in order to successfully obtain the optimal operating policies of the processes. Section 3.6 explains decomposition methods for fermentation process optimal control and last in section 3.7 conclusion

3.2 Importance of the fermentation process control

Bioprocesses are becoming more important and with an increasing role in the areas of health care, food, chemicals, agriculture, energy production etc. An increasing number of biological manufacturing routes are promising to compete with traditional chemical engineering production processes; this requires the implementation of modern control concepts (Leigh and Thoma, 1986). Various phases of the development of bioprocesses and their importance has been discussed in details by (Scragg, 1988), (Springham, 1991) and (Humphery, 1990).Successful production of important products from bioprocesses requires the implementation of modern control strategies so that the various stages of the process are efficiently operated at their optimal conditions. The automatic control of bioprocesses is of considerable interest to both research and industry since it may result in many process improvements, including an increase in amount of product produced per unit of biomass, and sustaining consistent levels of increased productivity. Other benefits contain optimizing use of raw materials and improving product quality, which could significantly decrease downstream processing costs (Bishop and Lobert, 1990). Implementation of advanced control strategy may also help in establishing economically viable process. It may compliment the endless efforts of technical experts to reduce the unit costs of production in order to survive in a highly competitive global market. Engineer concerned with implementation of computer based modern control strategies to bioreactors faces a more difficult task than those involved in other chemical processing units (Reuss, 1986). Bioprocesses are highly complex, non-linear, poorly defined and time dependent systems. There are factors that indicate the complexity of bioreactors i.e.

- There are 2000 or more complex set of different biochemical reactions, representing enzymes kinetics inside a micro-organisms.
- The operating conditions in a bioreactor may change due to day-day variations in feed composition.
- Microorganism may adapt to the environment through mutations

There are important characteristics that hinder realistic approach for implementation of advanced control strategies (Wang and Stephanopoulous, 1984), (Semones and Lim, 1987) and (Shi et al., 1989) in bioreactor

- Lack of on-line sensors for measurement of important state variables such as biomass, substrate and product concentration.
- Absence of good mathematical models from engineering point of view which would take into account the numerous factors which influence the growth of microorganism.
- Strong non-linear dynamic behavior with poorly defined or identified time varying parameters

These distinctiveness have had the regrettable result of leaving the biochemical engineering field behind other Engineering fields in implementation of advanced computer control schemes, and in exploiting the on-line optimization techniques (Villadsen, 1989; Williams, 1990). They obstruct the introduction of new bioprocesses for production of mass chemicals. In the present circumstances the fuel ethanol process can be considered a good example which has to contend with gasoline production from crude oil. Its importance was discussed in detail in (Maiorella, 1985; Springham, 1991; Warren, 1992). Ethanol is considered as a desirable fuel since carbon dioxide emissions to the environment are not an additional load to the greenhouse effect. The production of ethanol from biomass generates a whole new agri-industry based on high starch or sugar based crops. These positive features have created both small-scale and large scale ethanol processes in North America and Brazil (Murtagh, 1986; Springham, 1991; Laluce, 1991).

On-line optimal control known that is adaptive optimization requires high-quality on line information about the process operating state, which in turn depends upon logical instrumentation for on-line monitoring, methodical interpretation of the measured signals and estimation of important variables. There are three essential steps for adaptive optimization algorithm.

- On-line monitoring of key state parameters of the system in order to present feedback to the algorithm.
- Estimation of the optimum operating point for the given performance index or trajectory of the optimized variable.
- Adjustment of the manipulated variables to drive the system to the optimum within a reasonable time, when compared to the time constant of the process. In continuous processes all above steps must be executed at every sampling or control interval in order to account for all possible changes in the process conditions which may influence the optimum in either positive or negative ways.

The ability for optimal control of the fermentation processes (maximize or minimize product yield) is of great commercial importance in the biotechnology industry. The use of computers for the control of bioprocesses has increased significantly in the last ten to fifteen years. The advent of microprocessors and the decrease in cost of computers have had a significant effect in the area of real-time computer applications to fermentation processes. One aspect in which the capabilities of the computer can be truly realized is in implementation of adaptive control and optimization strategies (Rolf and Lim, 1985).

In general, the development and implementation of a suitable control strategy for the bioreactor depends on its mode of operation i.e. batch, fed-batch and continuous because each one has different operational problems. Research endeavor has been invested towards optimization of fed-batch and batch fermentation processes, since the majority of biotechnological processes are presently operated in either of these processes (Weigand, 1978) and (Engasser, 1988). Only a few on-line optimization and adaptive control of continuous bioreactors have been reported in literature. On an industrial scale, continuous units are limited due to technological problems associated with contamination, mutations and other complexities associated with design and operation of continuous fermenters. However the future will see a greater use of continuous fermentation processes to produce high quantities of chemicals or fuels from renewable resources. The advantages of continuous process: reduce cost, provide better operational control, improved yields is due to steady operating conditions (Guidoboni, 1984) and (Shama, 1988). The output can be expected to get better with implementation of adaptive optimization strategies which offer possibility of continuously operating at best conditions. Since raw material costs are high, even a small yield can results in considerable savings (Montague et al., 1989).The area of on-line optimization of continuous fermenter is with growing interests in control engineers.

A number of extensive reviews have been published covering different aspects of the area of computer control of fermentation processes (Dorby and Jost, 1977), (Weigand, 1978), (Aiba, 1979), (Rolf and Lim, 1982) (Clarke et al., 1985) (Johson, 1987), (Gerson et al., 1988) and (Montague et al., 1989). Their emphasis was on the application of computer as a process tool in the fermentation industry. They considered implementation of advanced control schemes and on-line optimization technique as the future challenge in the fermentation process control. They also consider the goal of optimal operation as the future direction in computer control of fermentation processes. Control philosophy has been influenced by modern biological genetics. (Sarkar and Modak, 2004) developed an optimization technique based on Genetic Algorithm (GE) to determine optimal substrate feeding policy for fed-batch bioreactors with a multi control variables.

Currently in industry the idea of designing a control system that does not rely on mathematical descriptions of the process is receiving increasing attention owing to remarkable progress in intelligent technology. The techniques used for the improvement of soft sensors, Artificial Neural Network (ANN) has been a accepted approach recently. The advantage of ANN is that it does not need any previous knowledge about the model of the process. Software sensors have been developed using ANN such as in (James et al., 2002). Control techniques have been developed to control fermentation processes in the literature of (Dairaku et al, 1983) and other researches. However, in many cases there are difficulties in constructing an exact mathematical model due to the time-varying nature and non-linear characteristic of fermentation process. It is more appropriate to use fuzzy control to monitor and improve the control operations by synthesizing the knowledge of the experienced operator and advances in computer technology. Fuzzy control has been helpful for bioprocess control during past years (Honda and Kobayashi, 2004). In the work of (Karakuza et al., 2005) they used fuzzy control system for industrial scale barker`s yeast production process using soft-sensors, where the controller determines molasses(as substrate) and air feeding rates to maintain the specific growth rate of the yeast cells and dissolved oxygen concentration at desired set points. The problems for optimal control of the fermentation process are with different formulation and specifics because of the process technology. That is why the review of the existing literature is separately below.

3.3 Batch and fed-batch control

Most of the published work on optimization of batch fermenters is concerned with finding an optimal profile of some important operating variables such as temperature, pH,etc, which can influence the process operation(Weigand, 1978),(Reinikainen et al., 1986 and(Winkler, 1988). In general optimization of fed-batch processes is concerned with the manipulation of the feed rate of a limiting substrate, which requires on-line estimation of critical state variables of the fermenter. An example is fed-batch of baker's yeast using computer (Williams et al., 1984), (Dekkers and Voetter, 1985), (Wu et al., 1985) and (Pomerleau and Perrier, 1990, 1992). Here the sugar feed rate has been manipulated to control the residual sugar concentration in the medium in order to obtain the maximum substrate conversion. The adaptive optimization algorithm for this fed-batch bioreactor determines the sugar feed rate, based on the on-line estimation of substrate and ethanol concentration medium, which in turn depends on state estimation. A similar principle was used in the control of fed-batch penicillin production (Montague et al., 1986), where the timing and rate of nutrient addition was manipulated in order to obtain the maximum penicillin production. Most favorable approach for the fed-batch fermentation of most organisms is to feed the growth limiting substrate at the same rate with demand for the substrate. Four different ways have been used in attempts to balance feed rate with demand:

- Open loop control schemes in which feed is added according to history data
- Indirect monitoring of feed rate based on non feed source parameter such as pH, off gas analysis, DO(dissolve oxygen) or concentration of organic products..
- Indirect control schemes based on mass balanced equations
- Direct control schemes based on direct, on-line measurements of substrates. The figure below describes schematically the concept

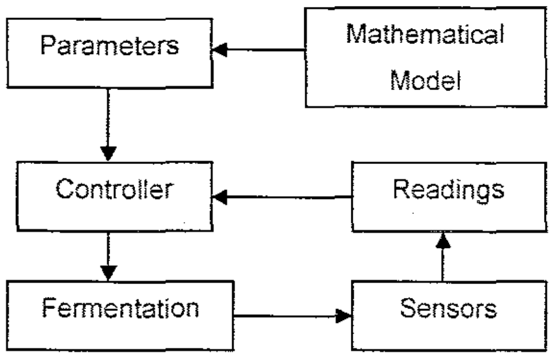


Figure 3.1: Adaptive closed-loop controls

(Ahmad et al., 2008) used optimal control of a nonlinear fed-batch fermentation process based on model prediction approach and on a detailed unstructured model for penicillin production. Moreover avoiding high computational cost, the nonlinear model was substituted with neuro-fuzzy piecewise linear models obtained from method called locally linear model tree. The optimization problem was obtained using Sequential Quadratic Programming (SQP) type method and the algorithm was implemented in Matlab.

3.4 Continuous bioprocess control

Adaptive control and on-line optimization of continuous processes have not been studied due to its limited popularity in the fermentation industry. However some attempts have been made to study optimal control of the following few bioprocesses.

- Baker's yeast in the Continuous Stirred Tank Bioreactor, CSTBR (Rolf and Lim, 1984, 1985a), (Semones and Lim, 1989) and (Chang and Lim1989, 1990).
- Waste-water treatment in an anaerobic bio-film fluidized sand-bed bioreactor(Ryhiner et al., 1992)
- Conversion of fumaric acid to aspartic acid in a packeted-bed immobilized-cell reactor.
- Ethanol production in a CSTR (Karim and Traugh, 1987), (Vigie et al., 1990) and (Queinnec et al., 1991).

3.5 Some important Adaptive Optimization Techniques

3.5.1 Steady-State optimization

This approach can be explained by considering a simple Continuous Stirred Tank Bioreactor, CSTBR, where biomass productivity needs to be maximized using the dilution rate D as the manipulated variable. In this optimization strategy, a step change in dilution rate from D1 to D2 is introduced at each control interval. The corresponding change in steady-state productivity from DX1 to DX2 is evaluated based on offline/ on-line information and the steady-state process gain is estimated as shown below.

$$\left[\frac{d(DX)}{dD}\right]_{steady-state} = \frac{D2X2 - D1X1}{D2 - D1} \tag{3.1}$$

The manipulated variable is then adjusted based on this observed steady-state process gain and repeated until the process gain approaches zero, i.e. No further improvement in productivity can be observed. However in the case of a process with an ever-changing environment this manual procedure has to be repeated. This is the time consuming process since each time one has to wait for the system to attain steady-state before adjusting the manipulated variable. This approach will be useful for any time-invariant process with a low time constant for which no reliable mathematical model is available. However this method is not suitable for bioprocess with high time-constants and with continuous environment disturbances (Rolf and Lim, 1982; Hamer and Richenberg, 1988).

3.5.2 Dynamic Optimization

This optimization allows the computation of the optimal operating point. i.e. the best time varying feed rates which ensure the maximum of a predefined performance index (productivity, or an economical index derived from the operation profile and the final concentrations). It is also called optimal control. However more thoroughly speaking is called open-loop optimal control in order to avoid confusion between it and the closed loop (feedback) optimal control. Most fed-batch fermentation models have high nonlinear dynamics and constraints are also frequently present on both state and the control variables (Banga et al, 2002). Efficient and dynamic optimization techniques are in need in order to successfully obtain the optimal operating point of the system. The optimization problem consists in determining the optimal feeding profile during a fixed time horizon so that a performance index is maximized or minimized subjected to some constraints. In addition, the convergence of the optimization algorithm is dependend on the correctness of the initial guess of the control variable that is iused in the Hamiltonian principle functional.

Gradient-free methods, also called direct search methods, are possible computational methods for these kind of problems and they can possibly attain the global optimum. Numerous direct search methods have been reported, such as iterative dynamic programming (Luus., 1993) and random optimization (Rhinehart, 1990). However, a large amount of the direct search methods are very time consuming and inefficient because they usually require a large number of iterations to search for global optimum.

Gradient based methods, Sequential Quadratic Programming (SQP) are useful for constrained optimization. This method is constructed by solving a quadratic programming sub-problem at each iteration. An estimate is made of the hessian of the Lagrangian function using a quasi-Newton updating technique. This is then used to create Quadratic Programming (QP) sub-problem whose solution is used to form a search direction for a line search procedure. However this kind of computational technique has to deal with problems of numerical valuation of derivatives and feasibility issues. Partial derivatives of the objective function as well as partial derivatives of the constraints are not easy to be obtained in nonlinear systems. The gradients have to be calculated by predetermined difference estimation in the SQP method. This procedure systematically perturbs each variable in order to calculate the partial derivatives of the objective function and the constraints with respect to the control. If the systematic partial derivatives are provided directly, the problem will be solved more precisely and professionally. In (Hough et al., 1971), (Pollok, 1979) and (Tenney, 1989) some research on professional fundamental view of the brewing process have been done whilst on (Sonnleiner and Kappeli, 1986) they consider the main physiological phenomena at cell level. (Gee and Ramirez, 1994) did reasearch considering mathematical studies. (Andres., 1996) decided to employ the same wort and yeast as industry and the same procedures to control only temperature. Temperature has important influence and several series of experiments have been done to obtain a mathematical description of the process. The objective was to acceralate the industrial fermentation, reaching the required ethanol level with less time. Dynamic programming was first chosen as the optimization method because of the algorithm formulation. To apply this method fermentation variables (temperature and time intervals) were discretized. The expectation was that dynamic programming could be carried with sufficient speed but when it was applied, important difficulties of long calculations and big memory demands appeared. Looking for speed, to accomplish first exploration, a distributed computation system was devised, using a local network of six computers and a decomposition method of discritization grid as six horizon bands.

As corroborated by several scientist (Cuthrell and Bieger, 1989), (Chen and Hwang, 1990) and (Luus, 1990) that there is difficulties of dynamic programming application and some alternatives have been proposed. This is the present issue, that is motivating iterative formulation of dynamic programming (Bojkov and Luus, 1994a, 1994b) and (Dadebo and Mcauley, 1995). After the use of dynamic programming, then the use of genetic algorithm was implemented (Goldberg, 1989) and (Davis, 1994). According to the Genetic Algorithm (GA) the numeric description of the temperature profiles (temperature values at the corner points of the sections were taken as chromosomes. (Michalewicz, 1994) developed a GA implementation by means of Matlab using directly the integer values of chromosomes instead of binary based procedure. (Tebbani et al., 2007) solved open-loop optimization problem of fed-batch bioreactor based on a non-linear objective function, namely the production or biomass growth at the final time, under nonlinear constraints on the state trajectory. This optimization problem is a singular problem, since the control variables appear linearly in both the performance index and system dynamics. Determining open-loop optimal feeding law can be carried out by applying Hamiltonian maximum Principle (Bryson and Ho,1975) and transforming the optimization problem into a two point boundary value problem (TPBVP). (Banga, 2005) and(Vassiliadis, 1994) develop another approach that transforms the initial problem into a Nonlinear Optimization Problem (NLP) and solving it using local search methods namely control vector parameterization and the complete parameterization. This is a direct approach of optimization that transforms the original dynamic optimization problem into a nonlinear (NLP) problem. For both these techniques, the state and control variables can be discretized and parameterized and a new NLP finite dimensional problem is derived and solved.

Control Vector Parameterization (CVP) approach, the control variable is approximated by interpolation functions usually polynomials and state variables are computed by integrating system dynamics. Gradients based local methods are the best option for solving NLP's provided that these problems are unimodal and smooth. Sequential Quadratic Linearization (SQL) methods are usually recognized as the state of art in this domain and number of authors has successfully solved bioreactor optimization problems using SQL within CVP network. In Complete Parameterization (CP) approach, the state and control variables are both approximated by predefined functions and the continuity of the trajectory is satisfied with the addition of constraints. The optimization was built around a hybrid method combining these two techniques: state and control variables are both optimized, but with an integration of the trajectory along different time steps. To guarantee a globally optimal solution, direct methods (including CVP) should use Global

Optimization methods (GO). Essentially, GO can be classified as deterministic and stochastic strategies (Torn et al., 1999) and (Guus et al., 1995). (Esposito and Floudas, 2000) have presented a deterministic global optimization approach which can solve nonlinear optimal control problem. (Singer et al., 2001), (Papamichail and Adgiman, 2002) are making good progress in deterministic global optimization of dynamic systems, yet several barriers regarding requirements and computational effort are still present. However, in practice the engineer can be satisfied if these methods provide a very good solution in modest computational time. Evolutionary computation (Bio-inspired methods) that were developed in late 1960 and early 1970 are:

- Genetic Algorithm(GE)(Holland, 1975),(Goldberg, 1989)and(Michalewicz, 1996)
- Evolutionary programming(Fogel et al.,1966) and (Fogel, 1998)
- Evolution strategies(Schwefel, 1995),(Beyer, 1996),(Beyer 2002)

Genetic Algorithms are so far the most popular types of methods but as many authors have reported during recent years they are usually not the most efficient and robust algorithms for GO. This complicates the selection of methods for a given type of problem. For the case of GO of nonlinear dynamic processes different recent experiments indicate that certain simple stochachatstic methods present the best performance. (Simutis and Lubbert, 1997), (Luus and Hennessy, 1999) and (Rodriguez-Acosta et al., 1999) have used other types of stochastic algorithm including different random search algorithms and reached similar conclusions. GE algorithm and related evolutionary approaches that were suggested for solving general optimal control problems by (Michalewiz et al., 1992; Seywald et al., 1995; Yamashita, 1997) have also been used for optimization of feb-batch optimization. (Lee, 1999) have performed comparison on different types of methods but more work still needs to be done in order to come output with some meaningful conclusion. A uniform set of benchmark problems should be used (performance index versus computational time). To asses the performance of diverse solution techniques in a fair way case studies have been used as benchmark problems, as follows

- Fed-batch Reactor for Ethanol production: This case study considered the dynamic optimization of fed-batch involving production of ethanol by Saccharomyces cerevisiae (Chen and Hwang, 1990; Luus, 1993; Banga et al., 2002; Jayaraman, 2001). The free terminal time optimal control problem was to maximize the yield of ethanol using the feed rate as control variable.

- Fed-batch reactor for penicillin production: This problem considered the dynamic optimization of fed-batch reactor for the production of penicillin. (Banga et al., 2002) used CVP scheme and stochastic optimization and (Luus et al., 1993) used Iterative dynamic programming. The optimal control problem was to maximize the total amount of penicillin produced (performance index) using the feed rate of substrate as the control variable.
- Park-Ramirez fed-batch Bioreactor: this problem was dealing with optimal production of secreted protein in the fed-batch reactor and originally formulated by (Park and Ramirez, 1988). The objective was to maximize the secreted protein by a yeast strain in a fed-batch culture. The dynamic model accounted for host-cell growth, gene expression and the secretion of expressed polypeptides.
- Lee-Ramirez Fed-batch bioreactor: this problem was initially offered by (Lee and Ramirez, 1994) and deals with the optimal fed-batch control of induced foreign protein production by recombinant bacteria. The nutrients and inducer feeding rates to the fermentor were control variables. The primary aim was to maximize the profitability of the process.
- The above problem was studied by (Tholudur and Ramirez, 1996) using neural networks parameter function models and by (Carrasco and Banga 1997) using an adaptive stochastic algorithm to obtain better results. The authors indicated that the original performance index exhibited a very low sensitivity with respect to the controls and (Tholudur and Ramirez, 1997) presented a modified parameter function set for this problem in order to increase sensitivity to the controls.

3.6 Decomposition methods for fermentation process optimal control

Engineering problems are usually large, non-convex, non-differentiable and highly coupled. The existing tools in engineering design deal with the highly-coupled nature of decomposed problem by means of estimations. This results in errors, making the coordination cumbersome,this lead to incorrect solutions and a large number of iterations may be required. Lagrangian theory allows decomposition of the original problem into completely decoupled sub-problems. In order to deal with non-convexity, the Lagrangian can be augmented (i.e. augmented Lagragian theory) to convexify the original problem (Fadel et al., 2005). These methods are useful because they converge to the correct solution of the original problem.

Research have been done on the Lagragian duality in the large-scale systems for past years. (Everett, 1963) pioneered the use of this theory for decomposition. In his research products of the Lagrange multipliers and the constraints are added to the objective functional and an unconstrained problem is then optimized. Meaning the new unconstrained problem is completely separated and each sub-system can be solved independently. (Lasdon and Schoeffler, 1965) extended the theory to deal with multi-level structures. They used Lagrange multipliers as variables of dual problem related to the original problem, and in this case are as important as the original problem variables since finding a solution to the dual can lead to primal solution. (Kronsjo, 1969) proposed dual two level two specially structured large-scale convex nonlinear programs. (Schoeffler, 1971) transformed the underlying constrained optimization problem into a two-level problem by fixing certain coordinating variables, and achieved a solution through passing values back and fourth between upper coordinating level and a sub-problems decomposition of the original problem. A decomposition-coordinating algorithm based on augmented Lagragian formulation was used by (Chancelier and Renaud, 1994). (Di Pillo and Lucidi, 2001) emphasize an exact augmented Lagragian, including a penalty term on the whole subset of the first-order necessary conditions corresponding equalities.

3.7 Conclusion

It is clear that although any fermentation process will have its own optimal control problems, there are certain themes in terms of control system consideration and design. A significant conclusion is that, in order to achieve improved control, the processes must become more computerized. Greater computer application leading to improved control and optimization.

These research efforts show that the use of augmented Lagrangian technique may provide satisfactory convergence properties. Based on the above research, augmented Lagragian will be used as the decomposition method to solve the optimal control problem for the fermentation processes. It can be seen that the optimization methods based on the multilevel techniques are applied to solve several fermentation problems using a computer cluster. The knowledge from chapter three is used in chapter four, chapter five and chapter six for development of decomposition methods for solving optimal control problem of batch, fed-batch and continuous process.

CHAPTER FOUR
DEVELOPMENT OF A DECOMPOSITION METHOD FOR OPTIMAL CONTROL CALCULATION FOR BATCH FERMENTATION PROCESS

4.1 Introduction

Control of batch fermentation processes is usually complex due to various reasons (Bastin and Dochain, 1990), (Mikler et al., 1995), (Shen et al., 1995) and (Staniskis, 1993). Complex process of microbial synthesis is proceeding in the fermenter. Several internal and external factors influence this process such as the concentrations of substrate and product, pH, temperature of the medium etc. Variations in the physical properties along the batch process may alter the dynamics of the process control loops. Changes in the quality of the raw material may also be a source of variance. The batch fermentation processes are described by nonlinear equations with time varying parameters (Bastin and Dochain, 1990), (Mikler et al., 1995) and (Staniskis, 1993). The problem for optimal control of the batch process is characterized by a non-quadratic criterion, a nonlinear dynamic model with slowly varying parameters, linear control and by implied by physical limitations constraints placed upon process variables. These problems have been solved by techniques based on (Mikler et al., 1995), (Staniskis, 1993) and (Breusegen and Bastin, 1990).

Section 4.2 describes the formulation of the problem for optimal control of a batch fermentation process. Decomposition method for solving discrete time formulation of the problems based on varational calculus using an augmented Lagrange functional and its decomposition in time domain is developed in section 4.3. The method is developed for a general description of the batch fermentation processes by means of unstructured models with slowly varying parameters. The method is characterized by a new coordinating vector which permits an augmented Lagrange functional to be used for solving nonconvex optimal control problem, and for preserving the full decomposition in the domain of the dual problem. Section 4.4 describes the algorithm of the method that is used to solve the optimal control problem.

4.2 Formulation of the problem for optimal control of a batch fermentation process

The aim of the optimal control of the batch fermentation process is to find trajectory of the temperature: $T(k), k = \overline{0, K-1}$ to produce maximum end concentration of the product at the end of operation period.

$$J = p(K) \to \max \tag{4.1a}$$

for the model equations: (2.81), (2.82), (2.83) and (2.72)-(2.75) written in the form

$$v(k+1) = f_v(k), v(0) = v_\emptyset, k = \overline{0, K-1} \tag{4.1b}$$

where $v = x, s, p,$ and satisfies the constraints

$$\begin{aligned} &x_{\min}(k) \le x(k) \le x_{\max}(k), \ k = \overline{0, K} \\ &s_{\min}(k) \le s(k) \le s_{\max}(k), \ k = \overline{0, K} \\ &p_{\min}(k) \le p(k) \le p_{\max}(k), \ k = \overline{0, K} \\ &T_{\min}(k) \le T(k) \le T_{\max}(k), \ k = \overline{0, K-1} \end{aligned} \tag{4.2}$$

The problem (2.81)-(2.83), (2.72)-(2.75) and (4.1)-(4.2) is solved on the basis of variational calculus using an augmented Lagrange's functional.

4.3 Augmented Lagrange functional formulation for batch fermentation process

The considered problem for optimal control has the following characteristics:

- Criterion only at the final point of the optimization horizon
- Nonlinear model with three state variables i.e. for batch: biomass concentration (s), substrate concentration (s) and product concentration (p) one control variable i.e. temperature (T)
- Long optimization horizon

These characteristics determine which methodology to be selected for solution of the optimal control problem. The theory of variational calculus gives possibilities to handle problem with high dimension and nonlinearities by using the Lagrange functional. The affine control however in the case of non-quadratic criterion determines that the derivative of the Lagrange functional, according to this control does not depend on it. This creates problems with the solution of the two-point boundary value problem. In order to overcome the linear dependability of the Lagragian on the control variable an augmented functional of Lagrange is considered. The augmented Lagrange's functional is assumed here in the form:

$$L = p(K) + \sum_{k=0}^{K-1} \sum_{v=x,s,p} \{\lambda_v(k)[-v(k+1) + f_v(k)] + \frac{1}{2}\mu_v[-v(k+1) + f_v(k)]^2 \} \tag{4.3}$$

where $\lambda_v, v = x, s, p,$ is the vector of the conjugate variables, $\mu_v, v = x, s, p$ is the vector of the penalty coefficients. The augmented functional has to be maximized according to state and control variables and minimized according to conjugate variables on the basis of the theory of duality, under process constraints presented in (4.2)

4.3.1 Necessary conditions for optimality for batch fermentation process

Solution of the problem for optimal control is based on the necessary conditions for optimality of the functional of Lagrange (4.3). The equations of these conditions are used to build the algorithm for calculation of the optimal state and control variables. This algorithm is used for parallel calculation in a cluster of computers.

- Necessary conditions for optimality according to the biomass

$$\frac{\partial L}{\partial x(k)} = \lambda_x(k)\left[1+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s+s(k)}\right]-\Delta t k_d\right]-$$

$$-\mu_x(k)[x(k+1)-f_x(k)]\left[1+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s+s(k)}\right]-\Delta t k_d\right]+$$

$$+\lambda_s(k)\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s+s(k)}\right]-\Delta t m\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{Ks+s(k)}\right]-\Delta t m\right]+$$

$$+\lambda_p(k)\left[\Delta t Y_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s+s(k)}\right]\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta t Y_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s+s(k)}\right]\right]=0=e_x(k), k=\overline{0,K-1}$$

$$\frac{\partial L}{\partial x(K)}=0 \tag{4.4}$$

- Necessary conditions for optimality according to the substrate

$$\frac{\partial L}{\partial s(k)} = \lambda_x(k)\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)(K_s+s(k))-x(k)s(k)}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]+$$

$$+\lambda_s(k)\left[1-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[1-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]$$

$$+\lambda_p(k)\left[\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]=0=e_s(k), k=\overline{0,K-1}$$

$$\frac{\partial L}{\partial s(K)}=0 \qquad (4.5)$$

- Necessary conditions for optimality according to the product

$$\frac{\partial L}{\partial p(k)} = \lambda_x(k)\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]$$

$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]+$$

$$+\lambda_s(k)\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]+$$

$$+\lambda_p(k)\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]+$$

$$=0=e_p(k),\ k=\overline{0,K-1}$$

$$\frac{\partial L}{\partial p(K)}=1 \tag{4.6}$$

- Necessary conditions for optimality according to temperature

$$\frac{\partial L}{\partial T(k)}=\lambda_x(k)\left[\frac{\Delta t\partial\mu_{\max}}{\partial T}\left[1-\frac{p(k)}{p*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{\left[K_s+s(k)\right]^2}\right]\right]-$$

$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\frac{\Delta t\partial\mu_{\max}}{\partial T}\left[1-\frac{p(k)}{p*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{\left[K_s+s(k)\right]^2}\right]\right]+$$

$$+\lambda_s(k)\left[-\Delta t\frac{\frac{\partial\mu_{\max}}{\partial T}Y_{XS}-\mu_{\max}\frac{\partial Y_{XS}}{\partial T}}{Y^2{}_{XS}}\left[1-\frac{p(k)}{p*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$\left.-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{\left[K_s+s(k)\right]^2}\right]\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[-\Delta t\frac{\frac{\partial\mu_{\max}}{\partial T}Y_{XS}-\mu_{\max}\frac{\partial Y_{XS}}{\partial T}}{Y^2{}_{XS}}\left[1-\frac{p(k)}{p*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$\left.-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{\left[K_s+s(k)\right]^2}\right]\right]+\lambda_p(k)\left[\Delta t\left[\frac{\partial Y_{PX}}{\partial T}\mu_{\max}+Y_{PX}\frac{\partial\mu_{\max}}{\partial T}\right]\left[1-\frac{p(k)}{p*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\right.$$

$$+\Delta t Y_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\Bigg]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta t\left[\frac{\partial Y_{PX}}{\partial T}\mu_{\max}+Y_{PX}\frac{\partial \mu_{\max}}{\partial T}\right]\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\right. \tag{4.7}$$

$$\left.+\Delta t Y_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]=0=e_T(k), k=\overline{0,K-1}$$

Where the partial derivatives of the kinetic parameters according to the temperature are:

$$\left.\begin{array}{l}\dfrac{\partial \mu_{\max}}{\partial T}=a_2+2a_3T(k),\ \dfrac{\partial K_s}{\partial T}=b_2+2b_3T(k),\\ \dfrac{\partial Y_{XS}}{\partial T}=c_2+2c_3T(k),\ \dfrac{\partial Y_{PX}}{\partial T}=d_2+2d_3T(k),\end{array}\right\}\quad k=\overline{0,K-1} \tag{4.8}$$

- Necessary conditions for optimality according to the conjugate variables

$$\frac{\partial L}{\partial \lambda_x(k)}=-x(k+1)+f_x(k)=0=e_{\lambda x}(k), k=\overline{0,K-1} \tag{4.9}$$

$$\frac{\partial L}{\partial \lambda_s(k)}=-s(k+1)+f_s(k)=0=e_{\lambda s}(k), k=\overline{0,K-1} \tag{4.10}$$

$$\frac{\partial L}{\partial \lambda_p(k)}=-p(k+1)+f_p(k)=0=e_{\lambda p}(k), k=\overline{0,K-1} \tag{4.11}$$

4.3.2 Coordinating vector for the batch fermentation process

The optimal trajectories of the variables can be calculated as a solution of the obtained set of nonlinear equations (4.4)-(4.11). It can be seen that the number of equations and variables is big. The solution could be found easier if the problem is decomposed. Decomposition in time domain could reduce complexity of the system equations (Tamura, 1975) and (Singandlitty, 1982).

The solutions in time domain will consists of K+1 separate solutions, everyone at separate time moment k. The decomposition in time domain could be obtained on the basis of the coordinating procedure in two level computing structure(given in Figure 4.1) using the conjugate variables λ as coordinating ones(Tamura, 1975), but this method is applicable for normal functional of Lagrange. The used augmented functional could not be fully decomposed because of the quadratic terms and variables $x(k+1), s(k+1)$ and $p(k+1)$ in them. They play a role of interconnections in time domain (Lin, 1992). This type of functional can be fully decomposed if the considered above variables are selected also as a part of the coordinating vector. To overcome the difficulties with full decomposition of the augmented functional of Lagrange, it is proposed to consider

$$\delta_v(k) = v(k+1), v = x, s, p, \quad k = \overline{0, K-1} \tag{4.12}$$

as coordinating variables. Then the values of the coordinating variables are set from the second level of the two level calculating structure:

$$\lambda_v(k) = \lambda_v{}^j(k), v = x, s, p, \; k = \overline{0, K} \tag{4.13}$$

$$\delta_v(k) = \delta_v{}^j(k), v = x, s, p, \; k = \overline{0, K} \tag{4.14}$$

where j is the index of the coordinating process iterations. When the values of the coordinating variables from equations (4.13) – (4.14) are substituted into the Lagrange's functional, its full decomposition according to the discrete time moments k is obtained. The functional is decomposed into K+1 sub-functionals $L_a(k)$ and each of them determines the optimal control and state at the given moment k. The above algorithm will be implemented in Matlab software environment in two ways:

- Sequentially in one computer
- Parallel, using the cluster of computers

The decomposition in time domain is based on a two level calculation structure as shown in Figure 4.1

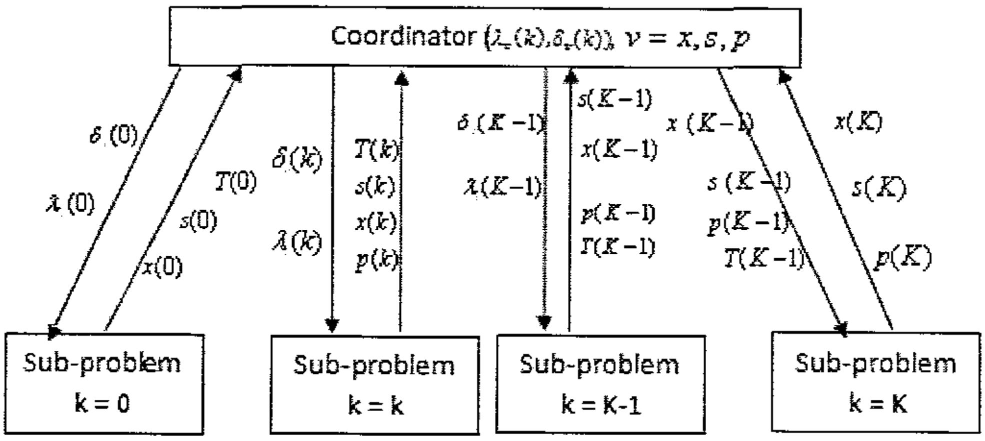


Fig 4.1 Two level calculation structure for batch fermentation process

4.3.3 First level sub-problems for batch fermentation process

The solutions of the first level sub-problems represent solution of the set of equations (4.4) – (4.8) in which the coordinating variables are substituted. These equations are solved for every separate moment of time in a parallel way. The structure of the equations (4.4) – (4.8) does not allow to use analytical solution that is why numerical solution is used. Gradient method is used in the form:

$$v^{j,i+1}(k) = v^{j,i}(k) + \alpha_v e_v^{\ j,i}(k), v = x,s,p,T, \quad k = \overline{0,K-1} \qquad (4.15)$$

where $\alpha_v > 0$ are the steps of the gradient procedures, $e_v(k)$ are the gradients according to the equations (4.4) – (4.8) and *i* is the iteration index of the procedures. The first level sub-problems are determined under the set from the second level coordinating variables according to the necessary conditions for optimality of the sub-functionals $L_a(k)$ and also (4.4)-(4.8) can be simplified further by using necessary conditions for optimality according to the conjugate variables presented by using (4.9)–(4.11) in the form:

For the biomass

$$\frac{\partial L}{\partial x(k)} = e_x^{j,i}(k) = \left[1 + \Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s + s(k)} - \Delta tk_d\right]\left[\lambda_x{}^j(k) - \mu_x e_{\lambda x}{}^j(k)\right] +$$
$$+\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s + s(k)} - \Delta tm\right]\left[\lambda_s{}^j(k) - \mu_s e_{\lambda s}{}^j(k)\right] + \tag{4.16}$$
$$+\left[\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s + s(k)}\right]\left[\lambda_p{}^j(k) - \mu_p e_{\lambda p}{}^j(k)\right], k = \overline{0, K-1}$$

For the substrate

$$\frac{\partial L}{\partial s(k)} = e_s^{j,i}(k) = \left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)K_s}{[K_s + s(k)]^2}\right]\left[\lambda_x(k) - \mu_x e_{\lambda x}(k)\right] +$$
$$+\left[1-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)K_s}{[K_s + s(k)]^2}\right]\left[\lambda_s{}^j(k) - \mu_s e_{\lambda s}{}^j(k)\right] + \tag{4.17}$$
$$+\left[\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)K_s}{[K_s + s(k)]^2}\right]\left[\lambda_p{}^j(k) - \mu_p e_{\lambda p}{}^j(k)\right], k = \overline{0, K-1}$$

For the product

$$\frac{\partial L}{\partial p(k)} = e_p^{j,i}(k) = \left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[\frac{-1}{p^*}\right]\frac{x(k)s(k)}{K_s + s(k)}\right]\left[\lambda_x{}^j(k) - \mu_x e_{\lambda x}{}^j(k)\right]$$
$$\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[\frac{-1}{p^*}\right]\frac{x(k)s(k)}{K_s + s(k)}\right]\left[\lambda_s{}^j(k) - \mu_s e_{\lambda s}{}^j(k)\right] +$$
$$+\left[1 + \Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[\frac{-1}{p^*}\right]\frac{x(k)s(k)}{K_s + s(k)}\right]\left[\lambda_p{}^j(k) - \mu_p e_{\lambda p}{}^j(k)\right], k = \overline{0, K-1}$$

(4.18)

For the temperature

$$\frac{\partial L}{\partial T(k)} = e_T(k) =$$

$$= \left[\frac{\Delta t \partial \mu_{\max}}{\partial T} \left[1 - \frac{p(k)}{p*} \right]^n \frac{x(k)s(k)}{K_s + s(k)} + \Delta t \mu_{\max} \left[1 - \frac{p(k)}{p*} \right]^n \left[\frac{x(k)s(k) \frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2} \right] \right] *$$

$$* \left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k) \right] +$$

$$+ \left[-\Delta t \frac{\frac{\partial \mu_{\max}}{\partial T} Y_{XS} - \mu_{\max} \frac{\partial Y_{XS}}{\partial T}}{Y^2_{XS}} \left[1 - \frac{p(k)}{p*} \right]^n \frac{x(k)s(k)}{K_s + s(k)} + \right.$$

$$\left. + \frac{\Delta t \mu_{\max}}{Y_{XS}} \left[1 - \frac{p(k)}{p*} \right]^n \left[\frac{x(k)s(k) \frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2} \right] \right] \left[\lambda_s(k) - \mu_s(k) e_{\lambda s}(k) \right] +$$

$$+ \left[\Delta t \frac{\partial \mu_{\max}}{\partial T} Y_{PX} + \mu_{\max} \frac{\partial Y_{PX}}{\partial T} \left[1 - \frac{p(k)}{p*} \right]^n \frac{x(k)s(k)}{K_s + s(k)} - \right.$$

$$\left. - \Delta t \mu_{\max} Y_{PX} \left[1 - \frac{p(k)}{p*} \right]^n \left[\frac{x(k)s(k) \frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2} \right] \right] \left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k) \right], k = \overline{0, K-1} \tag{4.19}$$

These equations can be simplified further for the purpose of calculations. Some expressions in them are repeated, that is why common notations are used to represent them:

$$\Delta t \mu_{\max} \left[1 - \frac{p(k)}{p*} \right]^n \frac{s(k)}{K_s + s(k)} = \varphi_x(k) \tag{4.20}$$

$$\frac{\Delta t \mu_{\max}}{p^*} \left[1 - \frac{p(k)}{p*} \right]^{n-1} n \frac{x(k)s(k)}{K_s + s(k)} = \varphi_p(k) \tag{4.21}$$

$$\Delta t \mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)K_s}{[K_s+s(k)]^2} = \varphi_s(k) \tag{4.22}$$

$$\Delta t\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s+s(k)} = \varphi_1(k) \tag{4.23}$$

$$\Delta t \mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{[K_s+s(k)]^2} = \varphi_2(k), k=\overline{0,K-1} \tag{4.24}$$

Using notations from (4.20) - (4.24) equations from (4.16)-(4.19) can be written in the form:

For the biomass

$$\begin{aligned}\frac{\partial L}{\partial x(k)} &= e_x^{j,i}(k) = \left[1+\varphi_x^{j,i}(k) - \Delta t k_d\right]\left[\lambda_x^j(k) - \mu_x(k) e_{\lambda x}^j(k)\right] + \\ &+ \left[-\frac{\varphi_x^{j,i}(k)}{Y_{XS}} - \Delta t m\right]\left[\lambda_s^j(k) - \mu_s(k) e_{\lambda s}^j(k)\right] + \\ &+ \varphi_x^{j,i}(k) Y_{PX}\left[\lambda_p^j(k) - \mu_p(k) e_{\lambda p}^j(k)\right], k=\overline{0,K-1}\end{aligned} \tag{4.25}$$

For the substrate

$$\begin{aligned}\frac{\partial L}{\partial s(k)} &= e_s^{j,i}(k) = \varphi_s^{j,i}(k)\left[\lambda_x^j(k) - \mu_x(k) e_{\lambda x}^j(k)\right] + \\ &+ \left[1-\frac{\varphi_s^{j,i}(k)}{Y_{x/s}}\right]\left[\lambda_s^j(k) - \mu_s(k) e_{\lambda s}^j(k)\right] + \\ &+ Y_{PX}\varphi_s(k)\left[\lambda_p^j(k) - \mu_p(k) e_{\lambda p}^j(k)\right], k=\overline{0,K-1}\end{aligned} \tag{4.26}$$

For the product

$$\begin{aligned}\frac{\partial L}{\partial p(k)} &= e_p^{j,i}(k) = \left[-\varphi_p(k)\right]\left[\lambda_x^j(k) - \mu_x e_{\lambda x}^j(k)\right] + \\ &+ \left[\frac{\varphi_p(k)}{Y_{XS}}\right]\left[\lambda_s^j(k) - \mu_s e_{\lambda s}^j(k)\right] + \\ &+ \left[1-\varphi_p(k)\right]\left[\lambda_p^j(k) - \mu_p(k) e_{\lambda p}^j(k)\right], k=\overline{0,K-1}\end{aligned} \tag{4.27}$$

For the temperature

$$\frac{\partial L}{\partial T(k)} = e_T{}^{j,i}(k) =$$

$$= \left[\varphi_1(k)\frac{\partial \mu_{\max}}{\partial T} + \varphi_2(k)\frac{\partial K_s}{\partial T}\right]\left[\lambda_x{}^j(k) - \mu_x(k)e_{\lambda x}{}^j(k)\right] + \left[\left[-\varphi_1(k)\right]\left[\frac{\frac{\partial \mu_{\max}}{\partial T}Y_{XS} - \mu_{\max}\frac{\partial Y_{XS}}{\partial T}}{Y^2{}_{XS}}\right]\right. +$$

$$\left. + \frac{\varphi_2(k)}{Y_{XS}}\frac{\partial K_s}{\partial T}\right]\left[\lambda_s{}^j(k) - \mu_s(k)e_{\lambda s}{}^j(k)\right] + \left[\varphi_1(k)\left[\frac{\partial Y_{PX}}{\partial T}\mu_{\max} + Y_{PX}\frac{\partial \mu_{\max}}{\partial T}\right] - \right. \tag{4.28}$$

$$\left. - Y_{PX}\varphi_2(k)\frac{\partial K_s}{\partial T}\right]\left[\lambda_p{}^j(k) - \mu_p(k)e_{\lambda p}{}^j(k)\right], k = \overline{0, K-1}$$

The calculation of the gradients is done with the old values of the state and control variables $\nu^{j,i}$, $\nu = x, s, p, T$. The calculation with the gradient procedure will continue until the necessary conditions for optimality towards state and control variables are fulfilled. This means that the values of the gradients are approximately equal to zero:

$$\left\| e_\nu(k) \right\| \leq \varepsilon_\nu, \varepsilon_\nu > 0, \nu = x, s, p, T, \quad k = \overline{0, K-1} \tag{4.29}$$

The new calculated values have to be checked for belonging to the area of constraints

$$x_{\min} \leq x(k) \leq x_{\max}, \quad k = \overline{0, K} \tag{4.30}$$

$$s_{\min} \leq s(k) \leq s_{\max}, \quad k = \overline{0, K} \tag{4.31}$$

$$p_{\min} \leq p(k) \leq p_{\max}, \quad k = \overline{0, K} \tag{4.32}$$

$$T_{\min} \leq T(k) \leq T_{\max} \quad , k = \overline{0, K-1} \tag{4.33}$$

Simple way to do and correct the obtained new values is to project them over the constraints domain according to the relation:

$$\nu^{j,i}(k) = \left\{ \begin{array}{l} \nu_{\min}, \; if \;\; \nu^{j,i}(k) < \nu_{\min} \\ \nu^{j,i}(k), \; if \;\; \nu_{\min} \leq \nu^{j,i}(k) \leq \nu_{\max} \\ \nu_{\max} \;\; if \;\; \nu^{j,i}(k) > \nu_{\max} \\ where \, \nu = x, s, p, \;\; k = \overline{0, K}, \;\; \nu = T, \;\; k = \overline{0, K-1} \end{array} \right\} \tag{4.34}$$

The obtained values of the trajectories of the state and control variables are function of the coordinating variables. They are optimal for the given values of the coordinating variables. If the coordinating trajectories are optimal ones, then the obtained state and control variables also will be the optimal according to the theory of the duality in optimal control and Lagrange methods.

4.3.4 The coordinating sub-problem for batch fermentation process

The coordinating variables are obtained from the necessary conditions for optimality of the augmented Lagrange functional according to them. For the conjugate variables these are equations (4.9)-(4.11). The necessary conditions for optimality according to the new introduced interconnections in time domain are:

$$\frac{\partial L}{\partial \delta_x(k)} = -\lambda_x(k) - \mu_x[-\delta_x(k) + f_x(k)] = -\lambda_x(k) - \mu_x e_{\lambda x}(k) = e_{\delta x}(k) = 0 \qquad (4.35)$$

$$\frac{\partial L}{\partial \delta_s(k)} = -\lambda_s(k) - \mu_s[-\delta_s(k) + f_s(k)] = -\lambda_s(k) - \mu_s e_{\lambda s}(k) = e_{\delta s}(k) = 0, \qquad (4.36)$$

$$\frac{\partial L}{\partial \delta_p(k)} = -\lambda_p(k) - \mu_p[-\delta_p(k) + f_p(k)] = -\lambda_p(k) - \mu_p e_{\lambda p}(k) = e_{\delta p}(k) = 0, \qquad (4.37)$$

where

$$-\delta_v(k) + f_v(k) = e_{\lambda v}(k), v = x, s, p,$$

according to equations (4.9)-(4.11). The necessary conditions for optimality (4.9)-(4.11) and (4.35)-(4.37) can be combined in a gradient procedure. Equations (4.35)-(4.37) give the values of the gradients for the conjugate variables multiplied by the penalty co-efficient. μ_v, $v = x, s, p$

It appears that:

$$\frac{\partial L}{\partial \delta_v(k)} = -\lambda_v(k) - \mu_v \frac{\partial L}{\partial_{\lambda v}(k)}, k = \overline{0, K-1} \qquad (4.38)$$

Gradient procedure can be built for both type of coordinating variables, as follows:

For conjugate variables:

$$\lambda_v^{j+1}(k) = \lambda_v^{j}(k) - \alpha_{\lambda v} e^{j}{}_{\lambda v}(k), \qquad (4.39)$$

where

$$e^j{}_{\lambda_v}(k) = -\delta_v{}^j(k) + f^j{}_v(k), v = x, s, p, \quad k = \overline{0, K-1}, \tag{4.40}$$

$$\delta_v{}^{j+1}(k) = \delta_v{}^j(k) + \alpha_{\delta_v} e_{\delta_v}(k), \; k = \overline{0, K-1} \tag{4.41}$$

where α_{λ_v} and α_{δ_v} are the steps of the procedures for calculation of the conjugate variables and the interconnection in time, respectively. As the calculations are based on gradient iterative procedure, they will stop when the gradient is going to zero, or practically achieving some conditions for optimality, given by

$$\left\|e^{j+1}{}_{\lambda v}(k) - e^j{}_{\lambda v}(k)\right\| \le \varepsilon_{\lambda v}, \varepsilon_{\lambda v} > 0, \qquad k = \overline{0, K}$$

$$\left\|e^{j+1}{}_{\delta v}(k) - e^j{}_{\delta v}(k)\right\| \le \varepsilon_{\delta v}, \varepsilon_{\delta v} > 0, \; k = \overline{0, K-1} \tag{4.42}$$

where $\varepsilon_{\delta v}$, ε_{λ_v} are a very small positive numbers. For every step of the gradient procedure for the coordinating process the full iteration cycle of the sub problems of the first level is done. The calculations of the coordinating variables are done with the values of state and control variables of the first level sub problems and the calculations of the first level sub problems are done with the values of the calculated on the previous iteration coordinating variables. The iterative process of coordination and the optimal solution of the whole problem for optimal control is obtained when the optimal solution of the coordinating sub problem is reached, this means conditions (4.42) are fulfilled and if these conditions are not reached the error is computed.

$$e^{j+1} = \left\{\left\|e^{j+1}{}_{\lambda v}\right\|^2 + \left\|e^{j+1}{}_{\delta v}\right\|^2\right\}^{1/2} \tag{4.43}$$

The new penalty coefficient is calculated from the conditions.

$$if\; j = 1\; or\; e^{j+1} < e^j\,, then\; \mu^{j+1} = \mu^j \tag{4.44}$$

$$if\; j > 1 and\; e^{j+1} \ge e^j \text{then}\; \mu^{j+1} = \alpha\mu^j, \alpha = \;[0.1 \;\; 10.0] \tag{4.45}$$

4.4 Algorithm of the Augmented Lagrange for batch fermentation process

The calculations in the two-level structure can be grouped and represented in the following algorithm:

1. On the coordinating level the initial values of the coordinating and gradient procedures, variables and parameters, initial trajectories of the state, control and coordinating variables are set as follows:
   - Maximal number of step of the coordinating process – M2.
   - Maximal number of steps for state and control variables gradient procedure- M1.
   - Steps of the gradient procedures
     - For coordinating conjugate variables -$\alpha_{\lambda_\nu}, \nu = x, s, p$.
     - For coordinating interconnection in time $\alpha_{\delta_\nu}, \nu = x, s, p$
     - For state variables and control variable $\alpha_\nu, \nu = x, s, p, T$
   - Values for error and termination of the calculations for coordinating procedures. $\varepsilon_{\lambda_\nu}, \varepsilon_{\delta_\nu}, \nu = x, s, p$ and for gradient procedures on first level. $\varepsilon_\nu, \nu = x, s, p, T$
   - Initial trajectories of the conjugate variables. $\lambda^1_\nu(k), \nu = x, s, p, k = \overline{0, K-1}$
   - Initial values of the state variables, $x(0)$, $s(0)$, $p(0)$.
   - Number of steps in the optimization horizon – K and value of the sampling period Δt.
   - Initial trajectories of the control variables $T^{1,1}(k), k = \overline{0, K-1}$
   - The initial trajectories of the control variables are substituted in the model equations (2.81)-(2.83) and (2.72)-(2.75) and the state trajectories are calculated.
   - The initial trajectories of the interconnections in time domain are calculated from $\delta_\nu(k) = \nu(k+1), k = \overline{0, K-1}$
   - Initial trajectories of the coordinating variables and initial state and control trajectories are sent to the first level.
   - Coefficients of the model $a_1 \to a_3$, $b_1 \to b_3$, $c_1 \to c_3$, and $d_1 \to d_3$, k_d, m, n and p^*

2. On the first level the following calculations are done:
   - Derivatives of equations (4.8) are calculated.
   - The kinetic coefficients $\mu_{\max}, K_s, Y_{XS}, Y_{PX}$ from equations (2.72)-(2.75) are calculated.
   - The expressions $\varphi_x(k), \varphi_s(k), \varphi_p(k), \varphi_1(k), \varphi_2(k), k = \overline{0, K-1}$ are calculated from equations (4.20)-(4.24).
   - The gradients $e_\nu^{i,j}(k), k = \overline{0, K-1}, \nu = x, s, p, T$ are calculated from equations (4.25)-(4.28).
   - The errors for the end of the gradient procedures are calculated according to equation (4.29). If the error is less than ε_ν, the calculations of the first level stop and the current values of the state and control variables are the optimal ones. If the error conditions are not satisfied the new values of the state and control values are calculated.
   - The new values of state and control variables are calculated from equation (4.15).

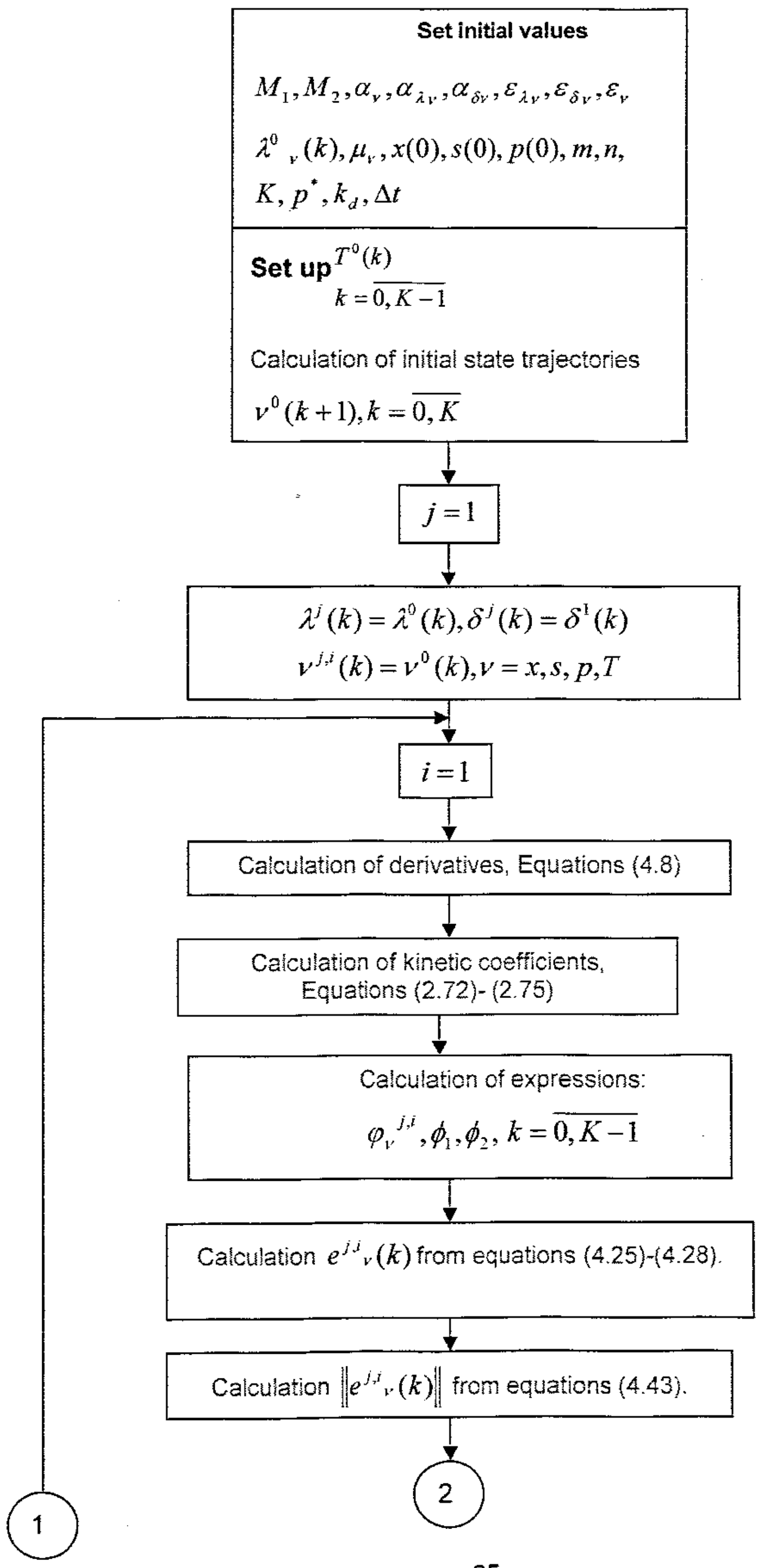
- The obtained state and control variables are projected over the constraint domain according to equation (4.34). The obtained trajectories are sent to the second level.

3. On the second level the following calculations are done:

- The gradient $e_{\lambda_\nu}(k), k=\overline{0,K-1}, \nu = x,s,p$ are calculated from equations (4.9)-(4.11).
- The conditions for the end of the calculations are checked according to Equation (4.42). If the conditions are fulfilled the optimal solution of the coordinating and of the whole problem are obtained. If the conditions are not satisfied the new values of the coordinating variables are calculated.
- New values of the conjugate variables are calculated according to equation (4.39)-(4.40).
- The new values of the interconnections are calculated from equation (4.41). These trajectories are sent to the first level sub problems where the calculations from p.2a till 3b are repeated and so on. The schematic diagram of the algorithm is given on Fig 4.2.

4.6 Description of the Matlab sequential program for optimal control of batch fermentation process

The complex relationship between physiochemical and biological variables and sudden unexplained changes in the process are reflected in the values of the equation kinetic parameters equations (2.72)-(2.75) which can be considered as functions of the physiochemical variables. Then the physiochemical variables can be considered as control inputs for the process. The optimal value of these variables will determine the optimal value of the biomass and product concentration. The production of yeast and alcohol is dependent in a strong way on the temperature. This is why the thesis developments are based on considerations that the process kinetics coefficients depend on the temperature. Table 4.2 defines process kinetic values that are calculated from Matlab by using *polyfit function* from a certain given values of the equation kinetic parameters equations


Set initial values
$M_1, M_2, \alpha_\nu, \alpha_{\lambda\nu}, \alpha_{\delta\nu}, \varepsilon_{\lambda\nu}, \varepsilon_{\delta\nu}, \varepsilon_\nu$
$\lambda^0{}_\nu(k), \mu_\nu, x(0), s(0), p(0), m, n,$
$K, p^*, k_d, \Delta t$
Set up $T^0(k)$ $k=\overline{0,K-1}$
Calculation of initial state trajectories
$\nu^0(k+1), k=\overline{0,K}$
$j=1$
$\lambda^j(k)=\lambda^0(k), \delta^j(k)=\delta^1(k)$
$\nu^{j,i}(k)=\nu^0(k), \nu=x,s,p,T$
$i=1$
Calculation of derivatives, Equations (4.8)
Calculation of kinetic coefficients, Equations (2.72)- (2.75)
Calculation of expressions:
$\varphi_\nu{}^{j,i}, \phi_1, \phi_2, k=\overline{0,K-1}$
Calculation $e^{j,i}{}_\nu(k)$ from equations (4.25)-(4.28).
Calculation $\left\|e^{j,i}{}_\nu(k)\right\|$ from equations (4.43).
2
1

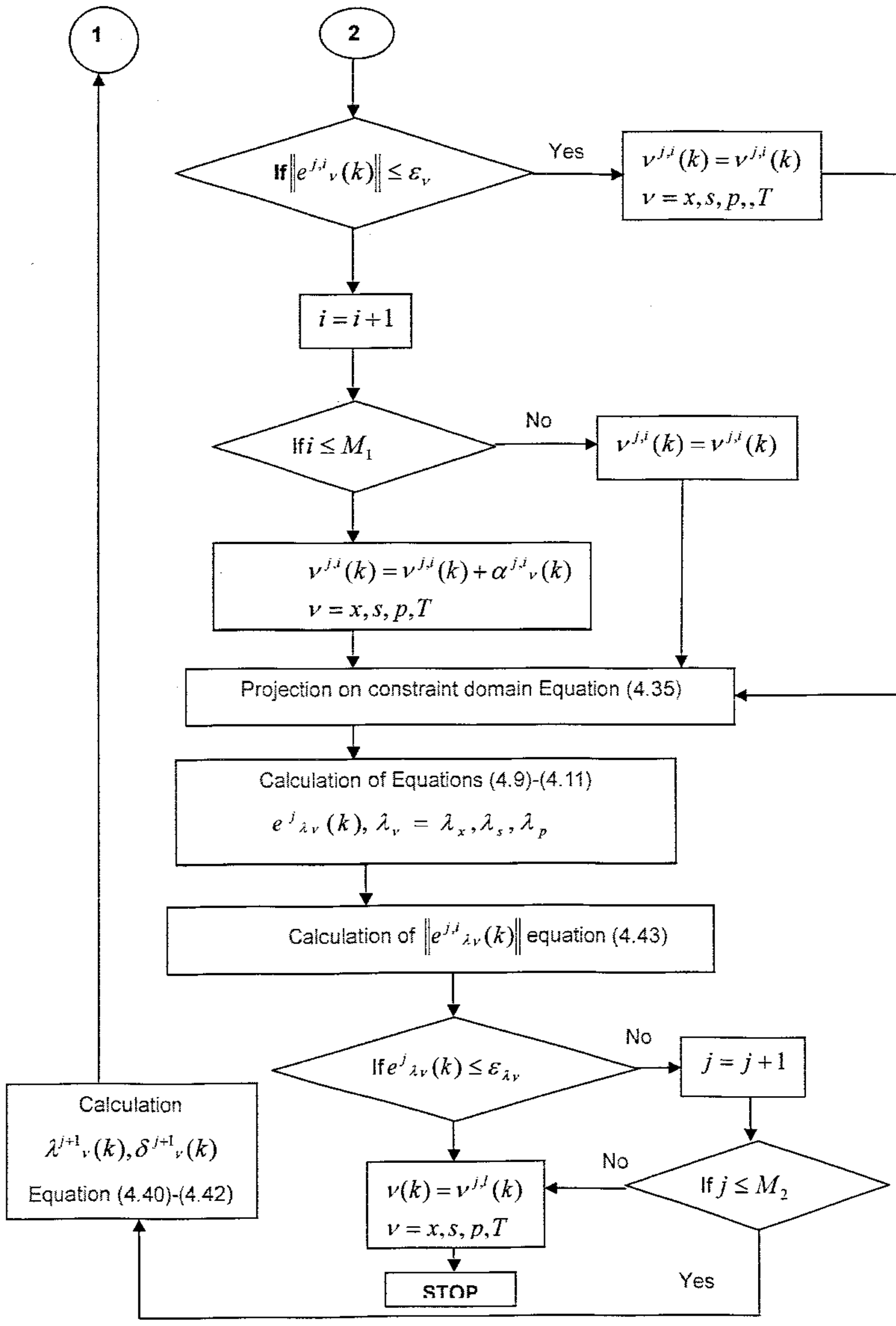

1
2
If $\|e^{j,i}{}_{v}(k)\| \le \varepsilon_v$
Yes
$v^{j,i}(k) = v^{j,i}(k)$
$v = x, s, p,, T$
$i = i + 1$
If $i \le M_1$
No
$v^{j,i}(k) = v^{j,i}(k)$
$v^{j,i}(k) = v^{j,i}(k) + \alpha^{j,i}{}_{v}(k)$
$v = x, s, p, T$
Projection on constraint domain Equation (4.35)
Calculation of Equations (4.9)-(4.11)
$e^{j}{}_{\lambda v}(k),\ \lambda_v = \lambda_x, \lambda_s, \lambda_p$
Calculation of $\|e^{j,i}{}_{\lambda v}(k)\|$ equation (4.43)
If $e^{j}{}_{\lambda v}(k) \le \varepsilon_{\lambda v}$
No
$j = j + 1$
Calculation
$\lambda^{j+1}{}_{v}(k), \delta^{j+1}{}_{v}(k)$
Equation (4.40)-(4.42)
No
If $j \le M_2$
$v(k) = v^{j,i}(k)$
$v = x, s, p, T$
STOP
Yes

Figure 4.2: Flow chart of the algorithm for calculation of batch fermentation process state and control trajectories.

4.5 Description and calculation results from the sequential program of the algorithm of the method in Matlab for batch fermentation process

The algorithm presented in section 4.4 is implemented in Matlab in a sequential way(Appendix A). The table below represents the relationship between mathematical notation and program notation of the Matlab program.

Table 4.1: Correspondence between mathematical and programs notations for batch fermentation process

Name	Notation in the text	Notation in Matlab
State variable Biomass Substrate Product		
	x	x
	s	s
	p	p
Control variable Temperature	T	T
Steps of the gradient procedures For state variables	α_x	ax
	α_s	as
	α_p	ap
	$\alpha_{\lambda x}$	alx
	$\alpha_{\lambda s}$	als
	$\alpha_{\lambda p}$	alp
	$\alpha_{\delta x}$	adx
	$\alpha_{\delta p}$	ads
	$\alpha_{\delta p}$	adp
Conjugate variable	λ_x	lx
	λ_s	ls
	λ_p	lp
Interconnections in time domain	δ_x	dx
	δ_s	ds
	δ_p	dp
Coefficients	a_1, a_2, a_3	a1, a2, a3

	b_1, b_2, b_3	b1, b2, b3
	c_1, c_2, c_3	c1, c2, c3
	d_1, d_2, d_3	d1, d2, d3
	K_d	Kd
	K_s	Ks
	m	m
	n	n
	Y_{xs}	Yxs
	Y_{px}	Ypx
	s_i	si
	p^*	pl
	Δt	dt
	$\mu_{\max}$	mmax
Initial values of state variables	x(0)	x0
	s(0)	s0
	p(0)	p0
Penalty coefficients	μ_x	mx
	μ_s	ms
	μ_p	mp
Min and max of state variables	x_{max}	xmax
	x_{min}	xmin
	s_{max}	smax
	s_{min}	smin
	p_{max}	pmax
Min and max of control variables	T_{max}	Tmax
	T_{min}	Tmin
Derivatives of the model parameters	$\frac{\partial \mu_{\max}}{\partial T} = a_2 + 2a_3T(k)$	dmmaxdT(k)=a2+2*a3*T(k);
	$\frac{\partial K_s}{\partial T} = b_2 + 2b_3T(k)$	dKsdT(k)=b2+2*b3*T(k);
	$\frac{\partial Y_{xs}}{\partial T} = c_2 + 2c_3T(k)$	dYxsdT(k)=c2+2*c3*T(k);

	$\frac{\partial Y_{px}}{\partial T} = d_2 + 2d_3T(k)$	dYpxdT(k)=d2+2*d3*T(k);
Error	$e_{\lambda x}(k) = \partial x(k) - x(k+1)$	elx(k) = dx(k)-x(k+1)
	$e_{\lambda s}(k) = \partial s(k) - s(k+1)$	els(k) = ds(k)-s(k+1)
	$e_{\lambda p}(k) = \partial p(k) - p(k+1)$	elp(k) = dp(k)-p(k+1)
	$e_{\delta x}(k) = \mu_x(k) e_{\lambda x}{}^{j}(k)$	edx(k)=mx*elx(k)
	$e_{\delta s}(k) = \mu_s(k) e_{\lambda s}{}^{j}(k)$	eds(k)=ms*els(k)
	$e_{\delta p}(k) = \mu_p(k) e_{\lambda p}{}^{j}(k)$	edp(k)=mp*elp(k)
	$e_{ex}(k) = \lambda_x{}^{j}(k) - \mu_x(k) e_{\lambda x}{}^{j}(k)$	eex(k)=lx(k)-mx*elx(k)
	$e_{es}(k) = \lambda_s{}^{j}(k) - \mu_s(k) e_{\lambda s}{}^{j}(k)$	ees(k)=ls(k)-ms*els(k)
	$e_{ep}(k) = \lambda_p{}^{j}(k) - \mu_p(k) e_{\lambda p}{}^{j}(k)$	eep(k)=lp(k)-mp*elp(k)
Formulas	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n = \varphi_f(k)$	ff(k)=dt*mmax(k) *((1-pk/pl)^0.52)
	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s + s(k)} = \varphi_x(k)$	fx(k)=ff(k)*s(k)/(Ks(k) + s(k))
	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{[K_s + s(k)]^2} = \varphi_s(k)$	fs(k)=ff(k)*Ks(k) * x(k)/((Ks(k) + s(k))^2)
	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^{n-1} \frac{x(k)s(k)}{K_s + s(k)} = \varphi_p(k)$	fp(k)=dt*mmax(k)*((1-p(k)/pl)^(n-1))*x(k)*s(k)/(pl*(Ks(k) + s(k)))
	$\Delta t\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} = \varphi_1(k)$	f1(k)=ff(k)*x(k)*s(k)/(Ks(k) + s(k))
	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{[K_s + s(k)]^2} = \varphi_2(k)$	f2(k)=f1(k)/((Ks(k) + s(k)))^2

Table 4.2: Model parameter values for batch fermentation process

Symbol	Name	Value	Unit
a1	Model co-efficient	0.0004	
a2	Model co-efficient	0.0001	
a3	Model co-efficient	0.0002	
b1	Model co-efficient	0.2	
b2	Model co-efficient	0.03	
b3	Model co-efficient	0.01	
c1	Model co-efficient	2	
c2	Model co-efficient	0.3	
c3	Model co-efficient	0.021	
d1	Model co-efficient	0.0040	
d2	Model co-efficient	0.0060	
d3	Model co-efficient	0.001	
Kd	Death constant	0.00008	
pl	Inhibition factor p*.	50	
n	power coefficient	0.52	
m	Maintenance co-efficient.	0.0011	
x	Biomass concentration		g/l
s	Substrate concentration		g/l
p	Product concentration		g/l
dt	Sampling period	0.25	
u_{max}	Maximum growth rate		
K_s	Monod's constant		
Y_{xs}	Stochiometry coefficient for biomass growth		

Program *batch_seque.m* is developed to implement the decomposition method in a sequential way. The program calculates the optimal values of the state and control variables. The whole program is in appendix A. The function *modelbatch_func2* is used to calculate the trajectories of the derivatives of the state space vector, according to the Equations (2.78 – 80). Then the program is called by the solver *ode15s* for integration of the model equations. The input variables are *t* the time at moment *t* from the considered time interval *y* initial, or value of the state vector in the moment *t*. The output variables are the values of the state space vector derivatives *ydot*. The program is given in appendix A.

4.6.1 Results from sequential calculation of batch fermentation process

The program for sequential calculation is used to calculate the optimal trajectories of the control and state variables of the batch fermentation process. The optimal trajectories are presented in figure 4.3 -4.8.

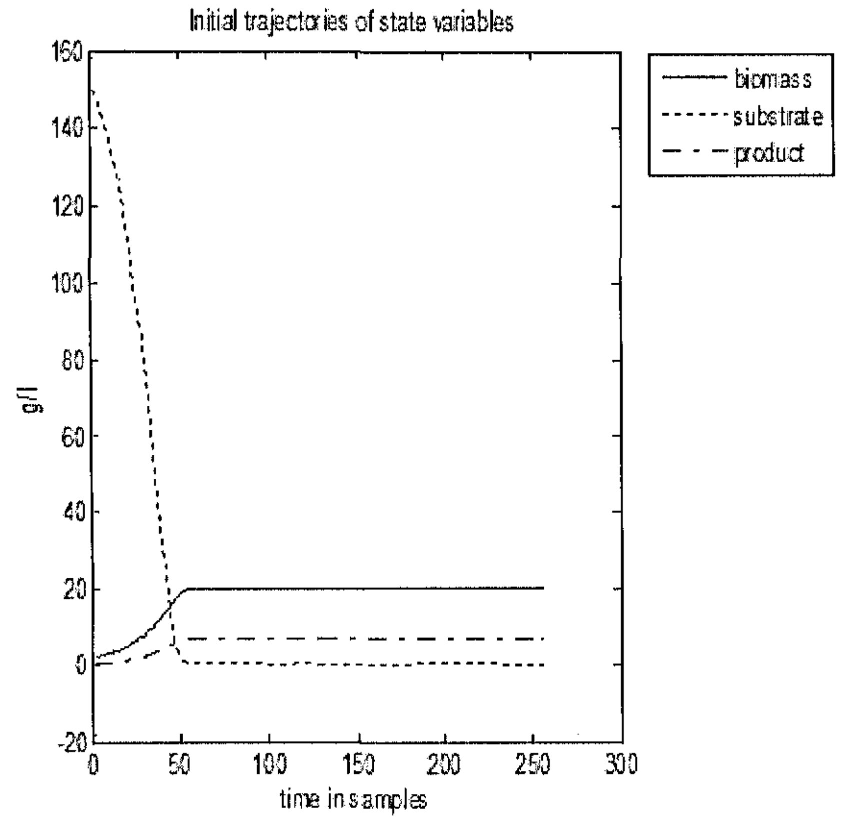


Figure 4.3: Initial trajectories of state variables for batch fermentation process

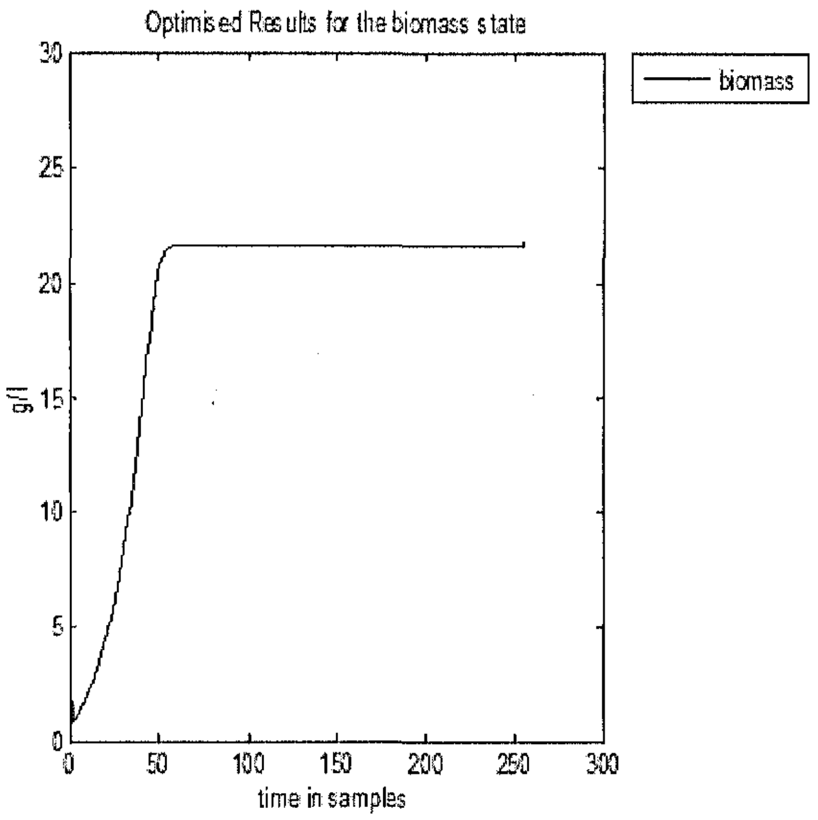


Figure 4.4: Optimal trajectory of biomass for batch fermentation process

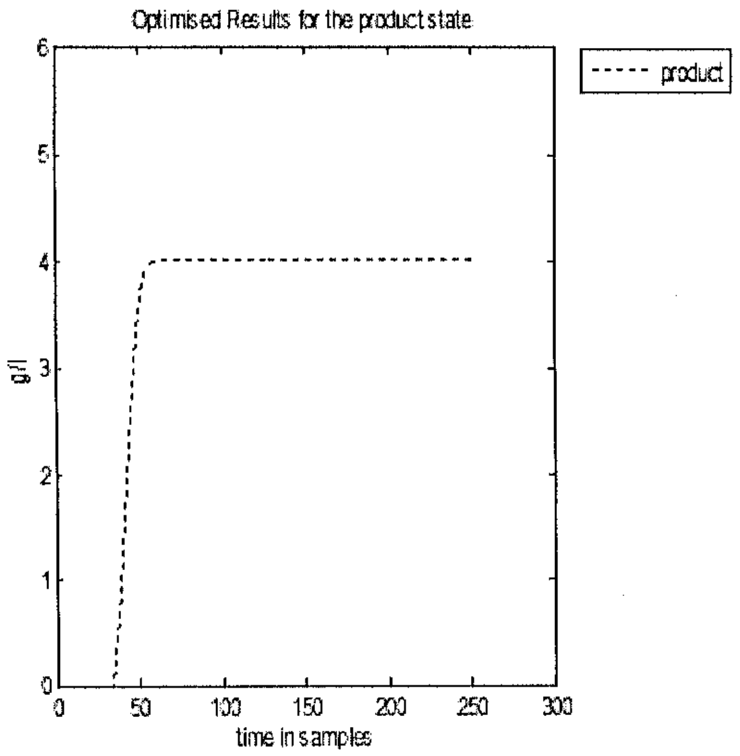


Figure 4.5: Optimal trajectory of product for batch fermentation process

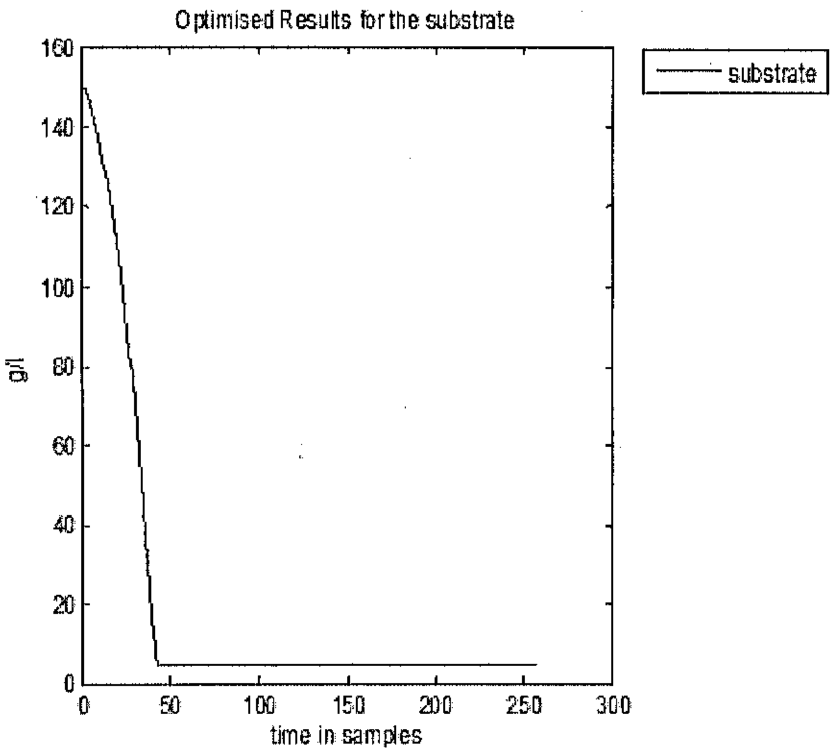


Figure 4.6: Optimal trajectory of substrate for batch fermentation process

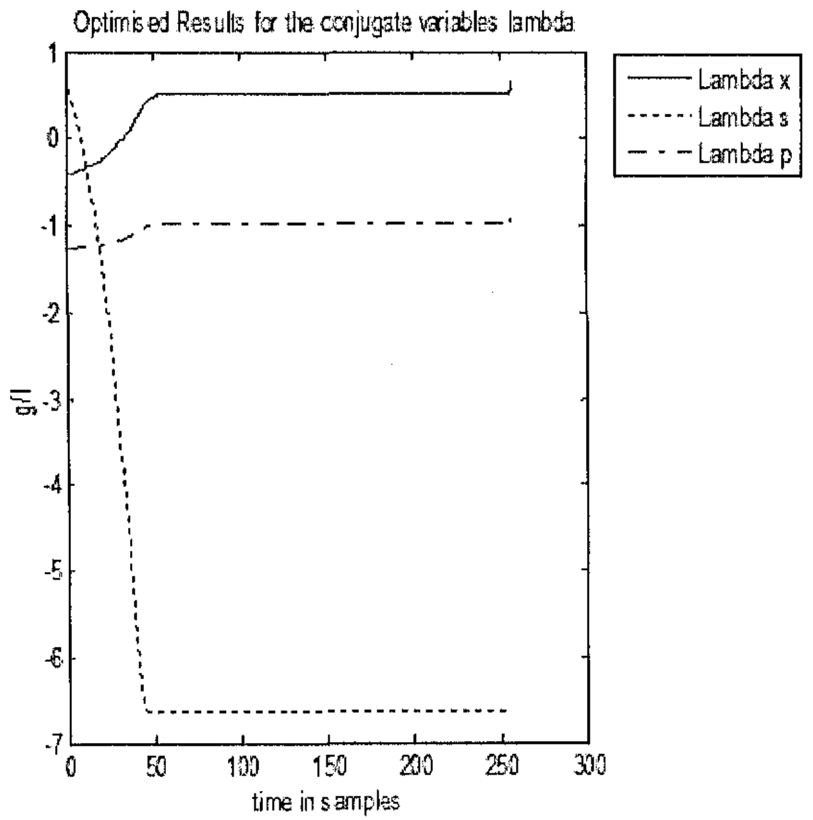


Figure 4.7: Optimal trajectory of conjugate variables for batch fermentation process

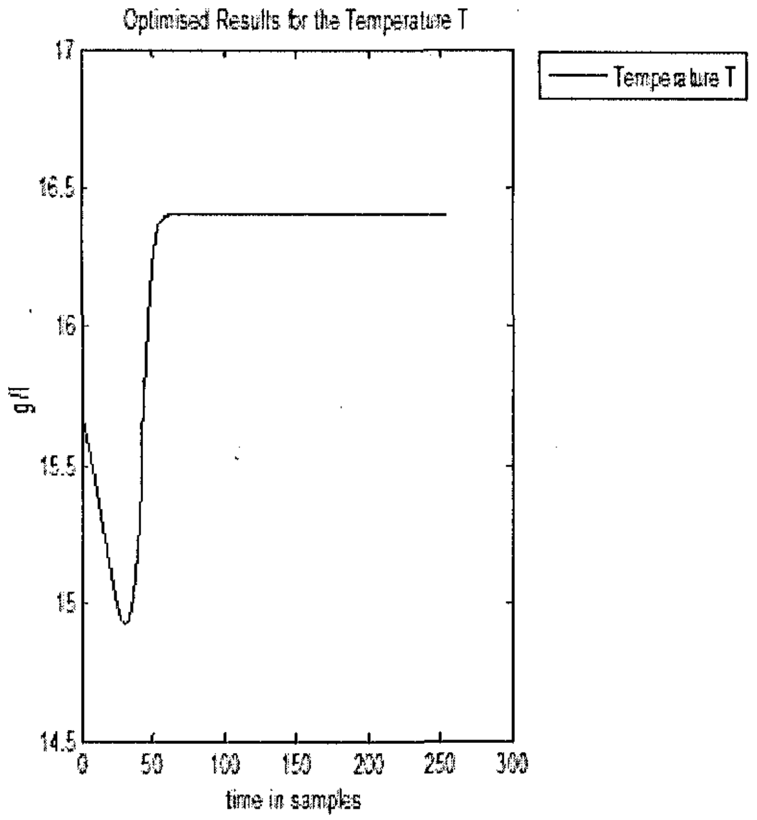


Figure 4.8: Optimal trajectory of temperature for batch fermentation process

4.7 Discussion of the results for batch fermentation process

Substrate and product levels showed the greatest rate of change about short time after inoculation. Thereafter they moved at a steady state, decreasing towards the end. These runs could be regarded as fairly typical batch fermentation process with a relative small inoculums.

4.8 Conclusion

This chapter describes the formulation of the problem for optimal control of a batch fermentation process. An augmented Lagrange functional is used and a decomposition method is developed for solving discrete time formulation of the problem, based on varational calculus and decomposition in time domain. The sequential algorithm of the decomposition method is describes and implemented in Matlab in serial mode of calculation. The parallel implementation of the decomposition method is described in chapter eight. The problem for optimal control of fed-batch fermentation process is considerd and solved in chapter five.

CHAPTER FIVE
DEVELOPMENT OF A DECOMPOSITION METHOD FOR OPTIMAL CONTROL PROBLEM CALCULATION FOR FED- BATCH FERMENTATION PROCESS

5.1 Introduction

The fed-batch fermentors are used to produce vaccines, enzymes, antibiotics, baker's yeast etc. It is difficult to model and control these processes because many uncertainties and nonlinearities are encounterd, the mechanism of the process is poorly understood, and the process variables are difficult to measure. The optimal control of fed-batch fermentors consists of two stages (Johnson, 1987) and (Maeda, 1987). Control variables such as pH, temperature, agitation and aeration are kept steady to pursue certain trajectories using conventional set point controllers (Fishman and Biryukov, 1974).

The first stage of control problem is considered in the thesis on the basis of dynamic mathematical model of *Saccharomyces cerevisiae* fermentation and finite horizon cost criterion for maximum production at the end of the process. This optimal control problem is characterized by nonquadratic criterion, nonlinear dynamic model with linear control and constraints over the process variables corresponding to the physical limitations. The problem can be solved by means of techniques based on the dynamic programming, Green's Theorem and Variational Calculus (Kishimoto et al., 1987). The optimal trajectory of the feedrate is singular kind of control. Nonlinear two point boundry value problem is solved which is characterized by many calculations and computing errors.

In section 5.2 formulation of the problem for optimal control of fed-batch fermentation process is developed. In the chapter decomposition method is firstly developed for batch processes to solve the above problems on the basis of an augmented Lagrange functional and new coordinating vector for its time domain decomposition is applied and is developed in section 5.3. It overcomes the pointed above difficulties. This method is extended here to solve the problem for optimal control of the fed-batch process in the case when the kinetic coefficients depend on the temperature. The algorithm of the method for solving the optimal control problem is developed in section 5.4

5.2 Formulation of the problem for optimal control of fed-batch fermentation process

The aim of the optimal control of the fed-batch fermentation process is to find trajectories of the feed flow rate and temperature: $F(k), T(k), k = \overline{0, K-1}$ such that the concentration of the biomass at the end of operation period. $k = \overline{0, K}$ is maximum:

$$J = x(K) \rightarrow \max \tag{5.1}$$

Under the model equations (2.88)-(2.91) and (2.72)-(2.75) and satisfy the constraints

$$\begin{aligned}
&x_{\min}(k) \le x(k) \le x_{\max}(k),\ k = \overline{0, K} \\
&s_{\min}(k) \le s(k) \le s_{\max}(k), k = \overline{0, K} \\
&p_{\min}(k) \le p(k) \le p_{\max}(k), k = \overline{0, K} \\
&V_{\min}(k) \le V(k) \le V_{\max}, k = \overline{0, K-1} \\
&T_{\min}(k) \le T(k) \le T_{\max}(k), k = \overline{0, K-1} \\
&F_{\min}(k) \le F(k) \le F_{\max}(k),\ k = \overline{0, K-1}
\end{aligned} \tag{5.2}$$

5.3 Augmented Lagrange functional formulation for fed-batch fermentation process

The considered problem for optimal control has the following characteristics:

- Criterion only at the final point of the optimization
- Nonlinear model with four state variables i.e. for fed-batch: biomass concentration(x), substrate concentration(s), Volume(V) and product concentration(p),two control variables i.e. temperature(T) and input flow rate(F)
- Long optimization horizon
- Affine control according to input flow rate

These characteristics determine which methodology to be selected for solution of the optimal control problem. The theory of variational calculus gives possibilities to handle problem with high dimension and nonlinearities by using the Lagrange functional. The affine control however in the case of non-quadratic criterion determines that the derivative of the Lagrange, according to this control does not depend on it. This creates problems with the solution of the two-point boundary value problem. In order to overcome the linear dependability of the Lagragian on the control variable an augmented functional of Lagrange is considered. The augmented Lagrange function is assumed here in the form:

$$L = x(K) + \sum_{k=0}^{K-1} \sum_{v=x,s,p,V} \{\lambda_v(k)[-v(k+1)+f_v(k)] + \frac{1}{2}\mu_v[-v(k+1)+f_v(k)]^2\} \tag{5.3}$$

where $\lambda_v, v = x,s,p,V$ is the vector of the conjugate variables, $\mu_v, v = x,s,p,V$ is the vector of the penalty coefficients. The augmented functional has to be maximized according to state and control variables and minimized according to conjugate variables on the basis of the theory of duality, under process constraints presented in point (5.2)

5.3.1 Necessary conditions for optimality for fed-batch fermentation process

Solution of the problem for optimal control is based on the necessary conditions for optimality of the functional of Lagrange. The equations of these conditions are used to build the algorithm for calculation of the optimal state and control variables. This algorithm is used for parallel calculation in a cluster of computers.

- Necessary conditions for optimality according to biomass

$$\frac{\partial L}{\partial x(k)} = \lambda_x(k)\left[1 + \Delta t\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{s(k)}{K_s + s(k)}\right] - \Delta t k_d - \frac{\Delta t F(k)}{V(k)}\right] -$$

$$-\mu_x(k)[x(k+1) - f_x(k)]\left[1 + \Delta t\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{s(k)}{K_s + s(k)}\right] - \Delta t k_d - \frac{\Delta t F(k)}{V(k)}\right] +$$

$$+\lambda_s(k)\left[\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{s(k)}{K_s + s(k)}\right] - \Delta t m\right] -$$

$$-\mu_s(k)[s(k+1) - f_s(k)]\left[\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{s(k)}{Ks + s(k)}\right] - \Delta t m\right] +$$

$$+\lambda_p(k)\left[\Delta t Y_{PX}\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{s(k)}{K_s + s(k)}\right]\right] -$$

$$-\mu_p(k)[p(k+1) - f_p(k)]\left[\Delta t Y_{PX}\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{s(k)}{K_s + s(k)}\right]\right] = 0 = e_x(k) k = \overline{0, K-1}$$

$$\frac{\partial L}{\partial x(K)} = 1 \tag{5.4}$$

- Necessary conditions for optimality according to substrate

$$\frac{\partial L}{\partial s(k)} = \lambda_x(k)\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)(K_s+s(k))-x(k)s(k)}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]+$$

$$+\lambda_s(k)\left[1-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]-\frac{\Delta tF(k)}{V(k)}\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[1-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]-\frac{\Delta tF(k)}{V(k)}\right]$$

$$+\lambda_p(k)\left[\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]=0=e_s(k), k=\overline{0,K-1}$$

$$\frac{\partial L}{\partial s(K)}=0 \tag{5.5}$$

- Necessary conditions for optimality according to product

$$\frac{\partial L}{\partial p(k)} = \lambda_x(k)\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]+$$

$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]+$$

$$+\lambda_s(k)\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]+$$

$$+\lambda_p(k)\left[1+\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]-\frac{\Delta tF(k)}{V(k)}\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[1+\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]-\frac{\Delta tF(k)}{V(k)}\right]$$

$$=0=e_p(k),\ k=\overline{0,K-1}$$

$$\frac{\partial L}{\partial p(K)}=0 \tag{5.6}$$

- Necessary conditions for optimality according to Volume

$$\frac{\partial L}{\partial V(k)}=\lambda_x(k)\left[\Delta t\frac{F(k)x(k)}{V^2(k)}\right]-\mu_x(k)[x(k+1)-f_x(k)]\left[\Delta t\frac{F(k)x(k)}{V^2(k)}\right]+$$

$$+\lambda_s(k)\left[\Delta t\frac{-F(k)}{V^2(k)}[s_i-s(k)]\right]-\mu_s(k)[s(k+1)-f_s(k)]\left[\Delta t\frac{-F(k)}{V^2(k)}[s_i-s(k)]\right]+$$

$$+\lambda_p(k)\left[-\Delta t\frac{-F(k)p(k)}{V^2(k)}\right]-\mu_s(k)[p(k+1)-f_p(k)]\left[\Delta t\frac{F(k)p(k)}{V^2(k)}\right]+$$

$$+\lambda_v(k)[1]-\mu_v(k)[V(k+1)-f_v(k)].[1]=0=e_v(k),k=\overline{0,K-1}$$

$$\frac{\partial L}{\partial V(K)}=0 \tag{5.7}$$

- Necessary conditions for optimality according to Flow rate

$$\frac{\partial L}{\partial F(k)}=\lambda_x(k)\left[\frac{-x(k)\Delta t}{V(k)}\right]-\mu_x(k)[x(k+1)-f_x(k)]\left[\frac{-\Delta tx(k)}{V(k)}\right]+$$

$$+\lambda_s(k)\left[\frac{\Delta t[s_i-s(k)]}{V(k)}\right]-\mu_s(k)[s(k+1)-f_s(k)]\left[\frac{\Delta t[s_i-s(k)]}{V(k)}\right]+$$

$$+\lambda_p(k)\left[\frac{-\Delta tp(k)}{V(k)}\right]-\mu_p(k)[p(k+1)-f_p(k)]\left[\frac{-\Delta tp(k)}{V(k)}\right]+$$

$$+\lambda_v(k)\Delta t-\mu_v(k)[V(k+1)-f_v(k)]\Delta t=0=e_F(k),\ k=\overline{0,K-1} \tag{5.8}$$

- Necessary conditions for optimality according to temperature

$$\frac{\partial L}{\partial T(k)}=\lambda_x(k)\left[\frac{\Delta t\partial\mu_{\max}}{\partial T}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\frac{\Delta t\partial\mu_{\max}}{\partial T}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]$$

$$+\lambda_s(k)\left[-\Delta t\frac{\frac{\partial\mu_{\max}}{\partial T}Y_{XS}-\mu_{\max}\frac{\partial Y_{XS}}{\partial T}}{Y^2{}_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\frac{\Delta t\partial\mu_{\max}}{\partial T}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]$$

$$+\lambda_s(k)\left[-\Delta t\frac{\frac{\partial\mu_{\max}}{\partial T}Y_{XS}-\mu_{\max}\frac{\partial Y_{XS}}{\partial T}}{Y^2{}_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\Bigg]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[-\Delta t\frac{\frac{\partial\mu_{\max}}{\partial T}Y_{XS}-\mu_{\max}\frac{\partial Y_{x/s}}{\partial T}}{Y^2{}_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\Bigg]+$$

$$+\lambda_p(k)\left[\Delta t\left[\frac{\partial Y_{PX}}{\partial T}\mu_{\max}+Y_{PX}\frac{\partial\mu_{\max}}{\partial T}\right]\frac{x(k)s(k)}{K_s+s(k)}\left[1-\frac{p(k)}{p^*}\right]^n+\right.$$

$$+\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\Bigg]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta t\left[\frac{\partial Y_{PX}}{\partial T}\mu_{\max}+Y_{PX}\frac{\partial\mu_{\max}}{\partial T}\right]\frac{x(k)s(k)}{K_s+s(k)}\left[1-\frac{p(k)}{p^*}\right]^n+\right. \qquad (5.9)$$

$$+\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\Bigg]=0=e_T(k)k=\overline{0,K-1}$$

Where the partial derivatives of the kinetic parameters according to the temperature are:

$$\left.\begin{array}{l}\frac{\partial\mu_{\max}}{\partial T}=a_2+2a_3T(k),\ \frac{\partial K_s}{\partial T}=b_2+2b_3T(k),\\ \frac{\partial Y_{XS}}{\partial T}=c_2+2c_3T(k),\ \frac{\partial Y_{PX}}{\partial T}=d_2+2d_3T(k),\end{array}\right\}\quad k=\overline{0,K-1} \qquad (5.10)$$

- Necessary conditions for optimality according to the conjugate variables

$$\frac{\partial L}{\partial\lambda_x(k)}=-x(k+1)+f_x(k)=0=e_{\lambda x}(k)k=\overline{0,K-1} \qquad (5.11)$$

$$\frac{\partial L}{\partial\lambda_s(k)}=-s(k+1)+f_s(k)=0=e_{\lambda s}(k)k=\overline{0,K-1} \qquad (5.12)$$

$$\frac{\partial L}{\partial \lambda_p(k)} = -p(k+1) + f_p(k) = 0 = e_{\lambda p}(k) k = \overline{0, K-1} \qquad (5.13)$$

$$\frac{\partial L}{\partial \lambda_v(k)} = -V(k+1) + f_v(k) = 0 = e_{\lambda v}(k) k = \overline{0, K-1} \qquad (5.14)$$

Two level computing structure is introduced where the set of equations (5.4)-(5.14) are solved using decomposition of the necessary conditions for optimality in time domain and coordination to combine the decomposed solutions in the original problem.

5.3.2 Coordinating vector for fed-batch fermentation process

The optimal trajectories of the variables can be calculated as a solution of the obtained set of nonlinear equations (5.4)-(5.14). It can be seen that the number of equations and variables is big. The solution could be found easier if the problem is decomposed. Decomposition in time domain could reduce complexity of the system equations (Tumara, 1975) and (Singandlitty, 1982).

The solutions in time domain will consists of K separate solutions, everyone at separate time moment k. The decomposition in time domain could be obtained on the basis of the coordinating procedure in two level computing structure(Figure 4.1) using the conjugate variables λ as coordinating ones(Tumara, 1975), but this method is applicable for normal functional of Lagrange. The used augmented functional could not be fully decomposed because of the quadratic terms and variables $x(k+1), s(k+1)$, $p(k+1)$ and $V(k+1)$ in them. They play a role of interconnections in time domain (Lin, 1992). This type of functional can be fully decomposed if the considered variables are selected also as a part of the coordinating vector. To overcome these difficulties it is proposed to consider

$$\delta_v(k) = v(k+1), v = x, s, p, V, k = \overline{0, K-1} \qquad (5.15)$$

as coordinating variables. Then the values of the coordinating variables are set from the second level of the two level calculating structure:

$$\lambda_v(k) = \lambda_v{}^j(k), v = x, s, p, V,\ k = \overline{0, K} \qquad (5.16)$$

$$\delta_v(k) = \delta_v{}^j(k), v = x, s, p, V,\ k = \overline{0, K} \qquad (5.17)$$

where j is the index of the coordinating process iterations. When the values of the coordinating variables from equations (5.16) – (5.17) are substituted into the Lagrange's functional, its full decomposition according to the discrete time moments k is obtained. The functional is decomposed into K sub-functionals $L_a(k)$ and each of them determines the optimal control and state at the given moment k. The above algorithm is implemented in Matlab software environment in two ways:

- Sequentially in one computer
- Parallel, using the cluster of computers

The decomposition in time domain is based on a two level calculation structure, given in Fig 4.1.

5.3.3 First level sub-problems for fed-batch fermentation process

The solutions of the first level sub-problems represent solution of the set of equations (5.4) – (5.10) in which the coordinating variables are substituted. These equations are solved for every separate moment of time in a parallel way. The structure of the equations (5.4) – (5.10) does not allow to use analytical solution. Gradient method is used in the form:

$$\nu^{j,i+1}(k) = \nu^{j,i}(k) + \alpha_\nu e_\nu^{\ j,i}(k), \nu = x, s, p, V, T, F, \ k = \overline{0, K-1} \tag{5.18}$$

$$v^{j,i+1}(K) = \ f_v[(K-1), T(K-1), F(K-1)], v = x, s, p, V$$

where $\alpha_\nu > 0$ are the steps of the gradient procedures, $e_\nu(k)$ are the gradients according to the equations (5.4) – (5.10) and *i* is the iteration index of the gradient procedures. The first level sub-problems are determined under the set from the second level coordinating variables according to the necessary conditions for optimality of the sub-functionals $L_a(k)$. Equations (5.4)-(5.10) can be simplified further by using necessary conditions for optimality according to the conjugate variables presented by (5.11)–(5.13) as follows:

For the biomass

$$\frac{\partial L}{\partial x(k)} = e_x^{j,i}(k) = \left[1 + \Delta t\mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^n \frac{s(k)}{K_s + s(k)} - \Delta t k_d - \Delta t\frac{F(k)}{V(k)}\right]\left[\lambda_x{}^j(k) - \mu_x e_{\lambda x}{}^j(k)\right] +$$
$$+\left[\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p*}\right]^n \frac{s(k)}{K_s + s(k)} - \Delta t m\right]\left[\lambda_s{}^j(k) - \mu_s e_{\lambda s}{}^j(k)\right] +$$
$$+\left[\Delta t Y_{PX}\mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^n \frac{s(k)}{K_s + s(k)}\right]\left[\lambda_p{}^j(k) - \mu_p e_{\lambda p}{}^j(k)\right], k = \overline{0, K-1} \tag{5.19}$$

For the substrate

$$\frac{\partial L}{\partial s(k)} = e_s^{j,i}(k) = \left[\Delta t\mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^n \frac{x(k)K_s}{[K_s + s(k)]^2}\right]\left[\lambda_x(k) - \mu_x e_{\lambda x}(k)\right] +$$
$$+\left[1 - \frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p*}\right]^n \frac{s(k)K_s}{[K_s + s(k)]^2} - \Delta t\frac{F(k)}{V(k)}\right]\left[\lambda_s{}^j(k) - \mu_s e_{\lambda s}{}^j(k)\right] + \tag{5.20}$$
$$+\left[\Delta t Y_{PX}\mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^n \frac{x(k)s(k)}{K_s + s(k)}\right]\left[\lambda_p{}^j(k) - \mu_p e_{\lambda p}{}^j(k)\right], k = \overline{0, K-1}$$

For the product

$$\frac{\partial L}{\partial p(k)} = e_p^{j,i}(k) = \left[\Delta t\mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^{n-1} n\left[\frac{-1}{p*}\right]\frac{x(k)s(k)}{K_s + s(k)}\right]\left[\lambda_x{}^j(k) - \mu_x e_{\lambda x}{}^j(k)\right]$$
$$\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p*}\right]^{n-1} n\left[\frac{-1}{p*}\right]\frac{x(k)s(k)}{K_s + s(k)}\right]\left[\lambda_s{}^j(k) - \mu_s e_{\lambda s}{}^j(k)\right] +$$
$$+\left[1 + \Delta t Y_{PX}\mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^{n-1} n\left[\frac{-1}{p*}\right]\frac{x(k)s(k)}{K_s + s(k)} - \Delta t\frac{F(k)}{V(k)}\right]\left[\lambda_p{}^j(k) - \mu_p e_{\lambda p}{}^j(k)\right], k = \overline{0, K-1} \tag{5.21}$$

For the Volume

$$\frac{\partial L}{\partial V(k)} = e_V^{j,i}(k) = \left[\Delta t\frac{F(k)x(k)}{V^2(k)}\right]\left[\lambda_x(k) - \mu_x e_{\lambda x}(k)\right] +$$

$$+\left[-\frac{\Delta t F(k)}{V^2(k)}\left[s_i - s(k)\right]\right]\left[\lambda_s{}^j(k) - \mu_s e_{\lambda s}{}^j(k)\right]+$$
$$+\left[\Delta t\frac{F(k)p(k)}{V^2(k)}\right]\left[\lambda_p{}^j(k) - \mu_p e_{\lambda p}{}^j(k)\right]+ \tag{5.22}$$
$$+\left[\lambda_V{}^j(k) - \mu_V e_{\lambda V}{}^j(k)\right], k = \overline{0, K-1}$$

For the flow rate

$$\frac{\partial L}{\partial F(k)} = e_F{}^{j,i}(k) = \left[-\Delta t\frac{x(k)}{V(k)}\right]\left[\lambda_x(k) - \mu_x e_{\lambda x}(k)\right]+$$
$$+\left[\frac{\Delta t}{V(k)}\left[s_i - s(k)\right]\right]\left[\lambda_s{}^j(k) - \mu_s e_{\lambda s}{}^j(k)\right]+$$

$$+\left[-\Delta t\frac{p(k)}{V(k)}\right]\left[\lambda_p{}^j(k) - \mu_p e_{\lambda p}{}^j(k)\right]+ \tag{5.23}$$
$$+\Delta t\left[\lambda_V{}^j(k) - \mu_V e_{\lambda V}{}^j(k)\right] = e_F(k), k = \overline{0, K-1}$$

For the temperature

$$\frac{\partial L}{\partial T(k)} = e_T(k) =$$
$$= \left[\frac{\Delta t \partial \mu_{\max}}{\partial T}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} - \Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)\dfrac{\partial K_s}{\partial T}}{\left[K_s + s(k)\right]^2}\right]\right]*$$
$$*\left[\lambda_x(k) - \mu_x(k)e_{\lambda x}(k)\right] + \left[-\Delta t\frac{\dfrac{\partial \mu_{\max}}{\partial T}Y_{XS} - \mu_{\max}\dfrac{\partial Y_{XS}}{\partial T}}{Y^2{}_{XS}}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)}+\right.$$

$$\left.+\frac{\Delta t \mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)\dfrac{\partial K_s}{\partial T}}{\left[K_s + s(k)\right]^2}\right]\right]\left[\lambda_s(k) - \mu_s(k)e_{\lambda s}(k)\right]+$$

$$+\left[\Delta t\frac{\frac{\partial\mu_{\max}}{\partial T}Y_{XS}+\mu_{\max}\frac{\partial Y_{XS}}{\partial T}}{Y^2{}_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$\left.-\Delta t\mu_{\max}Y_{PX}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right][\lambda_p(k)-\mu_p(k)e_{\lambda p}(k)],k=\overline{0,K-1} \qquad (5.24)$$

These equations can be simplified further for the purpose of calculations. Some expressions in them are repeated, that is why the common notations are used to represent them:

$$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\frac{s(k)}{K_s+s(k)}=\varphi_x(k) \qquad (5.25)$$

$$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1}\frac{1}{p^*}n\frac{x(k)s(k)}{K_s+s(k)}=\varphi_p(k) \qquad (5.26)$$

$$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)K_s}{[K_s+s(k)]^2}=\varphi_s(k) \qquad (5.27)$$

$$\Delta t\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}=\varphi_1(k) \qquad (5.28)$$

$$\Delta t\frac{F(k)}{V^2(k)}=\varphi_v(k) \qquad (5.29)$$

$$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{[K_s+s(k)]^2}=\varphi_2(k),k=\overline{0,K-1} \qquad (5.30)$$

Using notations from (5.25) - (5.30) equations from (5.19)-(5.24) can be written in the form:

For the biomass

$$\frac{\partial L}{\partial x(k)} = e_x^{\,j,i}(k) = \left[1 + \varphi_x^{\,j,i}(k) - \Delta t k_d - \Delta t \frac{F(k)}{V(k)}\right]\left[\lambda_x^{\,j}(k) - \mu_x(k) e_{\lambda x}^{\,j}(k)\right] +$$
$$+ \left[\frac{\varphi_x^{\,j,i}(k)}{Y_{XS}} - \Delta t m\right]\left[\lambda_s^{\,j}(k) - \mu_s(k) e_{\lambda s}^{\,j}(k)\right] + \qquad (5.31)$$
$$+ \varphi_x^{\,j,i}(k) Y_{PX}\left[\lambda_p^{\,j}(k) - \mu_p(k) e_{\lambda p}^{\,j}(k)\right], k = \overline{0, K-1}$$

For the substrate

$$\frac{\partial L}{\partial s(k)} = e_s^{\,j,i}(k) = \varphi_s^{\,j,i}(k)\left[\lambda_x^{\,j}(k) - \mu_x(k) e_{\lambda x}^{\,j}(k)\right] +$$
$$+ \left[1 - \frac{\varphi_s^{\,j,i}(k)}{Y_{XS}} - \Delta t \frac{F^{j,i}(k)}{V^{j,i}(k)}\right]\left[\lambda_s^{\,j}(k) - \mu_s(k) e_{\lambda s}^{\,j}(k)\right] + \qquad (5.32)$$
$$+ Y_{PX} \varphi_s(k)\left[\lambda_p^{\,j}(k) - \mu_p(k) e_{\lambda p}^{\,j}(k)\right], k = \overline{0, K-1}$$

For the product

$$\frac{\partial L}{\partial p(k)} = e_p^{\,j,i}(k) = \left[\varphi_p(k)\right]\left[\lambda_x^{\,j}(k) - \mu_x e_{\lambda x}^{\,j}(k)\right]$$
$$\left[-\frac{\varphi_p(k)}{Y_{XS}}\right]\left[\lambda_s^{\,j}(k) - \mu_s e_{\lambda s}^{\,j}(k)\right] + \qquad (5.33)$$
$$+ \left[1 - \varphi_p(k) - \Delta t \frac{F(k)}{V(k)}\right]\left[\lambda_p^{\,j}(k) - \mu_p(k) e_{\lambda p}^{\,j}(k)\right], k = \overline{0, K-1}$$

Flow the volume

$$\frac{\partial L}{\partial V(k)} = e_V^{\,j,i}(k) = \varphi_V(k) x(k)\left[\lambda_x^{\,j}(k) - \mu_x(k) e_{\lambda x}^{\,j}(k)\right] -$$
$$- \varphi_V(k)\left[s_i - s(k)\right]\left[\lambda_s^{\,j}(k) - \mu_s(k) e_{\lambda s}^{\,j}(k)\right] +$$
$$+ \varphi_V(k) p(k)\left[\lambda_p^{\,j}(k) - \mu_p(k) e_{\lambda p}^{\,j}(k)\right] + \left[\lambda_V^{\,j}(k) - \mu_V(k) e_{\lambda V}^{\,j}(k)\right] \qquad (5.34)$$

For the flow rate

$$\frac{\partial L}{\partial F(k)} = e_F^{\,j,i}(k) =$$
$$= -\frac{\Delta t x(k)}{V(k)}\left[\lambda_x^{\,j}(k) - \mu_x(k) e_{\lambda x}^{\,j}(k)\right] + \frac{\Delta t[s_i - s(k)]}{V(k)}\left[\lambda_s^{\,j}(k) - \mu_s(k) e_{\lambda s}^{\,j}(k)\right] -$$
$$- \frac{\Delta t p(k)}{V(k)}\left[\lambda_p^{\,j}(k) - \mu_p(k) e_{\lambda p}^{\,j}(k)\right] + \Delta t\left[\lambda_V^{\,j}(k) - \mu_V(k) e_{\lambda V}^{\,j}(k)\right], k = \overline{0, K-1} \tag{5.35}$$

For the temperature

$$\frac{\partial L}{\partial T(k)} = e_T^{\,j,i}(k) =$$
$$= \left[\varphi_1(k)\frac{\partial \mu_{\max}}{\partial T} - \varphi_2(k)\frac{\partial K_s}{\partial T}\right]\left[\lambda_x^{\,j}(k) - \mu_x(k) e_{\lambda x}^{\,j}(k)\right] + \left[\left[-\varphi_1(k)\right]\left[\frac{\frac{\partial \mu_{\max}}{\partial T} Y_{XS} - \mu_{\max}\frac{\partial Y_{XS}}{\partial T}}{Y^2_{XS}}\right] + \right.$$
$$\left. + \frac{\varphi_2(k)}{Y_{XS}}\frac{\partial K_s}{\partial T}\right]\left[\lambda_s^{\,j}(k) - \mu_s(k) e_{\lambda s}^{\,j}(k)\right] + \left[\varphi_1(k)\left[\frac{\partial Y_{PX}}{\partial T}\mu_{\max} + Y_{PX}\frac{\partial \mu_{\max}}{\partial T}\right] - \right.$$
$$\left. - Y_{PX}\varphi_2(k)\frac{\partial K_s}{\partial T}\right]\left[\lambda_p^{\,j}(k) - \mu_p(k) e_{\lambda p}^{\,j}(k)\right], k = \overline{0, K-1} \tag{5.36}$$

The calculation of the gradients for every step is done with the previous values of the state and control variables $\nu^{j,i}$, $\nu = x, s, p, V, F, T$. The calculation with the gradient procedure will continue until the necessary conditions for optimality towards state and control variables are fulfilled. This means that the values of the gradients are approximately equal to zero:

$$\left\| e_\nu(k) \right\| \le \varepsilon_\nu, \varepsilon_\nu > 0, \nu = x, s, p, V, F, T, k = \overline{0, K-1} \tag{5.37}$$

The values of the state variables, the final point are calculated from the optimal values at the previous point according to equation (5.18). The new calculated values have to be checked for belonging to the area of constraints.

$$x_{\min} \le x(k) \le x_{\max} \;, k = \overline{0,K} \tag{5.38}$$

$$s_{\min} \le s(k) \le s_{\max}, \;\; k = \overline{0,K} \tag{5.39}$$

$$p_{\min} \le p(k) \le p_{\max}, \;\; k = \overline{0,K} \tag{5.40}$$

$$V_{\min} \le V(k) \le V_{\min}, \;\; k = \overline{0,K} \tag{5.41}$$

$$F_{\min} \le F(k) \le F_{\max}, k = \overline{0,K-1} \tag{5.42}$$

$$T_{\min} \le T(k) \le T_{\max}, k = \overline{0,K-1} \tag{5.43}$$

Simple way to do and correct the obtained new values is to project them over the constraints domain according to the relation:

$$v^{j,i}(k) = \left\{ \begin{array}{l} v_{\min}, \; if \;\; v^{j,i}(k) < v_{\min} \\ v^{j,i}(k), \;\; if \;\; v_{\min} \le v^{j,i}(k) \le v_{\max}, \\ v^{j,i}(k) > v_{\max} \\ k = \overline{0,K} \;\; \text{for } v = x, s, p, V \;\; \text{and } k = \overline{0,K-1} \;\; \text{for } \; v = T, F \end{array} \right\} \tag{5.44}$$

The obtained values of the trajectories of the state and control variables are function of the coordinating variables. They are optimal for the given values of the coordinating variables. If the coordinating trajectories are optimal ones, then the obtained state and control variables also will be the optimal according to the theory of the duality in optimal control and Lagrange methods. The optimal values of the coordinating variables are obtained by the solution of the coordinating problem.

5.3.4 The coordinating sub-problem for fed-batch fermentation process

The coordinating variables are obtained from the necessary conditions for optimality of the augmented Lagrange functional according to them. For the conjugate variables these are equations (5.11)-(5.14). The necessary conditions for optimality according to the new introduced interconnections in time domain are:

$$\frac{\partial L}{\partial \delta_x(k)} = -\lambda_x(k) - \mu_x[\delta_x(k) - f_x(k)] = -\lambda_x(k) - \mu_x e_{\lambda x}(k) = e_{\delta x}(k) = 0 \tag{5.45}$$

$$\frac{\partial L}{\partial \delta_s(k)} = -\lambda_s(k) - \mu_s[\delta_s(k) - f_s(k)] = -\lambda_s(k) - \mu_s e_{\lambda_s}(k) = e_{\delta_s}(k) = 0 \quad (5.46)$$

$$\frac{\partial L}{\partial \delta_p(k)} = -\lambda_p(k) - \mu_p[\delta_p(k) - f_p(k)] = -\lambda_p(k) - \mu_p e_{\lambda_p}(k) = e_{\delta_p}(k) = 0 \quad (5.47)$$

$$\frac{\partial L}{\partial \delta_V(k)} = -\lambda_v(k) - \mu_v[\delta_v(k) - f_v(k)] = -\lambda_v(k) - \mu_v e_{\lambda_v}(k) = e_{\delta_v}(k) = 0 \quad (5.48)$$

where $\delta_v(k) - f_v(k) = e_{\lambda v}(k), v = x, s, p, V$according to equations (5.11)-(5.14). The necessary conditions for optimality (5.11)-(5.14) and (5.44)-(5.47) can be combined in a gradient procedure. Equations (5.44)-(5.47) give the values of the gradients for the conjugate variables multiplied by the penalty co-efficient. $\mu_v, \quad v = x, s, p, V$

It appears that:

$$\frac{\partial L}{\partial \delta_v(k)} = -\lambda_v(k) - \mu_v \frac{\partial L}{\partial_{\lambda_v}(k)} = -\lambda_v - \mu_v e_{\lambda_v}(k), \quad k = \overline{0, K-1} \quad (5.49)$$

Gradient procedure can be built for both type of coordinating variables.

For conjugate variables:

$$\lambda_v^{j+1}(k) = \lambda_v^{j}(k) - \alpha_{\lambda_v} e^j{}_{\lambda v}(k), \quad (5.50)$$

where

$$e^j{}_{\lambda v}(k) = -\delta_v^{j}(k) + f^j{}_v(k), v = x, s, p, V, k = \overline{0, K-1}, \quad (5.51)$$

$$\delta_v^{j+1}(k) = \delta_v^{j}(k) + \alpha_{\delta_v} e_{\delta_v}(k), \; k = \overline{0, K-1} \quad (5.52)$$

where α_{λ_v} and α_{δ_v} are the steps of the procedures for calculation of the conjugate variables and the interconnection in time, respectively. As the calculations are based on gradient iterative procedure, they will stop when the gradient is going to zero, or practically achieving some conditions for optimality, given by

$$\|e^{j+1}{}_{\lambda v}(k) - e^j{}_{\lambda v}(k)\| \le \varepsilon_{\lambda v}, \varepsilon_{\lambda v} > 0, k = \overline{0, K}$$

$$\|e^{j+1}{}_{\delta v}(k) - e^j{}_{\delta v}(k)\| \le \varepsilon_{\delta v}, \varepsilon_{\delta v} > 0, k = \overline{0, K-1} \quad (5.53)$$

where $\varepsilon_{\delta v}$, ε_{λ_v} are a very small positive numbers. For every step of the gradient procedure for the coordinating process the full iteration cycle of the sub problems of the first level is done. The calculations of the coordinating variables are done with the values of state and control variables of the first level sub problems and the calculations of the first level sub problems are done with the values of the calculated on the previous iteration coordinating variables. The iterative process of coordination and the optimal solution of the whole problem for optimal control is obtained when the optimal solution of the coordinating sub problem is reached, this means conditions (5.52) are fulfilled and if this condition is not reached the error is computed.

$$e^{j+1} = \left\{\left\|e^{j+1}{}_{\lambda v}\right\|^2 + \left\|e^{j+1}{}_{\delta v}\right\|^2\right\}^{1/2} \qquad (5.54)$$

The new penalty coefficient is calculated from the conditions.

$$if\; j = 1\; or\; e^{j+1} < e^j\; , then\; \mu^{j+1} = \mu^j \qquad (5.55)$$

$$if\; j > 1 and\; e^{j+1} \geq e^j \text{ then } \mu^{j+1} = \alpha\mu^j, \alpha = [0.1\;\; 10.0] \qquad (5.56)$$

5.4 Algorithm of the method for fed-batch fermentation process

The calculations in the two-level structure can be grouped and represented in the following algorithm:

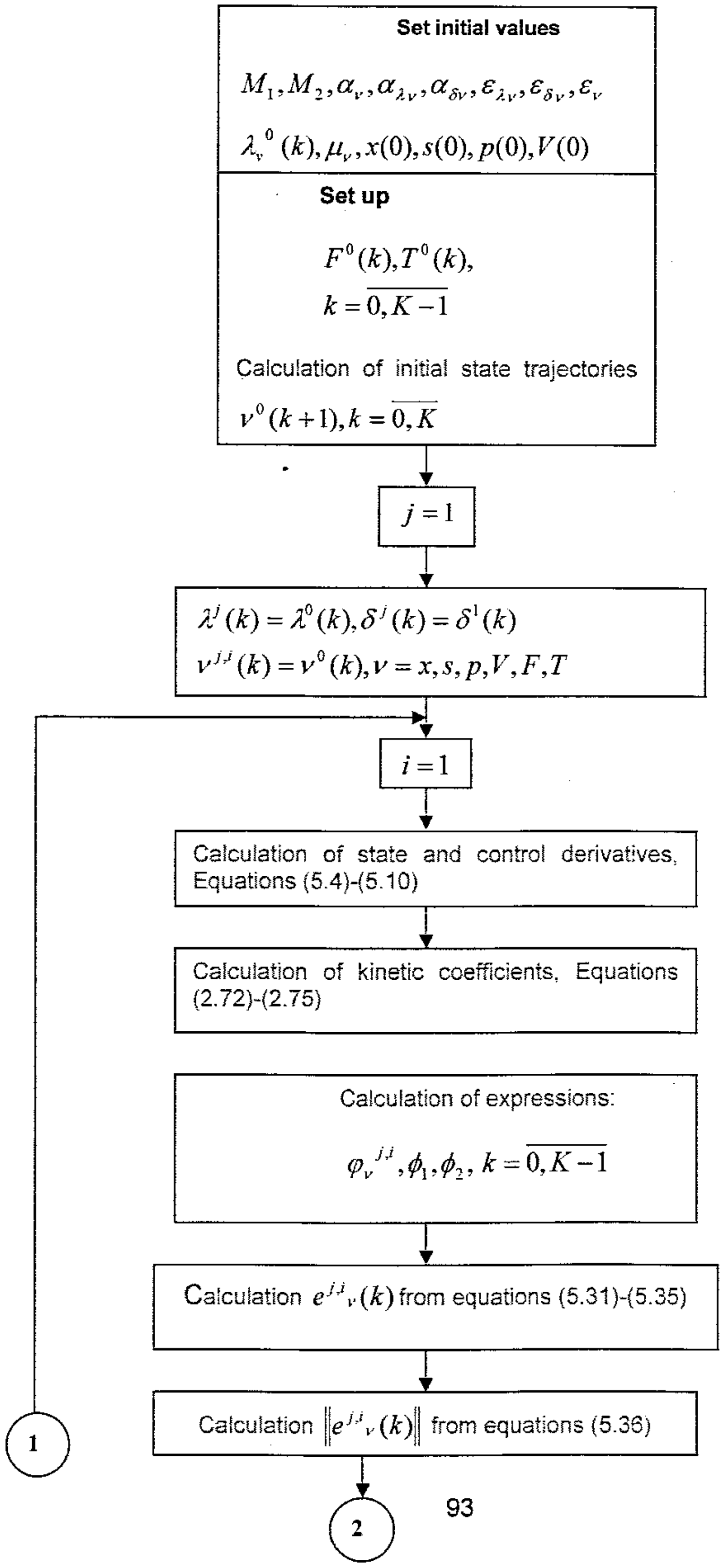
On the coordinating level the initial values of the coordinating and gradient procedures, initial trajectories of the state, control and coordinating variables are set as follows:

- Maximal number of step of the coordinating process – M2.
- Maximal number of steps for state and control variables gradient procedure- M1.
- Steps of the gradient procedures
  - For coordinating conjugate variables - $\alpha_{\lambda_v}, v = x, s, p, V$.
  - For coordinating interconnection in time - $\alpha_{\delta_v}, v = x, s, p, V$.
  - . For state variables and control variable $\alpha_v, v = x, s, p, T, F$
- Values for error and termination of the calculations for coordinating procedures. $\varepsilon_{\lambda_v}, \varepsilon_{\delta_v}, v = x, s, p, V$ and for gradient procedures on first level. $\varepsilon_v, v = x, s, p, V$
- Initial values trajectories of the conjugate variables. $\lambda^1{}_v(k), v = x, s, p, V, k = \overline{0, K-1}$
- Initial values of the state variables, $x(0)$, $s(0)$, $p(0)$ and V(0).
- Number of steps in the optimization horizon – K.
- Initial trajectories of the control variables $T^{1,1}(k),\; F^{1,1}(k), k = \overline{0, K-1}$

- The initial trajectories of the control variables are substituted in the model equations (2.88)-(2.91) and (2.72)-(2.75) and the state trajectories are calculated.
- The initial trajectories of the interconnections in time domain are calculated from $\delta_v(k) = v(k+1), k = \overline{0, K-1}$, $v = x, s, p, V$
- Initial trajectories of the coordinating variables and initial state and control trajectories are sent to the first level.

2. On the first level the following calculations are done:
   - Derivatives of equations (4.8) are calculated.
   - The kinetic coefficients $\mu_{\max}, K_s, Y_{XS}, Y_{PX}$ from equations (2.72)-(2.75) are calculated.
   - The expressions $\varphi_x(k), \varphi_s(k), \varphi_p(k), \varphi_V, \varphi_1(k), \varphi_2(k), k = \overline{0, K-1}$ are calculated from equations (5.25)-(5.30).
   - The gradients $e_v^{j,i}(k), k = \overline{0, K-1}, v = x, s, p, V, F, T$ are calculated from equations (5.31)-(5.36).
   - The errors for the end of the gradient procedures are calculated according to equation (5.37). If the error is less than ε_v, the calculations of the first level stop and the current values of the state and control variables are the optimal ones. If the error conditions are not satisfied the new values of the state and control values are calculated.
   - The new values of state and control variables are calculated from equation (5.18).
   - The obtained state and control variables are projected over the constraint domain according to equation (5.44). The obtained trajectories are sent to the second level.

3. On the second level the following calculations are done:
   - The final value of state trajectories *v(k)* are calculated
   - The gradients $e_{\lambda_v}(k), k = \overline{0, K-1}, v = x, s, p, V$ are calculated from equations (5.11)-(5.14).
   - The conditions for the end of the calculations are checked according to Equation (5.53). If the conditions are fulfilled the optimal solution of the coordinating and of the whole problem are obtained. If the conditions are not satisfied the new values of the coordinating variables are calculated.
   - New values of the conjugate variables are calculated according to equation (5.50)-(5.51).
   - The new values of the interconnections are calculated from equation (5.52). These trajectories are sent to the first level sub problems where the calculations from p.2a till 3b are repeated and so on. The schematic diagram of the algorithm is given on Fig 5.1



Set initial values
$M_1, M_2, \alpha_\nu, \alpha_{\lambda\nu}, \alpha_{\delta\nu}, \varepsilon_{\lambda\nu}, \varepsilon_{\delta\nu}, \varepsilon_\nu$
$\lambda_\nu^{\ 0}(k), \mu_\nu, x(0), s(0), p(0), V(0)$
Set up
$F^0(k), T^0(k),$
$k = \overline{0, K-1}$
Calculation of initial state trajectories
$\nu^0(k+1), k = \overline{0, K}$
$j = 1$
$\lambda^j(k) = \lambda^0(k), \delta^j(k) = \delta^1(k)$
$\nu^{j,j}(k) = \nu^0(k), \nu = x, s, p, V, F, T$
$i = 1$
Calculation of state and control derivatives, Equations (5.4)-(5.10)
Calculation of kinetic coefficients, Equations (2.72)-(2.75)
Calculation of expressions:
$\varphi_\nu^{\ j,i}, \phi_1, \phi_2,\ k = \overline{0, K-1}$
Calculation $e^{j,i}{}_\nu(k)$ from equations (5.31)-(5.35)
Calculation $\left\| e^{j,i}{}_\nu(k) \right\|$ from equations (5.36)
1
2

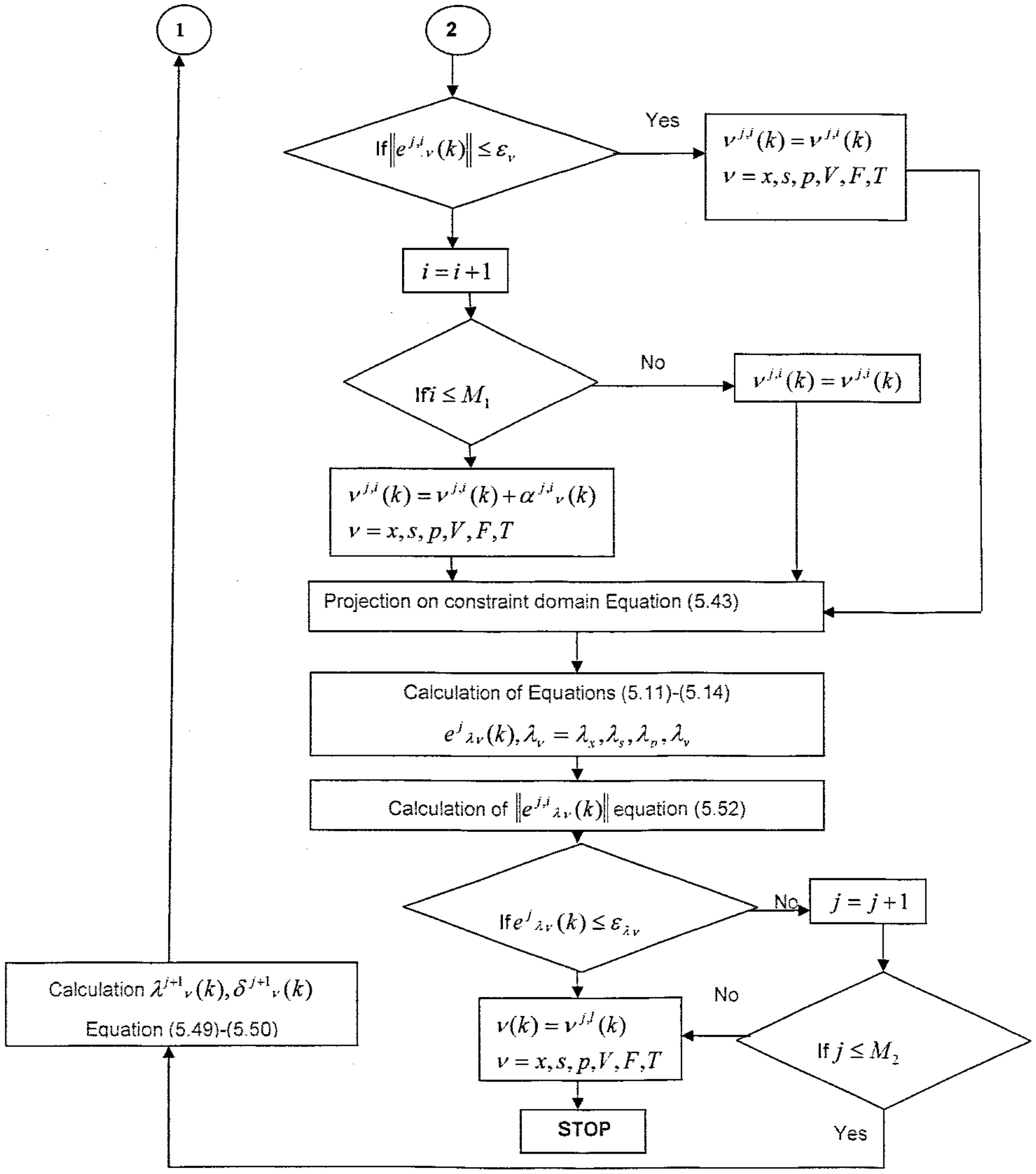

1
2
If $\|e^{j,i}{}_{\nu}(k)\| \le \varepsilon_{\nu}$
Yes
$\nu^{j,i}(k) = \nu^{j,i}(k)$
$\nu = x, s, p, V, F, T$
$i = i + 1$
If $i \le M_1$
No
$\nu^{j,i}(k) = \nu^{j,i}(k)$
$\nu^{j,i}(k) = \nu^{j,i}(k) + \alpha^{j,i}{}_{\nu}(k)$
$\nu = x, s, p, V, F, T$
Projection on constraint domain Equation (5.43)
Calculation of Equations (5.11)-(5.14)
$e^{j}{}_{\lambda\nu}(k), \lambda_{\nu} = \lambda_x, \lambda_s, \lambda_p, \lambda_v$
Calculation of $\|e^{j,i}{}_{\lambda\nu}(k)\|$ equation (5.52)
If $e^{j}{}_{\lambda\nu}(k) \le \varepsilon_{\lambda\nu}$
No
$j = j + 1$
Calculation $\lambda^{j+1}{}_{\nu}(k), \delta^{j+1}{}_{\nu}(k)$
Equation (5.49)-(5.50)
No
$\nu(k) = \nu^{j,l}(k)$
$\nu = x, s, p, V, F, T$
If $j \le M_2$
STOP
Yes

Figure 5.1: Flow chart of the algorithm for calculation of optimal trajectories of fed-batch fermentation process

5.5 Description and calculation results from the sequential program of the algorithm of the method in Matlab for fed-batch fermentation process

The algorithm presented in section 5.4 is implemented in Matlab in a sequential way(Appendix B). The table below represents the relationship between mathematical notation and program notation of the Matlab program.

Table 5.1: Correspondence between mathematical and programs notations for Fed-batch fermentation process

Name	Notation in the text	Notation in Matlab
State variable Biomass	x	x
Substrate	s	s
Product	p	p
volume	v	v
Control variable Temperature	T	T
Flow rate	F	F
Steps of the gradient procedures For state variables	α_x	ax
	α_s	as
	α_p	ap
	α_v	av
	$\alpha_{\lambda x}$	alx
	$\alpha_{\lambda s}$	als
	$\alpha_{\lambda p}$	alp
	$\alpha_{\lambda v}$	alv
	$\alpha_{\delta x}$	adx
	$\alpha_{\delta s}$	ads
	$\alpha_{\delta v}$	adv
Conjugate variable	λ_x	lx
	λ_s	ls
	λ_p	lp
	λ_v	lv

Interconnections in time domain	δ_x	dx
	δ_s	ds
	δ_p	dp
	δ_v	dv
Coefficients	a_1, a_2, a_3	a1, a2, a3
	b_1, b_2, b_3	b1, b2, b3
	c_1, c_2, c_3	c1, c2, c3
	d_1, d_2, d_3	d1, d2, d3
	K_d	Kd
	K_s	Ks
	m	m
	n	n
	Y_{xs}	Yxs
	Y_{px}	Ypx
	s_i	si
	p	pl
	Δt	dt
	$\mu_{\max}$	mmax
Initial values of state variables	x(0)	x0
	s(0)	s0
	p(0)	p0
	v(0)	v0
Penalty coefficients	μ_x	mx
	μ_s	ms
	μ_p	mp
	μ_v	mv
Min and max of state variables	x_{max}	xmax
	x_{min}	xmin
	s_{max}	smax
	s_{min}	smin
	p_{max}	pmax

	p_{min}	pmin
	v_{max}	vmax
	v_{min}	vmin
Min and max of control variables	T_{max}	Tmax
	T_{min}	Tmin
	F_{max}	Fmax
	F_{min}	Fmin
Derivaives of the model parameters	$\frac{\partial \mu_{\max}}{\partial T} = a_2 + 2a_3 T(k)$	dmmaxdT(k)= a2+2*a3*T(k);
	$\frac{\partial K_s}{\partial T} = b_2 + 2b_3 T(k)$	dKsdT(k)=b2+2*b3*T(k);
	$\frac{\partial Y_{xs}}{\partial T} = c_2 + 2c_3 T(k)$	dYxsdT(k)=c2+2*c3*T(k);
	$\frac{\partial Y_{px}}{\partial T} = d_2 + 2d_3 T(k)$	dYpxdT(k)=d2+2*d3*T(k);
Error	$e_{\lambda x}(k) = \partial x(k) - x(k+1)$	elx(k) = dx(k)-x(k+1)
	$e_{\lambda s}(k) = \partial s(k) - s(k+1)$	els(k) = ds(k)-s(k+1)
	$e_{\lambda p}(k) = \partial p(k) - p(k+1)$	elp(k) = dp(k)-p(k+1)
	$e_{\lambda v}(k) = \partial v(k) - v(k+1)$	elv(k) = dv(k)-v(k+1)
	$e_{\partial x}(k) = -\lambda_x - \mu_x(k) e_{\lambda x}{}^{j}(k)$	edx(k)=-lx-mx*elx(k)
	$e_{\partial s}(k) = -\lambda_x - \mu_s(k) e_{\lambda s}{}^{j}(k)$	eds(k)=-lx-ms*els(k)
	$e_{\partial p}(k) = -\lambda_x - \mu_p(k) e_{\lambda p}{}^{j}(k)$	edp(k)=-lx-mp*elp(k)
	$e_{\partial v}(k) = -\lambda_x - \mu_v(k) e_{\lambda v}{}^{j}(k)$	edv(k)=-lx-mv*elv(k)
Formulas	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n = \varphi_f(k)$	ff(k)=dt*mmax(k)*((1-pk/pl)^0.52)
	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s + s(k)} = \varphi_x(k)$	fx(k)=ff(k)*s(k)/(Ks(k) + s(k))
	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{\left[K_s + s(k)\right]^2} = \varphi_s(k)$	fs(k)=ff(k)*Ks(k)* x(k)/((Ks(k) + s(k))^2)

	$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p*}\right]^{n-1} n\frac{-1}{p^*}\frac{x(k)s(k)}{K_s+s(k)}=\varphi_p(k)$	fp(k)=dt*mmax(k)*((1-p(k)/pl)^(n-1))*x(k)*s(k)/(pl*(Ks(k)+s(k)))
	$\Delta t\frac{F(k)}{V^2(k)}=\varphi_v(k)$	fv(k)=dt*F(k)/(v(k)^2)
	$\Delta t\left[1-\frac{p(k)}{p*}\right]^{n}\frac{x(k)s(k)}{K_s+s(k)}=\varphi_1(k)$	f1(k)=ff(k)*x(k)*s(k)/(Ks(k) + s(k))
	$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p*}\right]^{n}\frac{x(k)s(k)}{\left[K_s+s(k)\right]^2}=\varphi_2(k)$	f2(k)=f1(k)/((Ks(k)+ s(k)))^2

5.6 Description of the Matlab sequential program for optimal control of fed-batch fermentation process

The complex relationship between physiochemical and biological variables and sudden unexplained changes in the process are reflected in the values of the equation kinetic parameters equations (2.72)-(2.75) which are the same as from section 4.6. Table 5.2 defines process kinetic values that are calculated from Matlab by using *polyfit function* from a certain given values of the equation kinetic parameters.

Program *fedbatch_seque.m* is developed to implement the decomposition method in a sequential way. The program calculates the optimal values of the state and control variables according to the algorithm 5.4. The whole program is in appendix B. The function *modelfedbatch_func2* is used to calculate the trajectories of the derivatives of the state space vector, according to the Equations (2.84 – 2.86). Then the program is called by the solver *ode15s* for integration of the model equations. The input variables are *t* the time at moment *t* from the considered time interval, *y* initial, or value of the state vector in the moment *t*. The output variables are the values of the state space vector derivatives *ydot*. The program is given in appendix B.

Table 5.2: Model parameter values for fed-batch process

Symbol	Name	Value	Unit
a1	Model co-efficient	0.0004	
a2	Model co-efficient	0.0001	
a3	Model co-efficient	0.0002	
b1	Model co-efficient	0.2	
b2	Model co-efficient	0.03	
b3	Model co-efficient	0.01	
c1	Model co-efficient	2	
c2	Model co-efficient	0.3	
c3	Model co-efficient	0.021	
d1	Model co-efficient	0.0360	
d2	Model co-efficient	0.008	
d3	Model co-efficient	0.003	
Kd	Death constant	0.00008	
pl	Inhibition factor p*.	50	
n	Powerc-oefficient	0.52	
m	Maintenance-coefficient.	0.0011	
x	Biomass concentration		g/l
s	Substrate concentration		g/l
p	Product concentration		g/l
dt	Sampling period	0.25	
u_{max}	Maximum growth rate		
K_s	Monod's constant		
Y_{xs}	Stochiometry coefficient for biomass growth		

5.6.1 Results from sequential calculation of fed-batch fermentation process

The program for sequential calculation is used to calculate the optimal trajectories of the control and state variables of the fed-batch fermentation process. The optimal trajectories are presented in figure 5.2 -5.9

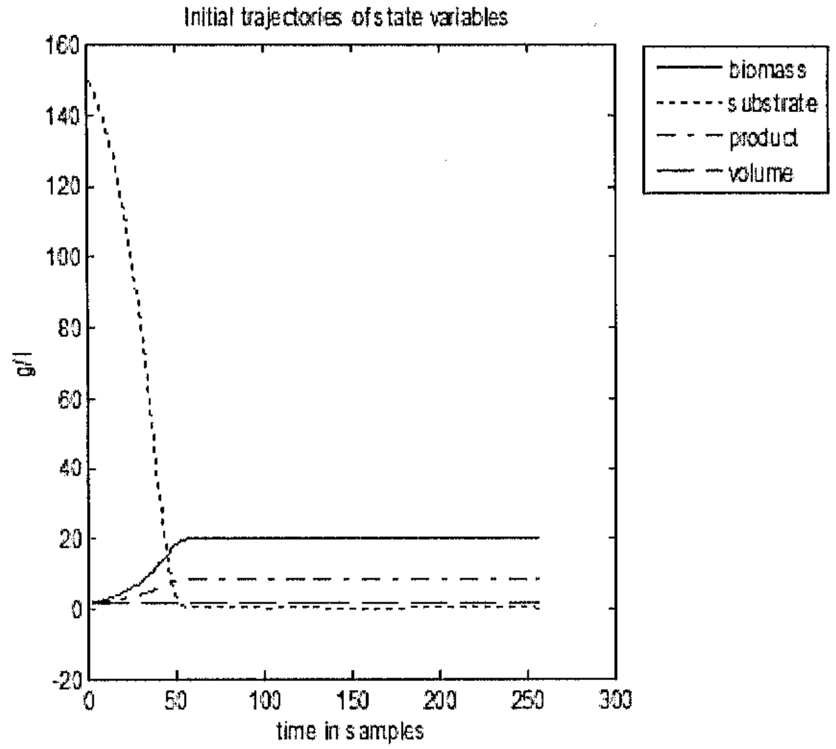


Figure 5.2: Initial trajectories of state variables for fed-batch fermentation process

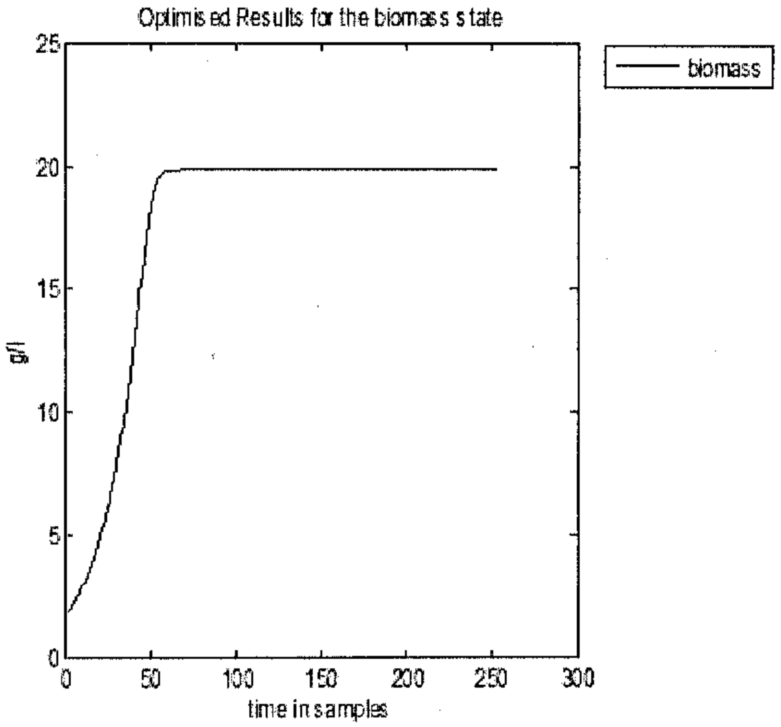


Figure 5.3: Optimal trajectory of biomass for fed-batch fermentation process

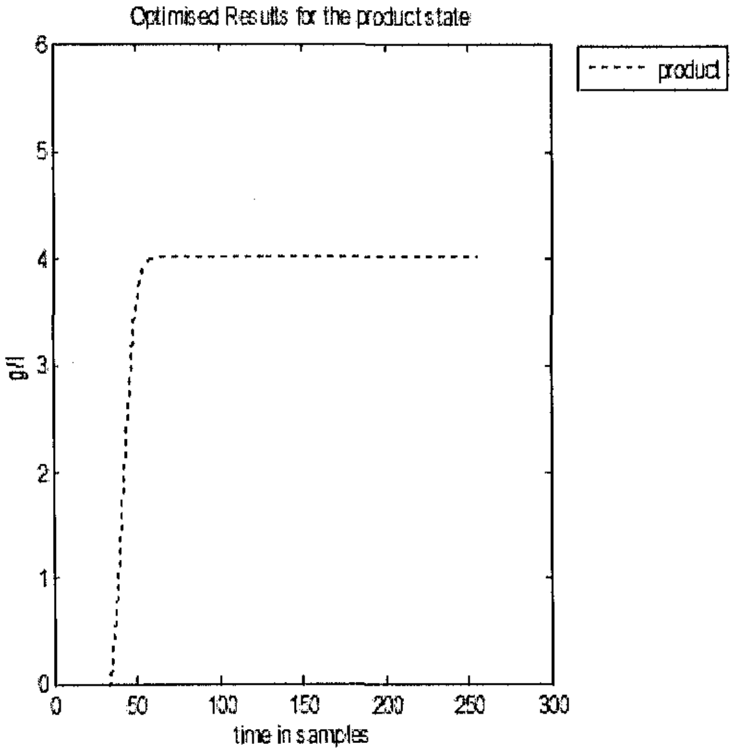


Figure 5.4: Optimal trajectory of product for fed-batch fermentation process

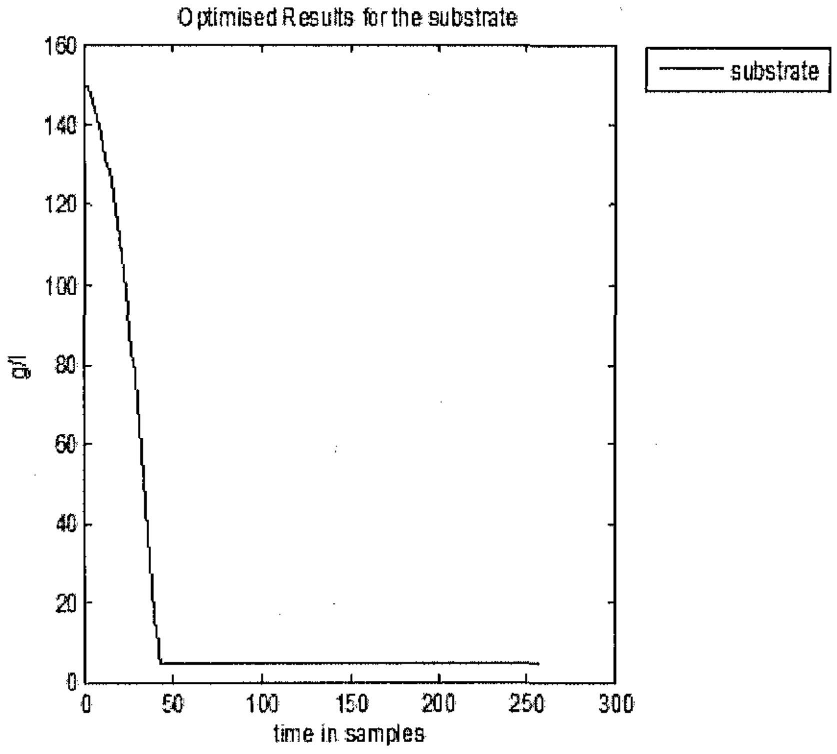


Figure 5.5: Optimal trajectory of substrate for fed-batch fermentation process

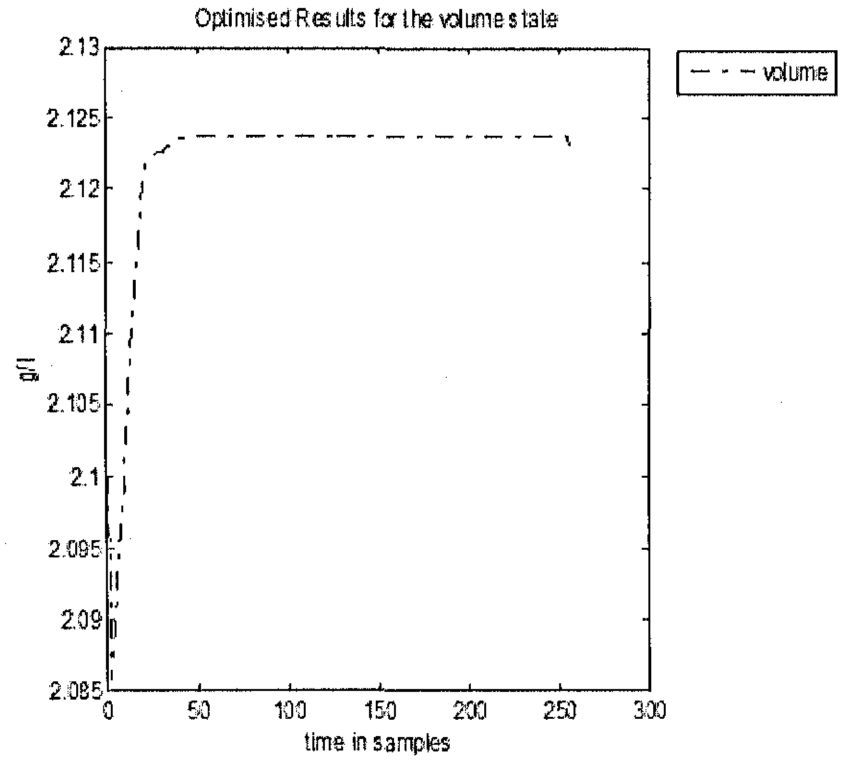


Figure 5.6: Optimal trajectory of volume for fed-batch fermentation process

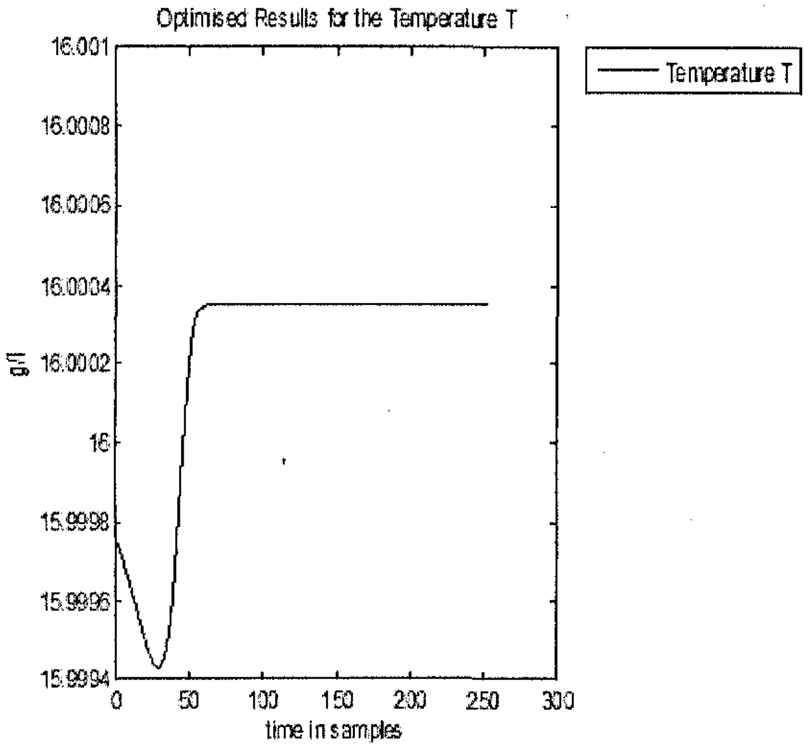


Figure 5.7: Optimal trajectory of temperature for fed-batch fermentation process

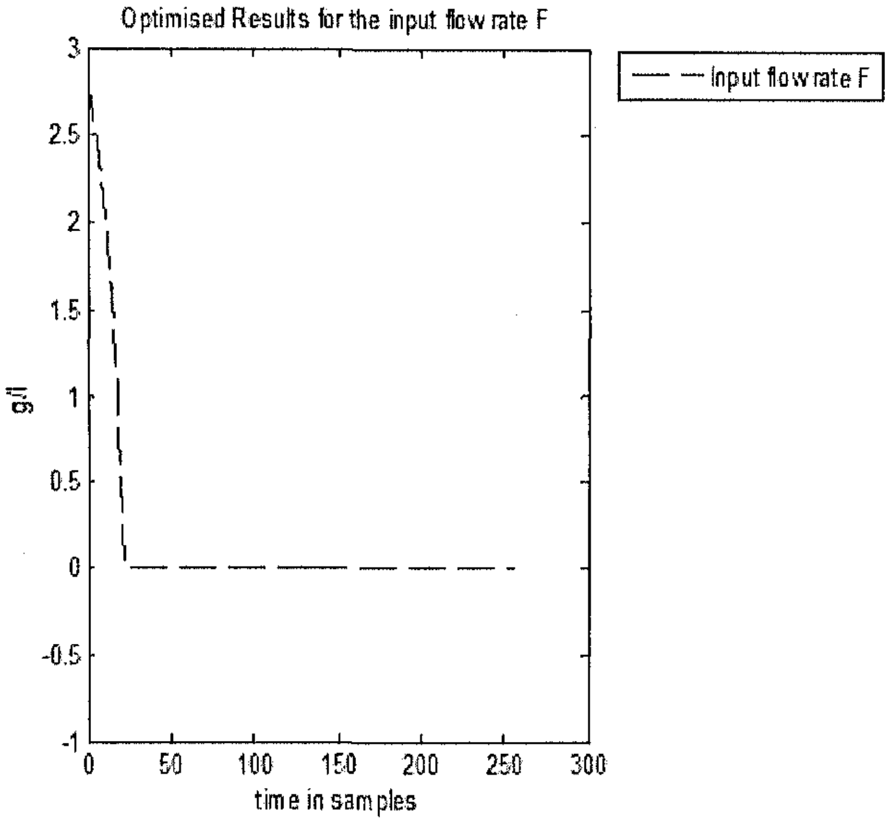


Figure 5.8: Optimal trajectory of input-flow rate for fed-batch fermentation process

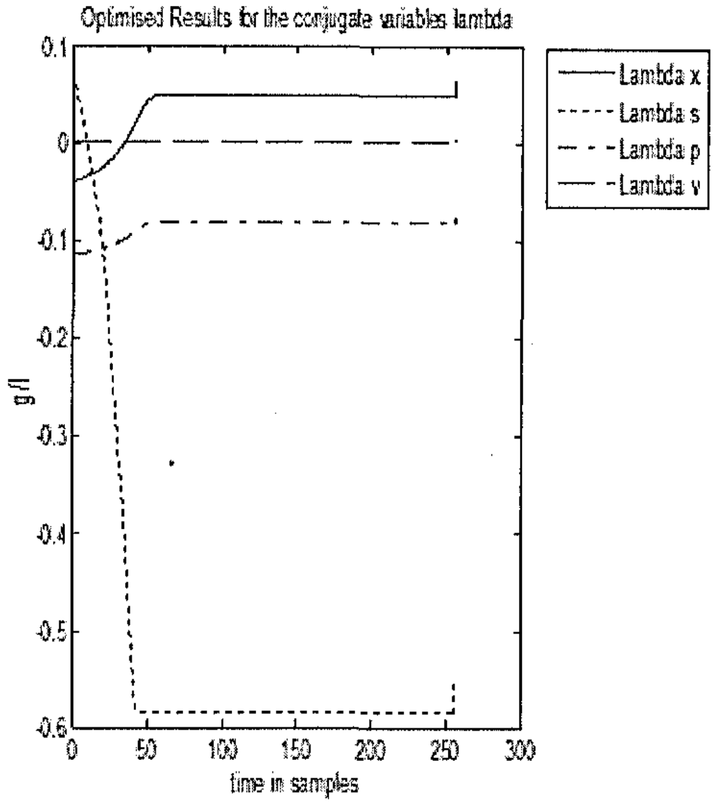


Figure 5.9: Optimal trajectory of conjugate variables for fed-batch fermentation process

5.7 Discussion of the results for fed-batch fermentation process

The biomass concentration generally continued to rise after feed was started, as at that stage growth rate exceeded dilution rate, but within few hours after increases in the feed rate, it invariably began to fall. Substarte levels generally rose slowly throughout the feed period as desired although it is fairly sensitive to feed rate. Product concentration on the other hand rises sharply for several hours after the feed was switched on and then reached a plateau, rising slowly after the feed was exhausted.

5.8 Conclusion

This chapter describes the formulation of the problem for optimal control of a fed-batch fermentation process. An augmented Lagrange functional is used and a decomposition method for solving discrete time formulation of the problem is developed based on varational calculus and decomposition in time domain. The sequential algorithm of the decomposition method is describes and implemented in Matlab.The results are shown and discussed. The parallel implementation of the decomposition method is described in chapter eight. The problem for optimal control of continuous fermentation process is considerd in chapter six.

CHAPTER SIX
DEVELOPMENT OF A DECOMPOSITION METHOD FOR OPTIMAL CONTROL PROBLEM CALCULATION FOR CONTINOUS FERMENTATION PROCESS

6.1 Introduction

The continuous process has some advantages when compared to the fed-batch and batch mode. If steady state is attained, quality of the final product is constant, as the operational conditions are invariable. Besides, the volume of the fermentor can typically be smaller, as shut-downs for harvesting and sterilization are less frequent. However, due to the high probability of contamination and culture degeneration caused by a long operational time, continuous processes are used only in few cases in the fermentation industry, such as cell mass and ethanol production.

Before a continuous fermentor reaches stable state operation, there is a transient period in which it is filled up to the working volume. This is called the start-up of the fermentor and refers to the progression of operations that leads it from an original state to the final state of continuous operation. It is important to determine the efficient start-up strategy of the fermentor as it saves time. Bio-chemical reactions are typically slower than chemical reactions and an inappropriate start-up may cause the fermentor to spend time to reach the steady state conditions.

The problem for minimum time control of continuous fermentation process is formulated in section 6.2 on the basis of a decomposition method. This method uses an augmented Lagrange functional in order to overcome the difficulties with the singular type of control and no quadratic criterion is developed in section 6.3. A coordination vector, constructed to achieve separability of the dual problem is developed. The problem for minimum time can be decomposed in time domain and be solved in a two level computing structure. A computer algorithm is developed to solve this problem in section 6.4.

6.2 Formulation of the problem for optimal control of continuous fermentation process

The problem for minimizing start-up time of continuous fermentation process is considered (Tzoneva and Patarinska, 1995). The problem for minimum time is a dynamic one and the used model is dynamic (Tsoneva and Popchev, 1995). The aim is to minimize the time necessary for the process to reach steady state values of concentration of biomass, substrate and product. Mathematically, discrete minimum time control problem for the continuous fermentation process is to find a control trajectory $D(k), k = \overline{0, K-1}$ which in minimum time

$$J = K\Delta t \tag{6.1}$$

leads the system presented by equations (2.96)- (2.98) and (2.72)-(2.75) from initial state $x(0) = x_0, s(0) = s_0, p(0) = p_0$ to the optimal steady state $x(K) = \bar{x}$, $s(K) = \bar{s}, p(K) = \bar{p}$ while satisfying constraints for state and control variables

$$x_{min}(k) \leq x(k) \leq x_{max}(k), k = \overline{0, K} \tag{6.2}$$

$$s_{min}(k) \leq s(k) \leq s_{max}(k), k = \overline{0, K} \tag{6.3}$$

$$p_{min}(k) \leq p(k) \leq p_{max}(k),\ k = \overline{0, K} \tag{6.4}$$

$$D_{min}(k) \leq D(k) \leq D_{max}(k), k \doteq \overline{0, K-1} \tag{6.5}$$

and constraints for the sampling interval

$$0 \leq \Delta t \leq \Delta t_{max} \tag{6.6}$$

Where Δt_{max} is determined on the basis of stability requirements for the fermentation process

6.3 Decomposition method for optimal control problem solution for continuos fermentation process

The considered problem for optimal control has the following characteristics

- The objective is to minimize a function of the final time with a free final time starting from known initial conditions.
- For different initial conditions, different optimum for the minimum time values will be obtained.
- In the start-up of a continuous fermentation process the known conditions for the state equations are the final conditions.
- The stated problem is characterized by nonquadratic criterion and bilinear model with the control entering linearly the model equations

These characteristics determine which methodology to be selected for solution of the optimal control problem. The minimum time problem is transformed into a problem for minimizing the value of the sampling interval in the limits of (6.6). As the sampling period is easier to be included in the model equations and to be minimized, the number of sampling steps will then be fixed. The sampling period also enters linearly into the model equations. This means that the solution of the optimal control problem can be singular because the first derivative of the functional of Lagrange towards the control variable and the sampling period will not include these variables. To overcome these difficulties an augmented functional of Lagrange proposed in (Tzoneva and Patarinska, 1995) is used to solve the problem.

$$L_a = K\Delta t + \sum_{v=x,s,p}\left\{\lambda_v(K)[v(K) - \bar{v}] + \tfrac{1}{2}\mu_v[v(K) - \bar{v}]^2\right\} + \sum_{k=0}^{K-1}\sum_{v=x,s,p}\left\{\lambda_v(k)[-v(k+1) + +f_v(k)] + \tfrac{1}{2}\mu_v[-v(k+1) + f_v(k)]^2\right\} \tag{6.7}$$

where $\bar{v} = \bar{x}, \bar{s}, \bar{p}$

6.3.1 Necessary conditions for optimality for continuous fermentation process

Solution of the problem for optimal control is based on the necessary conditions for optimality of the functional of Lagrange (6.7). The equations of these conditions are used to build the algorithm for calculation of the optimal state, control variables and sampling period. This algorithm is used for parallel calculation in a cluster of computers.

- Necessary conditions for optimality according to biomass

$$\frac{\partial L}{\partial x(k)} = \lambda_x(k)\left[1 + \Delta t\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s + s(k)}\right] - \Delta t k_d - \Delta t D(k)\right] -$$

$$-\mu_x(k)[x(k+1) - f_x(k)]\left[1 + \Delta t\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s + s(k)}\right] - \Delta t k_d - \Delta t D(k)\right] +$$

$$+\lambda_s(k)\left[\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s + s(k)}\right] - \Delta t m\right] -$$

$$-\mu_s(k)[s(k+1) - f_s(k)]\left[\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1 - \frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s + s(k)}\right] - \Delta t m\right] +$$

$$+\lambda_p(k)\left[\Delta t Y_{PX}\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s + s(k)}\right]\right] -$$

$$-\mu_p(k)[p(k+1) - f_p(k)]\left[\Delta t Y_{PX}\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s + s(k)}\right]\right] = 0 = e_x(k), k = \overline{0, K-1}$$

$$\frac{\partial L}{\partial x(K)} = \lambda_x(K) + \mu_x\left[x(K) - \bar{x}\right] = e_x(K) = 0 \tag{6.8}$$

- Necessary conditions for optimality according to substrate

$$\frac{\partial L}{\partial s(k)} = \lambda_x(k)\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)(K_s+s(k))-x(k)s(k)}{[K_s+s(k)]^2}\right]\right]-$$
$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]+$$
$$+\lambda_s(k)\left[1-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]-\Delta tD(k)\right]-$$
$$-\mu_s(k)[s(k+1)-f_s(k)]\left[1-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]-\Delta tD(k)\right]$$
$$+\lambda_p(k)\left[\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]-$$
$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)K_s}{[K_s+s(k)]^2}\right]\right]=0=e_s(k), k=\overline{0,K-1}$$
$$\frac{\partial L}{\partial s(K)} = \lambda_s(K)+\mu_s\left[s(K)-\bar{s}\right]=e_s(K)=0 \tag{6.9}$$

- Necessary conditions for optimality according to product

$$\frac{\partial L}{\partial p(k)} = \lambda_x(k)\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1}n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]+$$
$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1}n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]+$$
$$+\lambda_s(k)\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^{n-1}n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]\right]+$$

$$+\lambda_p(k)\left[1+\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]-\Delta tD(k)\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[1+\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]-\Delta tD(k)\right]$$

$$=0=e_p(k),\ k=\overline{0,K-1}$$

$$\frac{\partial L}{\partial p(K)}=\lambda_p(K)+\mu_p\left[p(K)-\overline{p}\right]=e_p(K)=0$$

(6.10)

- Necessary conditions for optimality of the Lagrange functional(6.7) according to temperature

$$\frac{\partial L}{\partial T(k)}=\lambda_x(k)\left[\frac{\Delta t\partial\mu_{\max}}{\partial T}\left[1-\frac{p(k)}{p^*}\right]^{n}\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n}\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_x(k)[x(k+1)-f_x(k)]$$

$$\left[\frac{\Delta t\partial\mu_{\max}}{\partial T}\left[1-\frac{p(k)}{p^*}\right]^{n}\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n}\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]+$$

$$+\lambda_s(k)\left[-\Delta t\frac{\frac{\partial\mu_{\max}}{\partial T}Y_{XS}-\mu_{\max}\frac{\partial Y_{XS}}{\partial T}}{Y^2{}_{XS}}\left[1-\frac{p(k)}{p^*}\right]^{n}\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$\left.-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^{n}\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[-\Delta t\frac{\frac{\partial\mu_{\max}}{\partial T}Y_{XS}-\mu_{\max}\frac{\partial Y_{XS}}{\partial T}}{Y^2{}_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$\left.-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]+$$

$$+\lambda_p(k)\left[\Delta t\left[\frac{\partial Y_{PX}}{\partial T}\mu_{\max}+Y_{PX}\frac{\partial\mu_{\max}}{\partial T}\right]\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\right.$$

$$\left.+\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta t\left[\frac{\partial Y_{PX}}{\partial T}\mu_{\max}+Y_{PX}\frac{\partial\mu_{\max}}{\partial T}\right]\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\right. \tag{6.11}$$

$$\left.+\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]=0=e_T(k),k=\overline{0,K-1}$$

Where the partial derivatives of the kinetic parameters are derived in equations (5.10)

- Necessary conditions for optimality of the augmented Lagrange functional(6.7) according to dilution rate

$$\frac{\partial L_a}{\partial D(k)}=\lambda_x(k)[-\Delta tx(k)]+\mu_x[-x(k+1)+f_x(k)][-\Delta tx(k)]+\lambda_s(k)\left[\Delta t\left(s^0-s(k)\right)\right]+$$
$$\mu_s[-s(k+1)+f_s(k)]\Delta t\left[\left(s^0-s(k)\right)\right]+\lambda_p(k)\left[-\Delta t\left(p(k)\right)\right]+$$
$$\mu_p\left[-p(k+1)+f_p(k)\right](-\Delta t(p(k))=e_D(k)=0,k=\overline{0,K-1} \tag{6.12}$$

- Necessary conditions for optimality of the augmented Lagrange functional (6.7) according to the conjugate variables

$$\frac{\partial L}{\partial\lambda_x(k)}=-x(k+1)+f_x(k)=0=e_{\lambda x}(k),k=\overline{0,K-1} \tag{6.13}$$

$$\frac{\partial L}{\partial \lambda_x(K)} = [x(K) - \bar{x}] = e_{\lambda x}(K), \tag{6.14}$$

$$\frac{\partial L}{\partial \lambda_s(k)} = -s(k+1) + f_s(k) = 0 = e_{\lambda s}(k), k = \overline{0, K-1} \tag{6.15}$$

$$\frac{\partial L}{\partial \lambda_s(K)} = [s(K) - \bar{s}] = e_{\lambda s}(K) \tag{6.16}$$

$$\frac{\partial L}{\partial \lambda_p(k)} = -p(k+1) + f_p(k) = 0 = e_{\lambda p}(k), k = \overline{0, K-1} \tag{6.17}$$

$$\frac{\partial L}{\partial \lambda_p(K)} = [p(K) - \bar{p}] = e_{\lambda p}(K) \tag{6.18}$$

- Necessary conditions for optimality of the augmented Lagrange functional according to the sampling interval Δt.

The sampling period are only calculated for the last period $k = K$ then $x(k) = x(K), s(k) = s(K)$ and $p(k) = p(K)$ and the augmented Lagrangian function is:

$$L_a = K\Delta t + \lambda_x(K)[x(K) - \bar{x}] + \frac{1}{2}\mu_x[x(K) - \bar{x}]^2 + \lambda_s(K)[s(K) - \bar{s}] + \frac{1}{2}\mu_s[s(K) - \bar{s}]^2 + \lambda_p(K)[p(K) - \bar{p}] + \frac{1}{2}\mu_p[p(K) - \bar{p}]^2 \tag{6.19}$$

Since $x(K), s(K)$ and $p(K)$ are all function of Δt, they can all be expressed by the following models

$$x(K) = 1 + \Delta t\mu_{max}\left[1 - \frac{p(K-1)}{p^*}\right]^n \left[\frac{x(K-1)s(K-1)}{K_s + s(K-1)}\right] - \Delta t k_d x(K-1) + \Delta t D(K-1)x(K-1) \tag{6.20}$$

$$s(K) = 1 - \frac{\Delta t\mu_{max}}{Y_{XS}}\left[1 - \frac{p(K-1)}{p^*}\right]^n \left[\frac{x(K-1)s(K-1)}{K_s + s(K-1)}\right] - \Delta t m x(K-1) + \Delta t D(K-1)[s^0(k) - \\ -s(K-1)] \tag{6.21}$$

$$p(K) = 1 + \Delta t\mu_{max} Y_{PX}\left[1 - \frac{p(K-1)}{p^*}\right]^n \left[\frac{x(K-1)s(K-1)}{K_s + s(K-1)}\right] - \Delta t D(K-1)p(K-1) \tag{6.22}$$

If the above equations (6.20), (6.21) and (6.22) are substituted into the augmented Lagrange functional of equation (6.19), then the necessary condition for optimality according to the sampling interval is:

$\frac{\partial L_a(K)}{\partial \Delta t} = K + \frac{\partial x(K)}{\partial \Delta t}\lambda_x(K) + \frac{\partial x(K)}{\partial \Delta t}\mu_x[x(K) - \bar{x}] + \frac{\partial s(K)}{\partial \Delta t}\lambda_s(K) + \frac{\partial s(K)}{\partial \Delta t}\mu_s[s(K) - -\bar{s}] +$
$\frac{\partial p(K)}{\partial \Delta t}\lambda_p(K) + \frac{\partial p(K)}{\partial \Delta t}\mu_p[p(K) - \bar{p}] = e_{\Delta t}(K) = 0$

Therefore

$K + \frac{\partial x(K)}{\partial \Delta t}[\lambda_x(K) + \mu_x[x(K) - \bar{x}]] + \frac{\partial s(K)}{\partial \Delta t}[\lambda_s(K) + \mu_s[s(K) - \bar{s}]] + + \frac{\partial p(K)}{\partial \Delta t}\big[\lambda_p(K) +$
$\mu_p[p(K) - \bar{p}]\big] = e_{\Delta t}(K) = 0$ (6.23)

The necessary conditions for optimality of the sampling interval are dependent on biomass, substrate and product. These variables depend on the sampling interval on control values at the moment K-1. In order to find explicit dependence of the necessary conditions for optimality (6.23) on Δt all state variables at the moment K are represented by their corresponding equations at the moment K-1. Then the derivatives $\frac{\partial v}{\partial \Delta t}, v = x, s, p$ are calculated. These calculations are done separately for every state variable and then the results are substituted in (6.23) as follows. Let

$\mu_{max}\left[1 - \frac{p(K-1)}{p^*}\right]^n \left[\frac{x(K-1)s(K-1)}{K_s + s(K-1)}\right] = \varphi_3(K-1)$ (6.24)

Since it is common in every equation.

For the biomass.

$\frac{\partial x(K)}{\partial \Delta t}[\lambda_x(K) + \mu_x[x(K) - \bar{x}]] = [\varphi_3(K-1) - k_d x(K-1) - D(K--1)x(K-$
$1)][\lambda_x(K) + \mu_x[x(K-1) + \Delta t\varphi_3(K-1) - \Delta t k_d x(K-1) - -\Delta t D(K-1)x(K-1) - \bar{x}]] =$
$\mu_x \Delta t[[\varphi_3(K-1) - k_d x(K-1) - D(K-1)x(K--1)]]^2 + [\varphi_3(K-1) - k_d x(K-1) -$
$D(K-1)x(K-1)][\lambda_x(K) + \mu_x[x(K-1) - -\bar{x}]]$ (6.25)

For the substrate

$\frac{\partial s(K)}{\partial \Delta t}[\lambda_s(K) + \mu_s[s(K) - \bar{s}]] = \Big[-\frac{\varphi_3(K-1)}{Y_{XS}} - mx(K-1) + D(K-1)[s^0 - s(K-$
$-1)]\Big]\Big[\lambda_s(K) + \mu_s\Big[s(K-1) - \Delta t\frac{\varphi_3(K-1)}{Y_{XS}} - \Delta t m x(K-1) + \Delta t D(K--1)[s^0 - s(K-1)] -$
$\bar{s}\Big]\Big] = \mu_s \Delta t\left[\frac{\varphi_3(K-1)}{Y_{XS}} + mx(K-1) - D(K-1)[s^0 - s(K--1)]\right]^2 - \Big[\frac{\varphi_3(K-1)}{Y_{XS}} + mx(K-1) -$

$$D(K-1)[s^0 - s(K-1)]\Big]\left[\lambda_s(K) + +\mu_s[s(K-1)-\bar{s}]\right] \tag{6.26}$$

For the product

$$\frac{\partial p(K)}{\partial \Delta t}\left[\lambda_p(K) + \mu_p[p(K)-\bar{p}]\right] = [\varphi_3(K-1)Y_{PX} - D(K-1)p(K-1)]\left[\mu_p[p(K--1) + \Delta t\varphi_3(K-1)Y_{PX} - \Delta t D(K-1)[p(K-1)-\bar{p}]]\right] = \mu_p\Delta t\left[[\varphi_3(K-1)Y_{PX} - -D(K-1)p(K-1)]\right]^2 + [\varphi_3(K-1)Y_{PX} - D(K-1)p(K-1]\left[\lambda_p(K) + +\mu_p[p(K-1)-\bar{p}]\right] \tag{6.27}$$

The equations (6.25), (6.26) and (6.27) are combined to form the necessary condition for optimality for calculation of Δt.as a coordinating variable. The equation (6.23) becomes:

$$K + \mu_x\Delta t\left[[\varphi_3(K-1) - k_d x(K-1) - D(K-1)x(K-1)]\right]^2 + \\ +[\varphi_3(K-1) - k_d x(K-1) - D(K-1)x(K-1)]\left[\lambda_x(K) + \mu_x[x(K-1)-\bar{x}]\right] + \\ +\mu_s\Delta t\left[\frac{\varphi_3(K-1)}{Y_{XS}} + mx(K-1) - D(K-1)[s^0 - s(K-1)]\right]^2 - \left[\frac{\varphi_3(K-1)}{Y_{XS}} + + mx(K-1) - \\ D(K-1)[s^0 - s(K-1)]\right]\left[\lambda_s(K) + \mu_s[s(K-1)-\bar{s}]\right] + +\mu_p\Delta t\left[[\varphi_3(K-1)Y_{PX} - \\ D(K-1)p(K-1)]\right]^2 + [\varphi_3(K-1)Y_{PX} - D(K--1)p(K-1]\left[\lambda_p(K) + \mu_p[p(K-1)-\bar{p}]\right] = \\ = 0 \tag{6.28}$$

6.3.2 Coordinating vector for continuous fermentation process

Equations (6.8)-(6.12) and (6.28) determine the necessary conditions for optimality of the minimum time control problem. It can be seen that the number of equations and variables is big. The solution could be found easier if the problem is decomposed. Decomposition in time domain could reduce complexity of the system equations (Tamura, 1975) and (Singandlitty, 1982).

The solutions in time domain will consists of K+1 separate solutions, everyone at separate time moment k. The decomposition in time domain could be obtained on the basis of the coordinating procedure in two level computing structure(given in Figure 6.2) using the conjugate variables λ as coordinating ones(Tamura, 1975), but this method is applicable for normal functional of Lagrange. The used augmented functional could not be fully decomposed because of the quadratic terms and variables $x(k+1), s(k+1)$ and $p(k+1)$in them. They play a role of interconnections in time domain (Lin, 1992). The unkown sampling interval Δt also prevents the full decomposition. If its value is known by selecting it to be a coordinating variable, the full decomposition can be obtained. This type of functional can be fully decomposed if the considered variables are selected also as a part of the coordinating vector. To overcome these difficulties it is proposed to consider

$$\begin{aligned}&\delta^{j}{}_{v}(k) = v(k+1), v = x, s, p, k = \overline{0, K}\\&\Delta^{j} t = \Delta t\end{aligned} \tag{6.29}$$

Where j is the index of the coordinating procedure. Then the values of the coordinating variables are set from the second level of the two level calculating structure:

$$\lambda_{v}(k) = \lambda_{v}{}^{j}(k), v = x, s, p, k = \overline{0, K},\ \Delta t = \Delta t^{j} \tag{6.30}$$

$$\delta_{v}(k) = \delta_{v}{}^{j}(k), v = x, s, p, k = \overline{0, K} \tag{6.31}$$

where j is the index of the coordinating process iterations. When the values of the coordinating variables from equations (6.30)–(6.31) are substituted into the Lagrange's functional, its full decomposition according to the discrete time moments k is obtained. The functional is decomposed into K+1 sub-functionals $L_a(k)$ and each of them determines the optimal control and state at the given moment k. The above algorithm is implemented in Matlab software environment in two ways:

- Sequentially in one computer
- Parallel, using the cluster of computers

The decomposition in time domain is based on a two level calculation structure, given in Figure 6.1

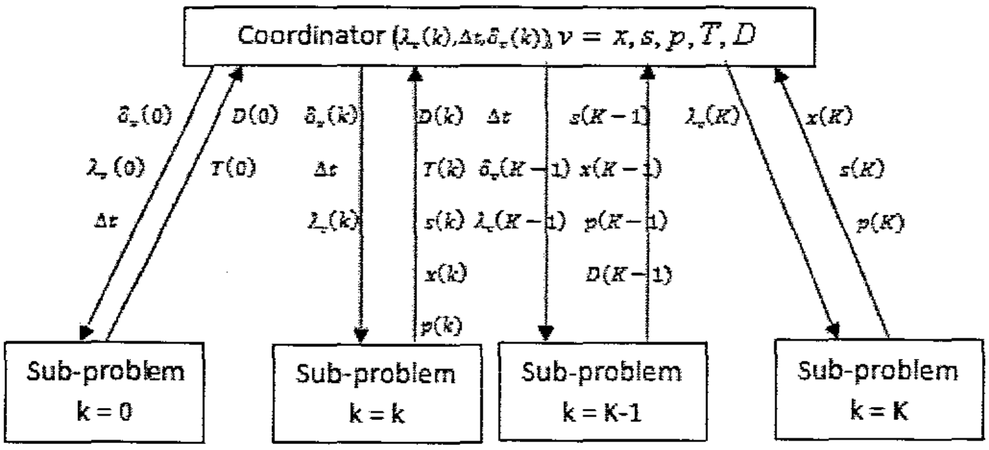


Figure 6.1: Two Level computing structure for minimum startup time

6.3.3 First level sub-problems for continuos fementation process

The solutions of the first level sub-problems represent solution of the set of equations (6.8) – (6.12) and (6.28) in which the coordinating variables are substituted. These equations are solved for every separate moment of time in a parallel way. The structure of the equations (6.8) – (6.12) and (6.28) does not allow to use analytical solution. Gradient method is used in the form:

$$v^{j,i+1}(k) = v^{j,i}(k) - \alpha_v e_v^{\,j,i}(k), v = x,s,p, k = \overline{0,K}, v = T,D,x,s,p, k = \overline{0,K-1} \qquad (6.32)$$

where $\alpha_v > 0$ are the steps of the gradient procedures, $e_v(k)$ are the gradients according to the equations (6.8) – (6.12) and (6.23) and *i* is the iteration index of the procedures. The first level sub-problems are determined under the set from the second level coordinating variables according to the necessary conditions for optimality of the sub-functionals $L_a(k)$. The equations (6.8) – (6.12) and (6.23) can be simplified further by using necessary conditions for optimality according to the conjugate variables presented by using (6.13)–(6.18) in the form:

For the biomass

$$\frac{\partial L}{\partial x(k)} = e_x^{j,i}(k) = \left[1+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s+s(k)} - \Delta tk_d - \Delta tD(k)\right]\left[\lambda_x^{\ j}(k)-\mu_x e_{\lambda x}^{\ j}(k)\right]+$$
$$+\left[\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s+s(k)} - \Delta tm\right]\left[\lambda_s^{\ j}(k)-\mu_s e_{\lambda s}^{\ j}(k)\right]+$$
$$+\left[\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s+s(k)}\right]\left[\lambda_p^{\ j}(k)-\mu_p e_{\lambda p}^{\ j}(k)\right], k=\overline{0,K-1}$$

(6.33)

For the substrate

$$\frac{\partial L}{\partial s(k)} = e_s^{j,i}(k) = \left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)K_s}{[K_s+s(k)]^2}\right]\left[\lambda_x^{\ j}(k)-\mu_x e_{\lambda x}^{\ j}(k)\right]+$$
$$+\left[1-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^n \frac{s(k)K_s}{[K_s+s(k)]^2} - \Delta tD(k)\right]\left[\lambda_s^{\ j}(k)-\mu_s e_{\lambda s}^{\ j}(k)\right]+ \qquad (6.34)$$
$$+\left[\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)K_s}{[K_s+s(k)]^2}\right]\left[\lambda_p^{\ j}(k)-\mu_p e_{\lambda p}^{\ j}(k)\right], k=\overline{0,K-1}$$

For the product

$$\frac{\partial L}{\partial p(k)} = e_p^{j,i}(k) = \left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[\frac{-1}{p^*}\right]\frac{x(k)s(k)}{K_s+s(k)}\right]\left[\lambda_x^{\ j}(k)-\mu_x e_{\lambda x}^{\ j}(k)\right]$$
$$\left[-\frac{\Delta t\mu_{\max}}{Y_{XS}}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[\frac{-1}{p^*}\right]\frac{x(k)s(k)}{K_s+s(k)}\right]\left[\lambda_s^{\ j}(k)-\mu_s e_{\lambda s}^{\ j}(k)\right]+$$
$$+\left[1+\Delta tY_{PX}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1} n\left[\frac{-1}{p^*}\right]\frac{x(k)s(k)}{K_s+s(k)} - \Delta tD(k)\right]\left[\lambda_p^{\ j}(k)-\mu_p e_{\lambda p}^{\ j}(k)\right], k=\overline{0,K-1}$$

(6.35)

For the temperature

$$\frac{\partial L}{\partial T(k)} = e_T^{j,i}(k) =$$

$$= \left[\frac{\Delta t \partial \mu_{\max}}{\partial T} \left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} - \Delta t \mu_{\max} \left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2} \right]\right] *$$

$$* \left[\lambda_x^{\,j}(k) - \mu_x(k) e_{\lambda x}^{\,j}(k)\right] +$$

$$+ \left[-\Delta t \frac{\frac{\partial \mu_{\max}}{\partial T} Y_{XS} - \mu_{\max} \frac{\partial Y_{XS}}{\partial T}}{Y^2_{XS}} \left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} + \right.$$

$$\left. + \frac{\Delta t \mu_{\max}}{Y_{XS}} \left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2} \right]\right] \left[\lambda_s^{\,j}(k) - \mu_s(k) e_{\lambda s}^{\,j}(k)\right] +$$

$$+ \left[\Delta t \frac{\partial \mu_{\max}}{\partial T} Y_{PX} + \mu_{\max} \frac{\partial Y_{PX}}{\partial T} \left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} - \right.$$

$$\left. - \Delta t \mu_{\max} Y_{PX} \left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2} \right]\right] \left[\lambda_p^{\,j}(k) - \mu_p(k) e_{\lambda p}^{\,j}(k)\right], k = \overline{0, K-1} \qquad (6.36)$$

For the dilution rate

$$\frac{\partial L}{\partial D} = e_D^{\,j,i} =$$

$$-[\Delta t x(k)] \left[\lambda_x^{\,j}(k) - \mu_x e_{\lambda x}^{\,j}(k)\right] + [\Delta t(s_i - s(k))] \left[\lambda_s^{\,j}(k) - \mu_s e_{\lambda s}^{\,j}(k)\right] - [\Delta t p(k)]*$$

$$\left[\lambda_p^{\,j}(k) - \mu_p(k) e_{\lambda p}^{\,j}(k)\right], k = \overline{0, K-1} \qquad (6.37)$$

These equations can be simplified further for the purpose of calculations. Some expressions in them are repeated, that is why the common notations are used to represent them:

$$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\frac{s(k)}{K_s+s(k)}=\varphi_x(k) \tag{6.38}$$

$$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1}\frac{-1}{p}n\frac{x(k)s(k)}{K_s+s(k)}=\varphi_p(k) \tag{6.39}$$

$$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)K_s}{[K_s+s(k)]^2}=\varphi_s(k) \tag{6.40}$$

$$\Delta t\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}=\varphi_1(k) \tag{6.41}$$

$$-\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{[K_s+s(k)]^2}=\varphi_2(k), k=\overline{0,K-1} \tag{6.42}$$

Using notations from (6.38) - (6.42) equations from (6.33)-(6.37) can be written in the form:

For the biomass

$$\begin{aligned}\frac{\partial L}{\partial x(k)}&=e_x^{j,i}(k)=\left[1+\varphi_x^{j,i}(k)-\Delta tk_d-\Delta tD(k)\right]\left[\lambda_x^{j}(k)-\mu_x(k)e_{\lambda x}^{j}(k)\right]+\\&+\left[\frac{\varphi_x^{j,i}(k)}{Y_{XS}}-\Delta tm\right]\left[\lambda_s^{j}(k)-\mu_s(k)e_{\lambda s}^{j}(k)\right]+\\&+\varphi_x^{j,i}(k)Y_{PX}\left[\lambda_p^{j}(k)-\mu_p(k)e_{\lambda p}^{j}(k)\right], k=\overline{0,K-1}\end{aligned} \tag{6.43}$$

For the substrate

$$\begin{aligned}\frac{\partial L}{\partial s(k)}&=e_s^{j,i}(k)=\varphi_s^{j,i}(k)\left[\lambda_x^{j}(k)-\mu_x(k)e_{\lambda x}^{j}(k)\right]+\\&+\left[1-\frac{\varphi_s^{j,i}(k)}{Y_{XS}}-\Delta tD^{j,i}\right]\left[\lambda_s^{j}(k)-\mu_s(k)e_{\lambda s}^{j}(k)\right]+\\&+Y_{PX}\varphi^{j,l}{}_s(k)\left[\lambda_p^{j}(k)-\mu_p(k)e_{\lambda p}^{j}(k)\right], k=\overline{0,K-1}\end{aligned} \tag{6.44}$$

For the product

$$\frac{\partial L}{\partial p(k)} = e_p^{j,i}(k) = [\varphi_p(k)][\lambda_x^j(k) - \mu_x e_{\lambda x}^j(k)] +$$
$$+ \left[\frac{\varphi_p(k)}{Y_{XS}}\right][\lambda_s^j(k) - \mu_s e_{\lambda s}^j(k)] + \tag{6.45}$$
$$+ [1 + \varphi_p(k) - \Delta t D(k)][\lambda_p^j(k) - \mu_p(k) e_{\lambda p}^j(k)], k = \overline{0, K-1}$$

For the temperature

$$\frac{\partial L}{\partial T(k)} = e_T^{j,i}(k) =$$
$$= \left[\varphi^{j,i}{}_1(k)\frac{\partial \mu_{\max}}{\partial T} + \varphi_2(k)\frac{\partial K_s}{\partial T}\right][\lambda_x^j(k) - \mu_x(k) e_{\lambda x}^j(k)] +$$
$$+ \left[[-\varphi_1(k)]\left[\frac{\frac{\partial \mu_{\max}}{\partial T} Y_{XS} - \mu_{\max}\frac{\partial Y_{XS}}{\partial T}}{Y^2{}_{XS}}\right] +\right.$$
$$\left.- \frac{\varphi_2(k)}{Y_{XS}}\frac{\partial K_s}{\partial T}\right][\lambda_s^j(k) - \mu_s(k) e_{\lambda s}^j(k)] + \left[\varphi_1(k)\left[\frac{\partial Y_{PX}}{\partial T}\mu_{\max} + Y_{PX}\frac{\partial \mu_{\max}}{\partial T}\right] +\right. \tag{6.46}$$
$$\left.+ Y_{PX}\varphi_2(k)\frac{\partial K_s}{\partial T}\right][\lambda_p^j(k) - \mu_p(k) e_{\lambda p}^j(k)], k = \overline{0, K-1}$$

For the dilution rate

$$\frac{\partial L}{\partial D} = e_D^{j,i} = -[\Delta t x(k)]\left[\lambda_x^j(k) - \mu_x e_{\lambda x}^j(k)\right] + [\Delta t(s_i - s(k))] - [\Delta t p(k)] *$$
$$* \left[\lambda_p^j(k) - \mu_p(k) e_{\lambda p}^j(k)\right], k = \overline{0, K-1} \tag{6.47}$$

For sampling interval

$$K + \mu_x \Delta t[[\varphi_3(K-1) - k_d x(K-1) - D(K-1)x(K-1)]]^2 + [\varphi_3(K--1) - k_d x(K-1) -$$
$$D(K-1)x(K-1)][\lambda_x(K) + \mu_x[x(K-1) - \bar{x}]] + +\mu_s \Delta t\left[\frac{\varphi_3(K-1)}{Y_{XS}} + mx(K-1) -\right.$$
$$\left.D(K-1)[s^0 - s(K-1)]\right]^2 - \left[\frac{\varphi_3(K-1)}{Y_{XS}} + +mx(K-1) - D(K-1)[s^0 - s(K-1)]\right][\lambda_s(K) +$$
$$\mu_s[s(K-1) - \bar{s}]] + +\mu_p \Delta t[[\varphi_3(K-1)Y_{PX} - D(K-1)p(K-1)]]^2 + [\varphi_3(K-1)Y_{PX} -$$
$$D(K--1)p(K-1]\left[\lambda_p(K) + \mu_p[p(K-1) - \bar{p}]\right] = e_{\Delta t}(K) = 0 \tag{6.48}$$

The calculation with the gradient procedure will continue until the necessary conditions for optimality towards state and control variables are fulfilled. This means that the values of the gradients are approximately equal to zero:

$$\|e_v(k)\| \le \varepsilon_v, \varepsilon_v > 0, v = x, s, p, D, T \quad k = \overline{0, K-1} \tag{6.49}$$

The obtained values for the sampling interval and for the trajectories of state and control are projected over the constraints domain (6.49) respectively to account for the constraints

$$v^{j,j}(k) = \left\{ \begin{array}{l} v_{\min}, \ if \ v^{j,j}(k) < v_{\min} \\ v^{j,j}(k), \ if \ v_{\min} \le v^{j,j}(k) \le v_{\max} \\ v_{\max} \ \ if \ v^{j,j}(k) > v_{\max} \\ \quad k = \overline{0, K} \ \text{ for } v = x, s, p, k = \overline{0, K-1} \\ \end{array} \right\} \tag{6.50}$$

6.3.4 The coordinating sub-problem for continuous fermentation process

The coordinating variables are obtained from the necessary conditions for optimality of the augmented Lagrange functional. For the conjugate variables these are equations from (6.13)-(6.18). The necessary conditions for optimality according to the new introduced interconnections in time domain are:

$$\frac{\partial L}{\partial \delta_x(k)} = -\lambda_x(k) - \mu_x[-\delta_x(k) + f_x(k)] = -\lambda_x(k) - \mu_x e_{\lambda x}(k) = e_{\delta x}(k) = 0 \tag{6.51}$$

$$\frac{\partial L}{\partial \delta_s(k)} = -\lambda_s(k) - \mu_s[-\delta_s(k) + f_s(k)] = -\lambda_s(k) - \mu_s e_{\lambda s}(k) = e_{\delta s}(k) = 0, \tag{6.52}$$

$$\frac{\partial L}{\partial \delta_p(k)} = -\lambda_p(k) - \mu_p[-\delta_p(k) + f_p(k)] = -\lambda_p(k) - \mu_p e_{\lambda p}(k) = e_{\delta p}(k) = 0, \tag{6.53}$$

$\delta_v(k)$ is also in the expression for $\lambda_v[-\delta_v(k) + f_v(k)]$ where

$-\delta_v(k) + f_v(k) = \ e_{\lambda v}(k), k = \overline{0, K-1}$

The necessary conditions for optimality (6.13), (6.15) and (6.17) and (6.51)-(6.53) can be combined in a gradient procedure. Equations (6.51)-(6.53) give the values of the gradients for the conjugate variables multiplied by the penalty co-efficient. $\mu_v, \ v = x, s, p$

The equations (6.51)-(6.53) can be represented in the following way:

$$\frac{\partial L}{\partial \delta_v(k)} = -\lambda_v(k) - \mu_v \frac{\partial L}{\partial_{\lambda_v}(k)} \tag{6.54}$$

Gradient procedure can be built for both type of coordinating variables.

For conjugate variables:

$$\lambda_v^{j+1}(k) = \lambda_v^{j}(k) + \alpha_{\lambda_v} e^{j}{}_{\lambda v}(k), \tag{6.55}$$

where

$$e^{j}{}_{\lambda v}(k) = -\delta_v^{j}(k) + f^{j}{}_{v}(k), v = x,s,p, k = \overline{0,K-1}, \Delta t = \Delta t^{j} \tag{6.56}$$

For the interconnection in time domain

$$\delta_v^{j+1}(k) = \delta_v^{j}(k) - \alpha_{\delta_v} e_{\delta_v}(k),\ k = \overline{0,K-1} \tag{6.57}$$

where $e_{\delta v}(k)$ is calculated according to equations(6.51)-(6.53), α_{λ_v} and α_{δ_v} are the steps of the procedures for calculation of the conjugate variables and the interconnection in time, respectively. As the calculations are based on gradient iterative procedure, they will stop when the gradient is going to zero, or practically achieving some conditions for optimality, given by

$$\|e^{j+1}{}_{\lambda v}(k) - e^{j}{}_{\lambda v}(k)\| \le \varepsilon_{\lambda v}, \varepsilon_{\lambda v} > 0, k = \overline{0,K}, v = x,s,p$$

$$\|e^{j+1}{}_{\delta v}(k) - e^{j}{}_{\delta v}(k)\| \le \varepsilon_{\delta v}, \varepsilon_{\delta v} > 0, k = \overline{0,K-1}, v = x,s,p \tag{6.58}$$

where $\varepsilon_{\delta v}$, ε_{λ_v} are a very small positive numbers. For every step of the gradient procedure for the coordinating process the full iteration cycle of the sub problems of the first level is done. The calculations of the coordinating variables are done with the values of state and control variables of the first level sub problems and the calculations of the first level sub problems are done with the values of the calculated on the previous iteration coordinating variables. The iterative process of coordination and the optimal solution of the whole problem for optimal control is obtained when the optimal solution of the coordinating sub problem is reached, this means conditions (6.58) are fulfilled and if this condition is not reached the error is computer.

$$e^{j+1} = \left\{\left\|e^{j+1}{}_{\lambda v}\right\|^2 + \left\|e^{j+1}{}_{\delta v}\right\|^2\right\}^{1/2}, v = x, s, p \tag{6.59}$$

The new penalty coefficient is calculated from the conditions.

$$if\, j = 1\ or\ e_v{}^{j+1} < e_v{}^j \text{ then } \mu_v{}^{j+1} = \mu_v{}^j \tag{6.60}$$

$$if\, j > 1 \text{ and } e_v{}^{j+1} \geq e_v{}^j \text{ then } \mu_v{}^{j+1} = \alpha\mu_v{}^j, \alpha = [\,0.1\ 10.0], v = x, s, p \tag{6.61}$$

The coordinating sub-problem for Δt is given by the analytical solution of the necessary conditions for optimality of equation (6.28) as follows

$$\mu_x \Delta t[\varphi_3(K-1) - k_d x(K-1) - D(K-1)x(K-1)]^2 + \mu_s \Delta t\left[\frac{\varphi_3(K-1)}{Y_{XS}} + +mx(K-1) - \right.$$
$$\left. D(K-1)[s^0 - s(K-1)]\right]^2 + .$$
$$+\mu_p \Delta t\left[\varphi_3(K-1)Y_{PX} - D(K-1)p(K-1)\right]^2 = -K - [\varphi_3(K-1) - k_d x(K-1) -$$
$$-D(K-1)x(K-1)][\lambda_x(K) + [x(K-1) - \bar{x}]] + \left[\frac{\varphi_3(K-1)}{Y_{XS}} + mx(K-1) - -D(K-1)[s^0 - \right.$$
$$\left. s(K-_1)]\right]\left[\lambda_s(K) + \mu_s[s(K-1) - \bar{s}]\right] - [\varphi_3(K-1)Y_{PX} - -D(K-1)p(K-1]\left[\lambda_p(K) + \right.$$
$$+\mu_p[p(K-1) - \bar{p}]] \tag{6.62}$$

Therefore the analytical solution is:

$$\Delta t^{j+1} = \Big\{-K - [\varphi_3(K-1) - k_d x(K-1) - D(K-1)x(K-1)][\lambda_x(K) + \mu_x[x(K-1) - \bar{x}]] +$$
$$+\left[\frac{\varphi_3(K-1)}{Y_{XS}} + mx(K-1) - D(K-1)[s_i - s(K-1)]\right]\left[\lambda_s(K) + \mu_s[s(K-1) - \bar{s}]\right] -$$
$$-[\varphi_3(K-1)Y_{PX} - D(K-1)p(K-1]\left[\lambda_p(K) + \mu_p[p(K-1) - \bar{p}]\right]\Big\} / \Big\{\mu_x[\varphi_3(K -$$
$$1) - k_d x(K-1) - D(K-1)x(K-1)]^2 + \mu_s\left[\frac{\varphi_3(K-1)}{Y_{XS}} + mx(K-1) - D(K-1)[s^0 - s(K -\right.$$
$$\left. 1)]\right]^2 + +\mu_p[\varphi_3(K-1)Y_{PX} - D(K-1)p(K-1)]^2\Big\} \tag{6.63}$$

6.3.5 Sufficient conditions for optimality for continuous fermentation process

In order to check if the solution found gives minimum value of the criterion, the second derivative of the Lagrangian is derived using equation (6.28). The second derivative is:

$$\frac{\partial^2 L_a}{\partial \Delta t^2} = \mu_x[\varphi_3(K-1)-k_d x(K-1) - D(K-1)x(K-1)]^2 + \mu_s\left[\frac{\varphi_3(K-1)}{Y_{XS}} + mx(K-1) - \right.$$

$$\left. -D(K-1)[s_i - s(K-1)]\right]^2 + \mu_p[\varphi_3(K-1)Y_{PX} - D(K-1)p(K-1)]^2 \qquad (6.64)$$

It can be seen that $\frac{\partial^2 L_a}{\partial \Delta t^2} > 0$ if $\mu_x > 0, \mu_s > 0, \mu_p > 0$. This is always satisfied as the penalty coefficients are selected as positive. The conclusion is that the solution of the problem determines the minimum time of the problem for minimum start-up time of the process

6.4 Algorithm of the method for continuous fermentation process

(1.) At second level of the computing structure the values of

- The numbers of steps K
- The number of iterations for the gradients procedures, M1, M2
- The values of the coordinating variables $\lambda^j(k) k = \overline{0, K}, \delta^j(k)$, $k = \overline{0, K-1}, \Delta t^j$
- Penalty coefficient $\mu_v, v = x, s, p$
- Errors $\varepsilon_v, e_{\lambda v}, v = x, s, p$
- Final values of state variables, $\bar{v} = \bar{x}, \bar{s}, \bar{p}$
- Initial values of x, s, p, T, D are set up $i = 1$

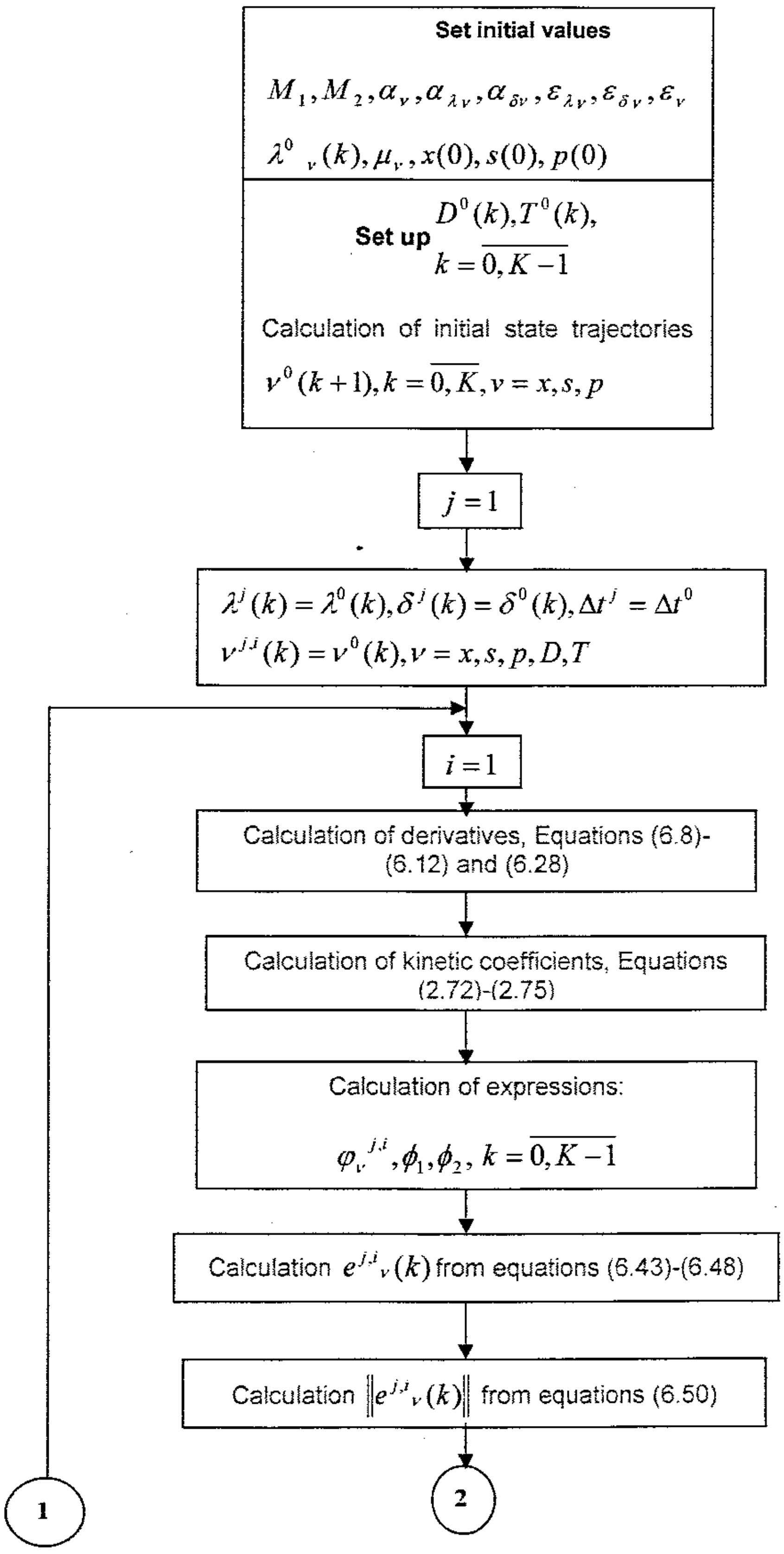
The coordinating variables are sent to the first level

(2.) At first level the initial trajectories of the control $D(k), T(k), k = \overline{0, K-1}$ are set and the initial trajectories of the states $x(k), s(k), p(k), k = \overline{0, K}$ are calculated.

(3.) The gradients $e^{j,l}{}_v(k), v = x, s, p, T, D, k = \overline{0, K-1}$ are calculated and the new state and control trajectories are obtained from equations (6.43)-(6.48). They are projected to the constraint domain (6.50)

(4.).The state and control error conditions are checked. If they are satisfied the obtained state and control trajectories are transferred to the second level. If the conditions are not satisfied, items 3, 4 are repeated, $i = i + 1$

(5.) The new values of the coordinating variables are calculated from equation (6.55)-(6.57), $j = j + 1$ and conditions (6.58) are checked. If they are satisfied the optimal solution of the coordinating sub-problem and of the global problem are obtained. If these are not satisfied, new values of the penalty coefficients are calculated according to (6.60)-(6.61) and items 3, 4, 5 are repeated and so on.



Set initial values
$M_1, M_2, \alpha_\nu, \alpha_{\lambda\nu}, \alpha_{\delta\nu}, \varepsilon_{\lambda\nu}, \varepsilon_{\delta\nu}, \varepsilon_\nu$
$\lambda^0{}_\nu(k), \mu_\nu, x(0), s(0), p(0)$
Set up $D^0(k), T^0(k), k=\overline{0,K-1}$
Calculation of initial state trajectories
$\nu^0(k+1), k=\overline{0,K}, \nu = x, s, p$
$j=1$
$\lambda^j(k)=\lambda^0(k), \delta^j(k)=\delta^0(k), \Delta t^j=\Delta t^0$
$\nu^{j,i}(k)=\nu^0(k), \nu=x,s,p,D,T$
$i=1$
Calculation of derivatives, Equations (6.8)-(6.12) and (6.28)
Calculation of kinetic coefficients, Equations (2.72)-(2.75)
Calculation of expressions:
$\varphi_\nu{}^{j,i}, \phi_1, \phi_2, k=\overline{0,K-1}$
Calculation $e^{j,i}{}_\nu(k)$ from equations (6.43)-(6.48)
Calculation $\left\|e^{j,i}{}_\nu(k)\right\|$ from equations (6.50)
1
2

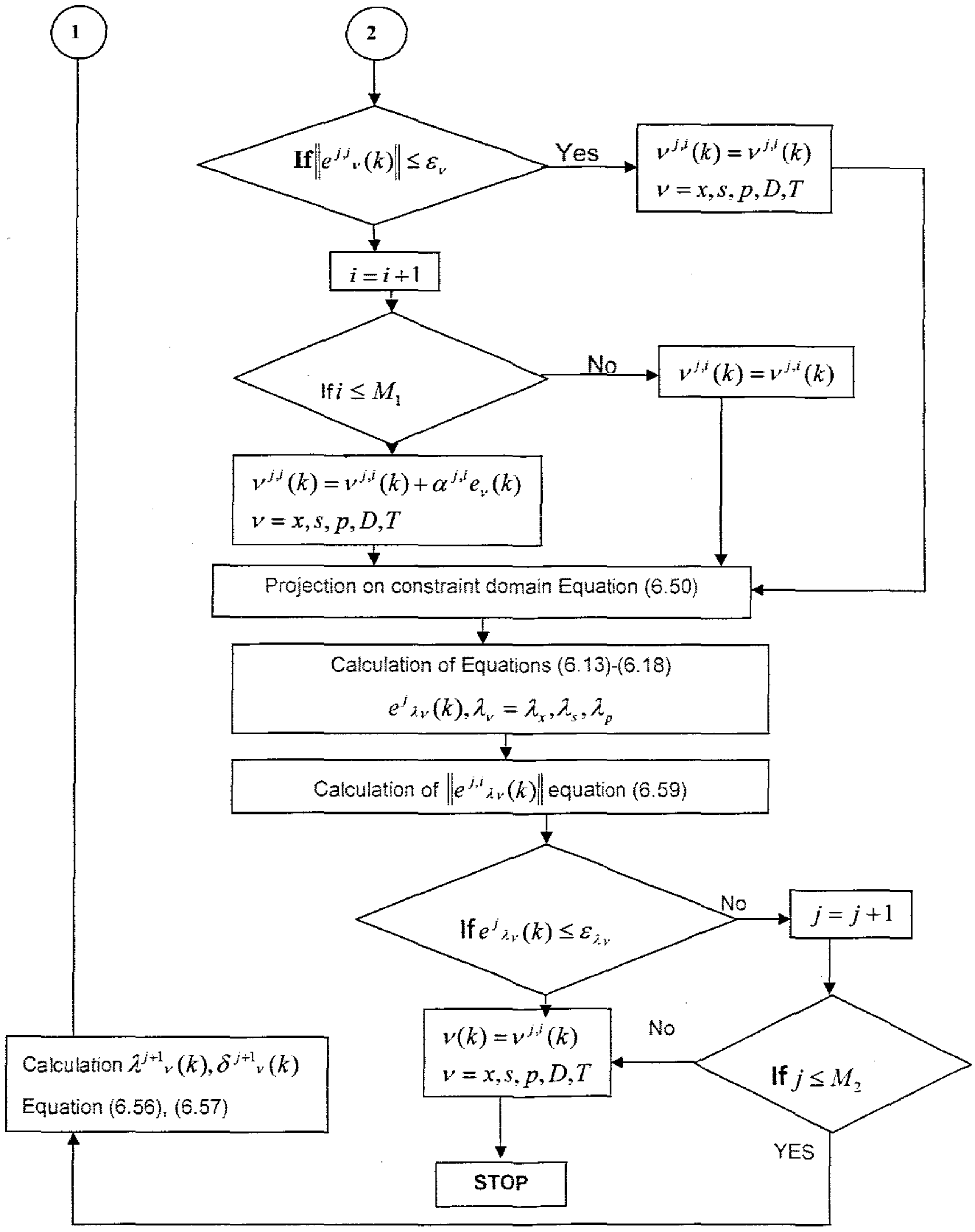

1
2
If $\|e^{j,i}{}_{\nu}(k)\| \le \varepsilon_{\nu}$
Yes
$\nu^{j,i}(k) = \nu^{j,i}(k)$
$\nu = x, s, p, D, T$
$i = i + 1$
If $i \le M_1$
No
$\nu^{j,i}(k) = \nu^{j,i}(k)$
$\nu^{j,i}(k) = \nu^{j,i}(k) + \alpha^{j,i} e_{\nu}(k)$
$\nu = x, s, p, D, T$
Projection on constraint domain Equation (6.50)
Calculation of Equations (6.13)-(6.18)
$e^{j}{}_{\lambda\nu}(k), \lambda_{\nu} = \lambda_x, \lambda_s, \lambda_p$
Calculation of $\|e^{j,i}{}_{\lambda\nu}(k)\|$ equation (6.59)
If $e^{j}{}_{\lambda_\nu}(k) \le \varepsilon_{\lambda_\nu}$
No
$j = j + 1$
No
If $j \le M_2$
YES
$\nu(k) = \nu^{j,i}(k)$
$\nu = x, s, p, D, T$
STOP
Calculation $\lambda^{j+1}{}_{\nu}(k), \delta^{j+1}{}_{\nu}(k)$
Equation (6.56), (6.57)

Figure 6.2: Flow chart of the algorithm for calculation of continuous fermentation process

6.5 Description and calculation results from the sequential program of the algorithm of the method in Matlab for continuos fermentation process

The algorithm presented in section 6.4 is implemented in Matlab in a sequential way(Appendix C). The table below represents the relationship between mathematical notations and program notations of the Matlab program.

Table 6.1: Correspondence between mathematical and programs notations for continuous fermentation process

Name	Notation in the text	Notation in Matlab
State variable Biomass	x	x
Substrate	s	s
Product	p	p
Control variable Temperature	T	T
Dilution	D	D
Steps of the gradient procedures for state variables	α_x	ax
	α_s	as
	α_p	ap
	$\alpha_{\lambda x}$	alx
	$\alpha_{\lambda s}$	als
	$\alpha_{\lambda p}$	alp
	$\alpha_{\delta x}$	adx
	$\alpha_{\delta s}$	ads
	$\alpha_{\delta p}$	adp
Conjugate variable	λ_x	lx
	λ_s	ls
	λ_p	lp
Interconnections in time domain	δ_x	dx
	δ_s	ds
	δ_p	dp
Coefficients	a_1, a_2, a_3	a1, a2, a3
	b_1, b_2, b_3	b1, b2, b3
	c_1, c_2, c_3	c1, c2, c3
	d_1, d_2, d_3	d1, d2, d3
	K_d	Kd
	K_s	Ks
	m	m

	n	n
	Y_{xs}	Yxs
	Y_{px}	Ypx
	s_i	si
	p	pl
	Δt	dt
	μ_{max}	mmax
Initial values of state variables	x(0)	x0
	s(0)	s0
	p(0)	p0
Penalty coefficients	μ_x	mx
	μ_s	ms
	μ_p	mp
Min and max of state variables	x_{max} x_{min}	xmax xmin
	s_{max}	smax
	s_{min}	smin
	p_{max}	pmax
	p_{min}	pmin
Min and max of control variables	T_{max}	Tmax
	T_{min}	Tmin
	D_{max}	Dmax
	D_{min}	Dmin
Derivatives of the model parameters	$\frac{\partial \mu_{max}}{\partial T} = a_2 + 2a_3 T(k)$	dmmaxdT(k) = a2+2*a3*T(k);
	$\frac{\partial K_s}{\partial T} = b_2 + 2b_3 T(k)$	dKsdT(k)=b2+2*b3*T(k);
	$\frac{\partial Y_{xs}}{\partial T} = c_2 + 2c_3 T(k)$	dYxsdT(k)=c2+2*c3*T(k);
	$\frac{\partial Y_{px}}{\partial T} = d_2 + 2d_3 T(k)$	dYpxdT(k)=d2+2*d3*T(k);
Error	$e_{\lambda x}(k) = \partial x(k) - x(k+1)$	elx(k) = dx(k)-x(k+1)
	$e_{\lambda s}(k) = \partial s(k) - s(k+1)$	els(k) = ds(k)-s(k+1)

	$e_{\lambda p}(k) = \partial p(k) - p(k+1)$	elp(k) = dp(k)-p(k+1)
	$e_{\delta x}(k) = -\lambda_x(k) - \mu_x e_{\lambda x}(k)$	edx(k)= -lx -mx*elx(k)
	$e_{\delta s}(k) = -\lambda_s(k) - \mu_s e_{\lambda s}(k)$	eds(k)= -ls-ms*els(k)
	$e_{\delta p}(k) = -\lambda_p(k) - \mu_p e_{\lambda p}(k)$	edp(k)=-lp-mp*elp(k)
Formulas	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^n = \varphi_f(k)$	ff(k)=dt*mmax(k)*((1-pk/pl)^0.52)
	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^n \frac{s(k)}{K_s + s(k)} = \varphi_x(k)$	fx(k)=ff(k)*s(k)/(Ks(k) + s(k))
	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^n \frac{x(k)s(k)}{[K_s + s(k)]^2} = \varphi_s(k)$	fs(k)=ff(k)*Ks(k) * x(k)/((Ks(k) + s(k))^2)
	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^{n-1} n \frac{-1}{p^*} \frac{x(k)s(k)}{K_s + s(k)} = \varphi_p(k)$	fp(k)=dt*mmax(k)*((1- p(k)/pl)^(n-1))*x(k)*s(k)/(pl*(Ks(k) + s(k)))
	$\Delta t\left[1 - \frac{p(k)}{p*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} = \varphi_1(k)$	f1(k)=ff(k)*x(k)*s(k)/(Ks(k)+ s(k))
	$\Delta t \mu_{\max}\left[1 - \frac{p(k)}{p*}\right]^n \frac{x(k)s(k)}{[K_s + s(k)]^2} = \varphi_2(k)$	f2(k)=f1(k)/((Ks(k) + s(k)))^2
	$\mu_{max}\left[1 - \frac{p(K-1)}{p^*}\right]^n \left[\frac{x(K-1)s(K-1)}{K_s + s(K-1)}\right] = \varphi_3(K-1)$	f3(K)=mmax*(1-p(K)/p1)^0.52*(x(K)*s(K))/(Ks + s(K))

6.6 Description of the Matlab sequential program for optimal control of continuos fermentation process

The complex relationship between physiochemical and biological variables and sudden unexplained changes in the process are reflected in the values of the equation kinetic parameters equations (2.72)-(2.75) which are the same as from section 4.6 of chapter four. Table 6.2 defines process kinetic values that are calculated from Matlab by using *polyfit function* from a certain given values of the mass-balance model equations kinetic parameters.

Table 6.2: Model parameter values for continuous process

Symbol	Name	Value	Unit
a1	Model co-efficient	0.0004	
a2	Model co-efficient	0.0001	
a3	Model co-efficient	0.0002	
b1	Model co-efficient	0.2	
b2	Model co-efficient	0.03	
b3	Model co-efficient	0.01	
c1	Model co-efficient	2	
c2	Model co-efficient	0.3	
c3	Model co-efficient	0.021	
d1	Model co-efficient	0.004	
d2	Model co-efficient	0.006	
d3	Model co-efficient	0.001	
Kd	Death constant	0.00008	
pl	Inhibition factor p*.	50	
n	power coefficient	0.52	
m	Maintenance co-efficient.	0.0011	
x	Biomass concentration		g/l
s	Substrate concentration		g/l
p	Product concentration		g/l
u_{max}	Maximum growth rate	0.5	
dt	Sampling period	0.01967	
Y_{px}	Stochiometry coefficient for biomass growth	0.421	
K_s	Monod's constant	0.5	
Y_{xs}	Stochiometry coefficient for biomass growth	0.172	

Program *continuos_seque.m* is developed to implement the decomposition method in a sequential way. The program calculates the optimal values of the state and control variables according algorithm 6.4. The whole program is in appendix C. The function *modelcontinuos_func2* is used to calculate the trajectories of the derivatives of the state space vector, according to the Equations (2.93 –2.95). Then the program is called by the solver ode15s for integration of the model equations. The input variables are *t* the time at moment *t* from the considered time interval, *y* initial, or value of the state vector in the moment *t*. The output variables are the values of the state space vector derivatives *ydot*. The program is given in appendix C

6.6.1 Results from sequential calculation of continuous fermentation process

The program for sequential calculation is used to calculate the optimal trajectories of the control and state variables of the batch fermentation process. The optimal trajectories are presented in figure 6.3 -6.10.The calculated minimum length of the sampling period is $\Delta t = 0.01967$, the number of steps in the optimization horizon is $K = 256$ and the minimum time is equal to $\Delta t{*}K = 5.0355$ for reaching steady state behavior.

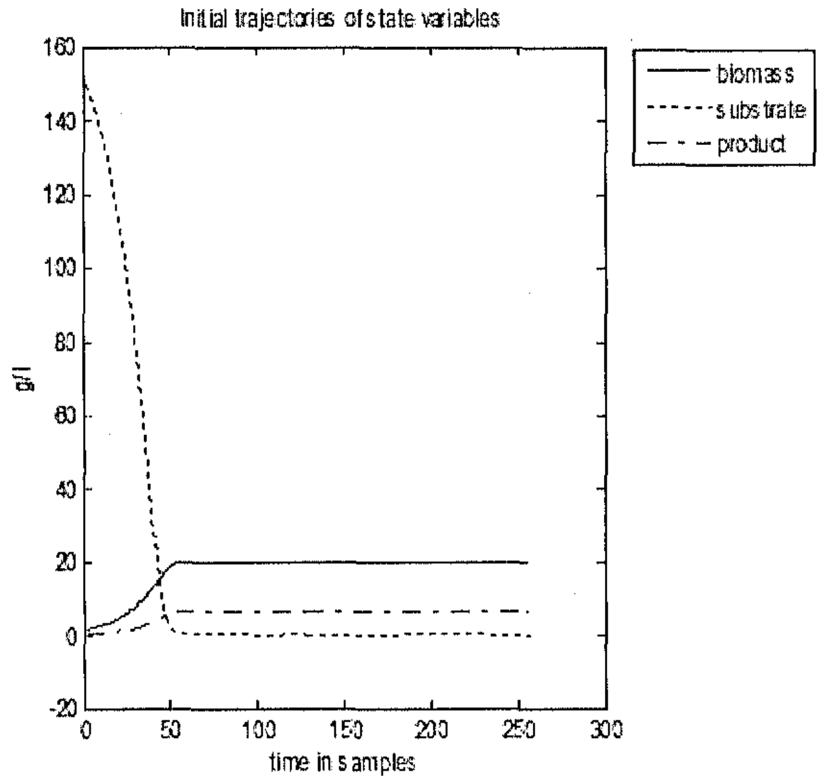


Figure 6.3: Initial trajectories of state variables for continuous fermentation process

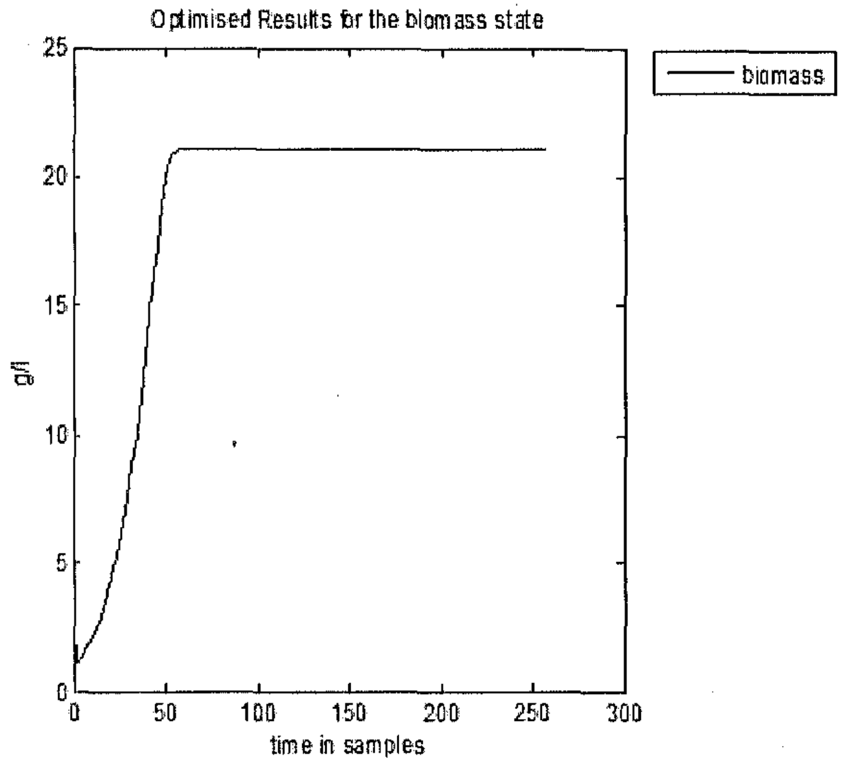


Figure 6.4: Optimal trajectory of biomass for continuous fermentation process

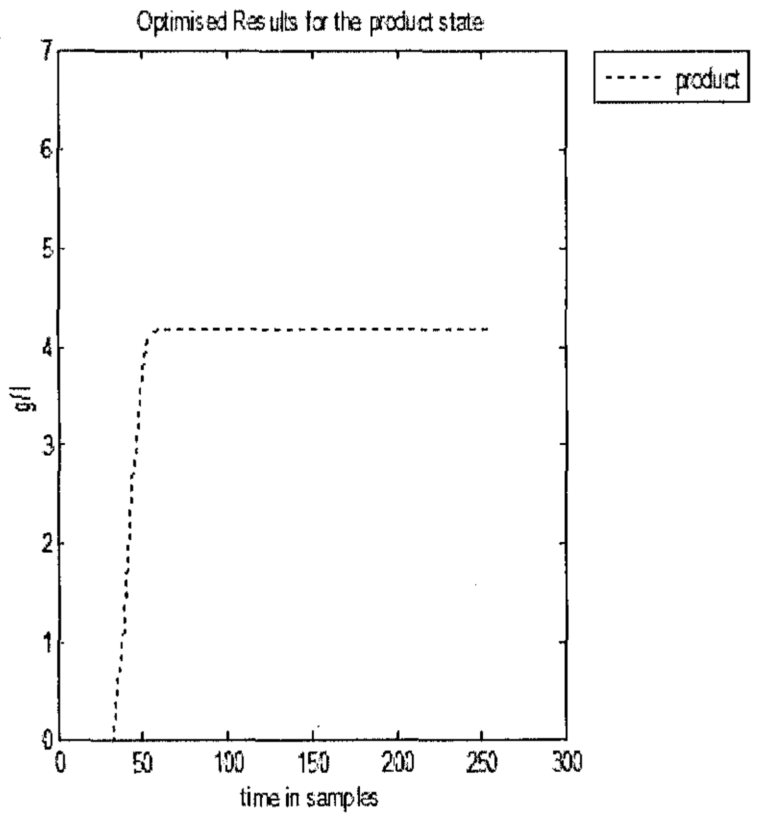


Figure 6.5: Initial trajectory of product for continuous fermentation process

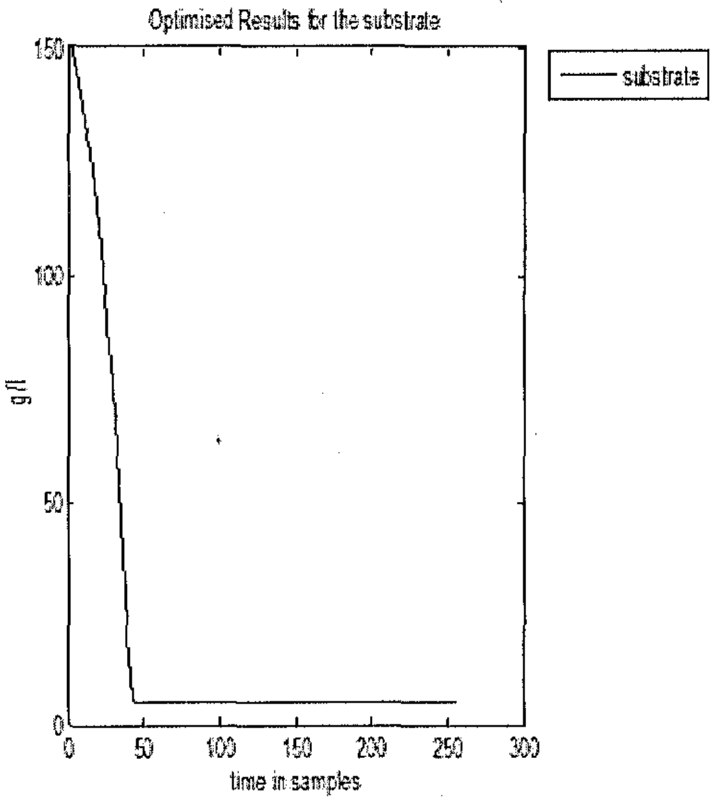


Figure 6.6: Optimal trajectory of substrate for continuous fermentation process

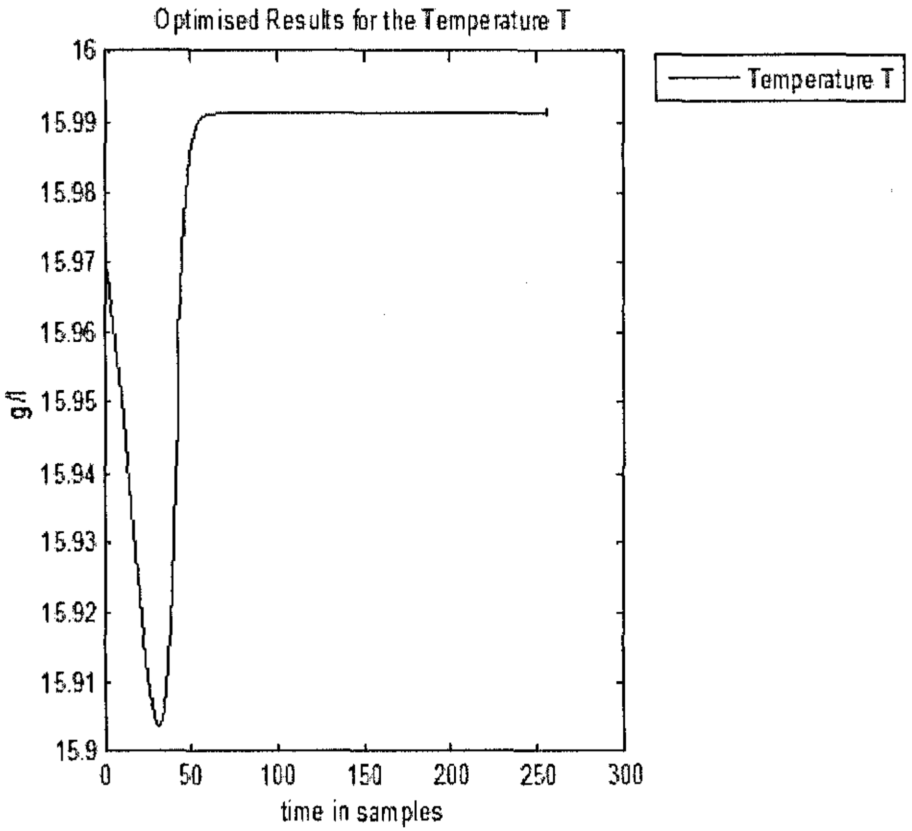


Figure 6.7: optimal trajectory of temperature for continuous fermentation process

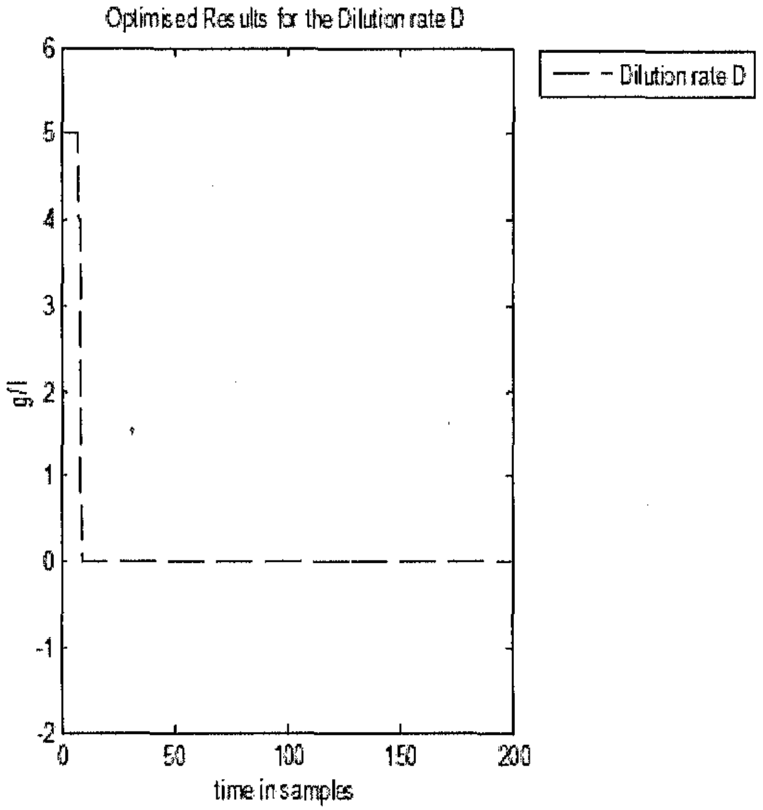


Figure 6.8: Optimal trajectory of Dilution rate for continuous fermentation process

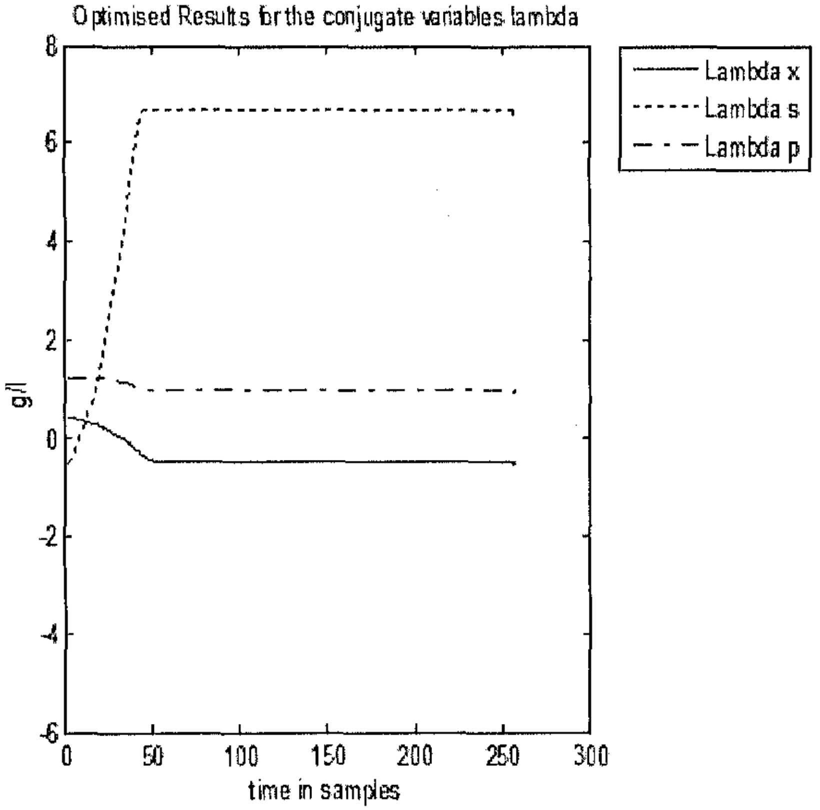


Figure 6.9: Optimal trajectories of conjugate variables for continuous fermentation process

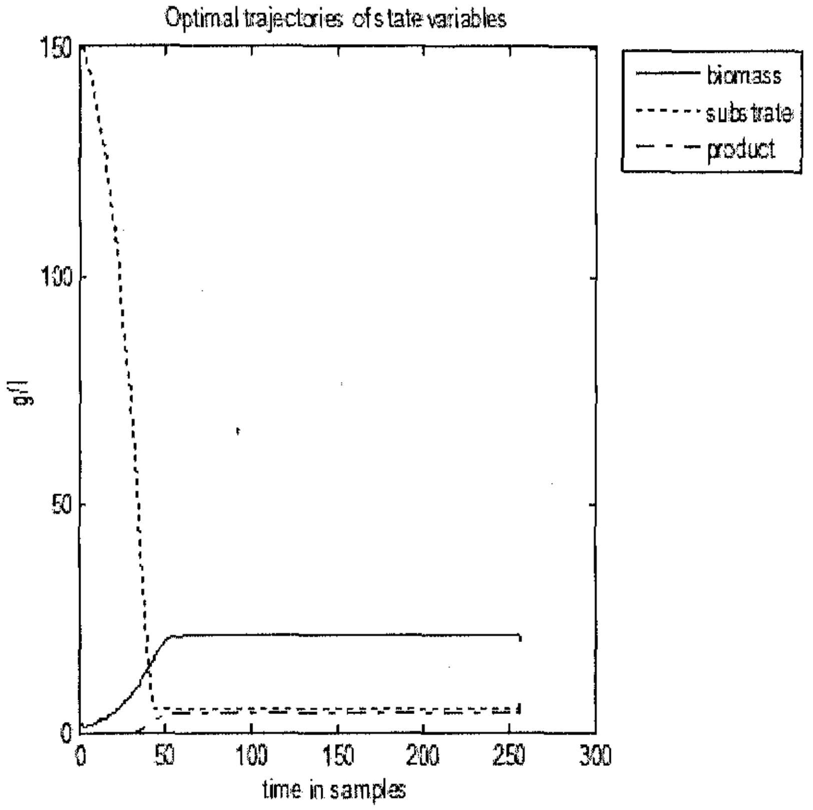


Figure 6.10: Optimal trajectories of state variables for continuous fermentation process

6.7 Discussion of the results for continuous fermentation process

The biomass concentration generally continued to rise after feed was started, as at that stage growth rate exceeded dilution rate, but within few hours after increases in the feed rate, it invariably began to fall. Substarte levels generally rose slowly throughout the feed period as desired although it is fairly sensitive to feed rate. Product concentration on the other hand rises sharply for several hours after the feed was switched on and then reached a plateau, rising slowly after the feed was exhausted.

6.8 Conclusion

This chapter describes the formulation of the problem for optimal control of a continuous fermentation process. An augmented Lagrange functional is used and a decomposition method is developed for solving discrete time formulation of the problem based on varational calculus and decomposition in time domain. The sequential algorithm of the decomposition method is described and implemented in Matlab. The parallel implementation of the decomposition method is given in chapter eight. Description of the Matlab cluster of computers as used for the calculation is given in chapter seven.

CHAPTER SEVEN
PARALLEL COMPUTING TOOLBOX AND MATLAB DISTRIBUTED COMPUTING SERVER ON A CLUSTER

7.1 Introduction

Computing costs have went done and computers have become larger and faster, the dynamic modelling and control of chemical progression at greater levels of detail is an area of growing interest. As opposed to commonly used steady-state process models, a dynamic model provides information about process, transient performance during startup and process upsets, and the coupling between unit operations. Inherently transient behaviour such as reaction kinetics can be included in the model. These advantages are mainly obvious for batch processes that do not show true steady-state behaviour in the first place. However, dynamic modelling and control involve greater computational effort than is needed to solve for a single steady-state operating condition. One technique useful in reducing computational time is to divide the problem into sub-problems and solve the parts in parallel on multiple computer processors. Parallel computing has been used to advantage in areas of science and engineering and has increasingly been useful to industrial engineering problems.

Section 7.1, the introduction is elaborating, why there is a need in applying parallel computing in chemical processes. Section 7.2 explains the concept of parallel computing. Section 7.3 gives a brief history of parallel Matlab environment. Section 7.4 gives some reasons why there should be parallel Matlab. Section 7.5 gives a brief survey in parallel Matlab. Section 7.6 describes Matlab Parallel Computing Toolbox and Matlab Distributed Computing Server and its concepts and section 7.7 describes program development based on algorithms developed in chapter 4,5,6,.

7.2 Parallel computing

Parallel computing is a form of computating that solves a problem using multiple computer resources at the same time. Hardware that supports parallel computing consists of multicore computer(Culler et al, 1998), symmetric multiprocessor, distributed computer such as cluster workstations(Wilkinson and Allen, 2005) and specialized parallel processors such as FPGA and GPU(Feng and Manocha, 2007) and application specific integrated circuit (AISC)(Robert et al, 1992). With the deployement of the hardware that supports parallel computing, use of multicore computer, parallel programming architecture becomes more important insolving complexy computing problems.

There are, however many reasons why parallel is the wave of the future in computing. The main obstacle for rapid introduction of parallel computing are the investments in softwares for sequential machines that have been made throughout the world, and difficulty in programming most parallel computers. The mechanism that is generally adopted is to increase the speed, completing larger tasks quicker and more efficiently. Like in all other cases speed is always the concern in computational world.

7.2.1 Amdahl's law

The performance of an algorithm on a parallel computing platform depends on parallelizing the algorithm to achieve better performance (Decegama, 1989). The small part of the program which cannot be parallelized limits the speedup available for parallelization. It can be stated mathematically

$$S = 1/((1-P)) \qquad (7.1)$$

where S is the speedup and P is the fraction that can be parallelized

7.2.2 Gustafson's law

It can also be represented in a mathematical form

$$S(P) = P - \acute{\alpha}(P-1) \qquad (7.2)$$

where S is equal to the speedup, P represents the number of processors and $\acute{\alpha}$ represents non-parallelizable part of the process(Decegama, 1989).

7.3 Parallel Computer Memory Architecture

There are different types of parallel computer memory architecture and are as follows

7.3.1 Shared memory

Shared memory parallel computers vary widely, but with carefully scheduling and problem subdivision it is possible for parallel computer to carry computation much more rapidly than single processor (Van Loan, 2000). In a shared memory environment each node is able to read and write in a shared memory. Each processor has its own memory and executes its own node program. The act of writing a program like this, is the act of writing a program for each node. The node programs typically reference local variables housed in the processors own memory and global variables that are situated in shared memory. The idea is for all node programs execute at the same time in a coordinated way that results in the required calculation. Figure 7.1 illustrates the situation.

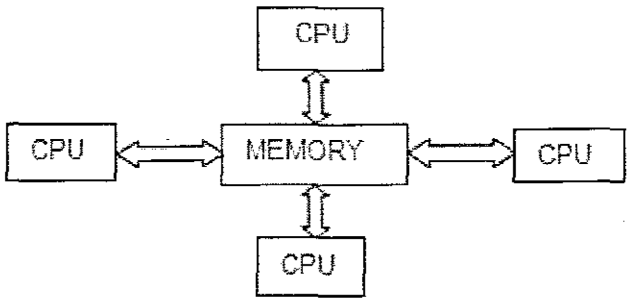


Figure 7.1: Shared memory multiprocessor

7.3.2 Distributed memory

In computer language, distributed memory refers to a multiple-processor computer system in which each computer has its own, private program memory(see Figure7.2). This requires tasks to be distributed to each CPU for processing. After that, data must be reassembled. This requires a communication network or switch. There is no global address space across all processors. Message passing interface (MPI) is associated with this memory architecture (Dongarra et. al, 1992).

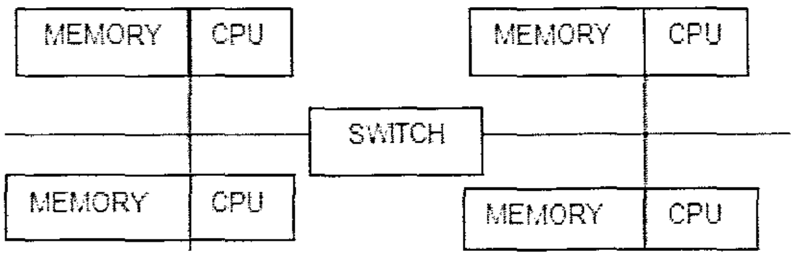


Figure 7.2: Distributed memory multi-processor

7.3.3 Distributed shared memory

This architectures employs both distributed and shared memory, .the shared memory is usually cache coherent symmetric multiprocessing (SMP) machine. Processors on a given SMP can address that other machine as global. Networking is required to move data from one SMP to other. This type of memory is used in multi-core computers, Figure7.3.

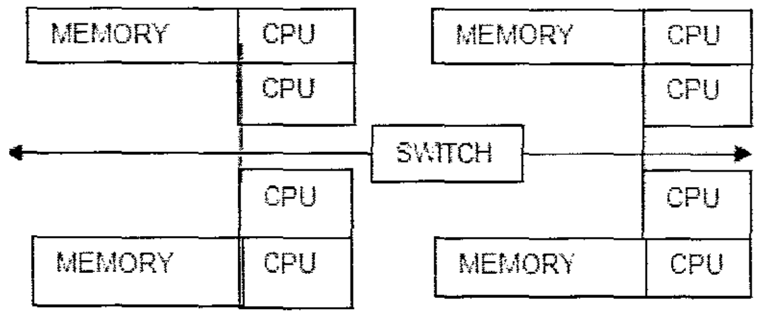


Figure 7.3: Distributed shared-memory

7.4 History of Parallel MATLAB

Cleve Moler, the original Matlab author and the cofounder of Mathworks worked on a version of Matlab for both an Intel hypercube and Ardent Titan in the late 1980s(Moler, 1995). Moler's 1995 article "Why there isn't a parallel Matlab"(Moler, 1995) described the three major obstacles in developing a parallel Matlab language: memory model, granularity of computations, and business situation. The variance between Matlab's global memory model and the distributed model of most parallel systems meant that the large data matrices had to be sent back and forth between the host and the parallel computer. Also at the time Matlab spent only a fraction of its time in parallelizable routines compared to parser and graphics routines which made a parallelization effort not very attractive. The last obstacle was simply a dose of reality for an organization with finite resources—there were simply not enough Matlab users that wanted to use Matlab on parallel computers, and the focus instead was on improving the uniprocessor Matlab.

A number of factors have made the parallel Matlab development a very important one inside Mathworks: Matlab has established into a most excellent technical computing environment supporting large scale projects, easy access to multiprocessor machines, and a need for a full fledged solution from the user community. Three approaches have been used to create a system for parallel computing with Matlab. The first approach aims at translating Matlab or similar programs into a lower-level language such as C or Fortran, and uses annotations and other mechanisms to generate parallel code from a compiler. Examples of such projects include Conlab(Jacobson et al, 1992), and Falcon(DeRose et al, 1995). Translating regular Matlab code to C or Fortran is a difficult problem. In fact, the Matlab Compiler software from the Mathworks switched from producing C code to producing wrappers around Matlab code and libraries to be able to support all the language features. Another technique is to make use of Matlab as a "browser" for parallel computations on a parallel computer while Matlab itself remains unchanged and the Matlab environment does not run natively on the parallel computer. This approach does not really classify as a "parallel Matlab" solution any more than a Web browser used to access a portal for launching parallel applications is itself a parallel application.

Last technique is to use Matlab via libraries. Recently, the MatlabMPI and pMatlab projects at MIT Lincoln Laboratory and the Multi Matlab project at Cornell University (with which The Math-works was also involved) are among the more successful and widely used libraries for parallel computing with Matlab. Other projects include ParaM and GAMMA projects(Panuganti et al, 2006), Parallel Toolbox for Matlab(Hollingsworth et al, 1996) (which uses PVM for message passing), and various MPI toolbox implementations for Matlab, including the most recent bcMPI (Blue Collar MPI) from Ohio Supercomputer center. The MathWorks introduced Parallel Computing Toolbox software and Matlab Distributed Computing Server in November 2004, (originally named Distributed Computing Toolbox™ and Matlab Distributed Computing Engine™, respectively). These fall into the last category of solutions. When they started to expand the capabilities of Matlab into parallel computing, they decided to target embarrassingly parallel problems, given that their initial survey showed a large number of the users wanted to simplify the process of running Monte Carlo or parameter sweep simulations on their groups computers.

Implicit and explicit multithreading of computations are other methods to parallelize Matlab computations on a single multicore or multiprocessor machine and will be discussed later in the chapter.

7.5 Why there should be parallel Matlab

Matlab has been a serial program. Cleve Moler(1995) of Mathworks stated not to develop parallel Matlab at that time. It has been fifteen years since the article was written, and many changes in the computing world have been seen. These changes invalidated disagreements that there should not be a parallel Matlab.

7.5.1 Memory model

Modern scientific and engineering problems develop in complexity, the computation time and memory requirements. The increment of processor speed and the amount of memory that can be mounted in one computer could not catch up with the rapidity of computation needs. Current technical problems basically do not fit into the memory of a one machine, making parallel computation to be essential. Combining with development in interconnect technologies, parallel computation has become a most useful tool for solving computational problems.

7.5.2 Granularity

Over the past fifteen years simple parallel Matlab jobs consists of only 2 m-files, have shown that multiple Matlab instances running on a parallel computer could be used to solve embarrassingly parallel problems, without any change to Matlab itself. Raise in computational problem size and processor speed have reduced the part of time spent in non-computation associated routines

7.5.3 Business situation

The past few years have seên the introduction of Beowulf clusters. Beowulf clusters are parallel machines that are prepared from commodity off-the-shelf (COTS) hardware. They are computers that are connected together with Ethernet switch or other common, non-proprietary interconnect. Beowulf clusters are mostly used by reasearchers over supercomputers because of their user-friendliness to setup and maintain and cheap enough .However, often not everyone in science can use parallel computing to solve problems. The leading way of parallel programming, MPI is too low-level and too error prone. Matlab is well known for its user-friendliness. There is a enormous potential market for a Matlab that could be used to program parallel computers.

7.6 Parallel Matlab survey

(Rajkiran et al, 2007) discussed the influence of parallel computing over the past years noted that irrespective of its complexity still remains the biggest task. This makes parallel computing to be recognized as the most challenging in the effective use of high-performance computing as highlighted by the DARPA High Productivity Computing Systems (HPCS) program. They mentioned the increase in high-level languages such as Matlab and Python that can provide enormous productivity advantages. Another key features for the Matlab success has been the availability of a different toolboxes (library components) for various fields.

(Coy and Edelman, 2003) did a survey and found through extensive web searching 27 parallel Matlab projects. These projects vary in their scope.

- Some are individual projects that offer basic embarrassingly parallel capabilities to Matlab.
- Some are for learning purposes or government lab investigate projects.

- Some are industrial projects that facilitate the user of Matlab in product expansion.
- Some run Matlab files into parallel code
- Some offer a parallel back-end to Matlab, using Matlab as a graphical front-end.
- Some others organize multiple Matlab processes to execute in parallel.
- While some are entering their second or third revision.

The expansion in IT promoting parallelization of many forms influence the Mathworks to alter its negative standpoint and to accelerate the development of parallel Matlab of its own(Moler, 2007). It appeared in 2004 in form of two products: Distributed Computing Toolbox (DCT) and Matlab Distributed Computing Engine. DCT is a toolbox that adds new constructs to the Matlabs C-like scripting programming language to provide parallel functionality. MDCE, a parallel run-time environment, provides the backend (behind-the-scenes) support for the DCT and is quite transparent to users. Version 1 of DCT/MDCE allowed to develop just coarse-grained parallel algorithms, which give rise to independent tasks without mutual interaction. This in Matlab terms "distributed" computing model follows in fact the manager-worker task-scheduling scheme and is suitable for embarrassingly parallel problems only. Since its initial release the Mathworks are strongly pushing forward the development of DCT/MDCE, resulting in a new version with substantial improvements released every year. In 2005 version 2 delivered among other important features the support for explicit communication among interdependent tasks, taking advantage of the industry-standard Message Passing Interface (MPI). This makes DCT/MDCE to a true message passing system and allows to develop parallel applications based on the general message passing model without restrictions. Version 3 of 2006 presented additional interesting parallel constructs, such as *distributed arrays* or the parfor loop, where one can recognize strong motivation from the so called *data* parallel approach typical for e.g. High Performance Fortran. From the user's point of observation the good news is, that since version 3.1 (2007) one can use up to four processors/cores of a symmetric multiprocessor without purchasing the costly DCE license.

The Mathworks announced new versions of its Matlab parallel computing products. Parallel Computing Toolbox (PCT) enables users to run applications in up to eight cores, empowering Matlab users to take more advantage of multi-core desktop computers without major programming efforts. Matlab Distributed Computing Server (MDCS) now supports Microsoft HPC Server 2008, providing users with access to the most recent enhancements in this cluster environment software. The Mathworks establish other products in the Matlab and Simulink product families that help users to make use of advanced hardware systems.

7.7 Matlab Parallel Computing Toolbox and Matlab Distributed Computing Server

Parallel Computing Toolbox (PCT) solve computationally and data intensive problem using Matlab and Simulink on multicore and multiprocessor computers. It consists of two separate products that make up the Mathworks Parallel computing environment: the Parallel Computing Toolbox (PCT) and the Matlab Distributed Computing Server (MDCS). PCT is a set of commands and functions that are executed from the Matlab command window, or from an m-file, just like other toolboxes. The computer on which PCT is running is defined here as the PCT client. MDCS is a service that runs in the background on computers running either worker or job manager processes. Figure 7.4 shows the basic Matlab distributed computing environment.(Parallel Computing toolbox users guide 4.2). The PCT client defines and submits jobs to the job manager. The job manager process, which is controlled by MDCS, "coordinates the execution of jobs and the evaluation of their tasks. The job manager can run on any computer that has MDCS installed. A worker is a separate process, also controlled by MDCS, which evaluates a task assigned to it by the job manager, and returns the results to the job manager. A worker process is usually installed on its own computer, so that multiple workers can operate simultaneously to complete a job faster. However, for initial development and debugging of a waveform, PCT, the job manager, and worker processes can all be installed on the same computer.

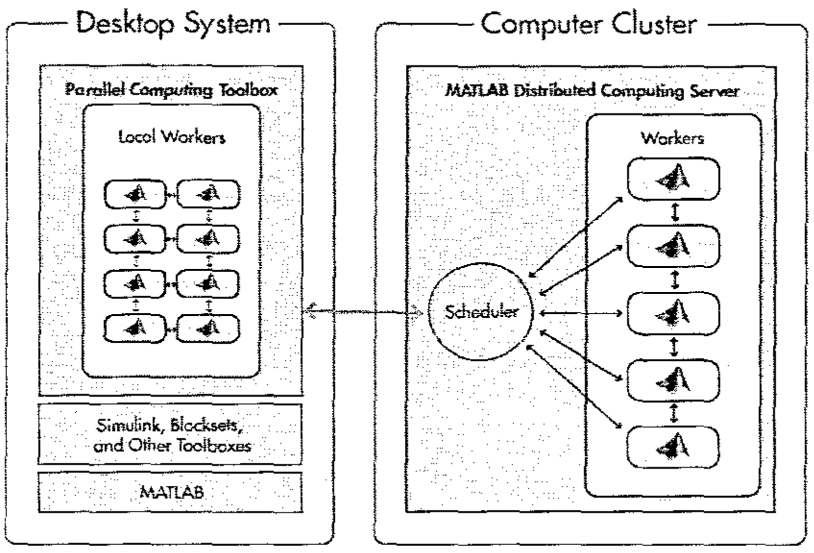

Desktop System
Parallel Computing Toolbox
Local Workers
Simulink, Blocksets, and Other Toolboxes
MATLAB
Computer Cluster
MATLAB Distributed Computing Server
Workers
Scheduler

Figure 7.4: Typical parallel cómputing cluster.(adapted from http://www.mathworks.com/products/parallel-computing/?s_cid=HP_FP_ML_parallelcomptbx)

To Develop parallel applications with parallel computing toolbox is possible in two ways. The toolbox permits application prototyping on the main computer with eight local nodes depending on the number of CPU's and with Matlab Distributed Computing server applications can be scaled to multiple computers on a cluster. In order to understand parallel processing some basic terms that needed to be known are discussed (Tokhi et al 2003).

7.7.1 Parallel Computing Toolbox Terminology

There are major terms that are continuously used in Parallel Computing Toolbox. The following table summarizers few of them.

7.7.2 Programming parallel applications in Matlab

In parallel computing choosing a type of parallelism can be very complicated; this is why it is important to consider this carefully.

MATLAB supports three kinds of parallelism: multithreaded, distributed computing, and explicit parallelism (Moler, 2007).

Table 7.1: Parallel Computing Terms

Client	The Matlab session that defines a job using the Distributed Computing Toolbox.
Job	The entire large-scale process to perform in Matlab composed of a set of tasks.
Job-manager	The part of the Matlab Distributed Computing Server that coordinates job execution, distributing tasks to individual workers for evaluation. This is represented in the client session by a job manager object.
Task	A task is the smallest unit of the work that needs to be done. In other words, it is the independently executable smallest unit of a program that is executed by one processor and concurrency among processors is exploited only across task
Worker	The session of Matlab in the Matlab Distributed Computing Engine that evaluates tasks by executing the tasks' functions. This is represented in the client session by a worker object.
Processes	A process performs the tasks assigned to it. Usually more than one task is assigned to a process. However, a process can consist of only one task. A program in the parallel executable format is composed of a number of processes, each of which performs a subset of tasks in the program.
Parallel Task	A task that can be executed by multiple processors safely
Serial Execution	Execution of a program sequentially, one statement at a time. In the simplest sense, this is what happens on a one processor machine. However, almost all parallel tasks. will have sections of a parallel program that must be executed in sequential way.
Parallel Execution	Execution of a program by many nodes, with each task able to execute the same or unlike statement at the same moment.

7.7.2.1 Multithreaded parallelism

In multithreaded one case of Matlab automatically creates multiple simultaneous instruction streams. Multiple cores, sharing the memory of a single machine execute these streams.

7.7.2.2 Distributed computing

In distributed, multiple occurrence of Matlab run multiple in-dependent computations on different computers, each with its own operating system.

7.7.2.3 Explicit parallelism

In explicit, some occasion of Matlab run on numerous processors, often with separate memories and simultaneously execute a single Matlab command on M-function including parallel *for loops* and distributed arrays.

7.7.3 Working in an interactive parallel environment

Parallel computing Toolbox extends the Matlab interactive environment, letting the user to use familiar Matlab environment for prototyping and developing task and data parallel applications. The *matlabpool* statement assigns a set of dedicated computational resources. It connects a Matlab session to a pool of Matlab workers that can run either locally on desktop or on a computer cluster. It sets up an interactive parallel execution environment which can execute a parallel Matlab code from the Matlab command prompt and retrieve results immediately as computations ends. In this environment parallel programming constructs such as *parfor* and *spmd* automatically detect the presence of workers and distribute computations and data between the Matlab session and the workers.

The Parallel Command Window (*pmode*) is an extension of the Matlab command window that sets up a data-parallel execution environment which can use distributed arrays, associated parallel functions, and message-passing functions. Commands issued at the parallel prompt are executed simultaneously on all participating workers and results are accessible immediately, it enables to better track application behavior across multiprocessor system at each step. Commands can be collected into a Matlab function and submitted for offline execution.

7.7.4 Working in Batch Environments

Parallel Computing Toolbox can use batch environments for offline execution, enabling to free Matlab client session for other activities while executing large Matlab and Simulink applications. MATLAB client session can be shut down while application executes and retrieve results later. The batch function can execute Matlab scripts offline. It offers a apparatus for data transfer between Matlab client and worker workspaces, which stops the user from explicitly managing data in multiple workspaces. The toolbox also provides job and task objects. These objects gives a lower-level but more general mechanism to perform parallel Matlab applications in batch environment. The batch command and the job and task objects can be used to off-load the execution of sequential Matlab programs from a client Matlab session to a node.

7.7.5 Cluster Using MATLAB Distributed Computing Server

Parallel Computing Toolbox offers the capability to use up to four local nodes on a multicore or multiprocessor computer using a one toolbox license. When used with Matlab Distributed Computing Server, can extent application to use any number of nodes running on a cluster of computers. Matlab Distributed Computing Server supports interactive and batch work-flows. Matlab applications that make use Parallel Computing Toolbox functions can be built into stand-alone executables and shared software components using Matlab compiler. The libraries can connect to Matlab Distributed Computing Server and execute Matlab computations on a computer cluster. Matlab Distributed Computing Server can be connected with many schedulers: the MathWorks job manager (provided with the product) or any other third-party scheduler such as Platform LSF®, Microsoft® Windows® Compute Cluster Server, Altair PBS Pro®, and TORQUE. Using the Configurations Manager in the toolbox can maintain named settings such as scheduler type, path settings, and cluster usage policies. Switching between schedulers or clusters normally needs changing the configuration name only. Matlab Distributed Computing Server energetically allows the need licenses on the cluster based on the user's profile when an application runs. As a result, administrator need to manage only a server license on the cluster rather than separate toolbox and block set licenses on the cluster for every cluster user.

7.7.6 Steps in running a distributed job using Parallel Computing Toolbox (PCT)

- **Find a Job Manager**: The network may have one or more job managers available. The command used to find a job manager creates an object in the current Matlab session to represent the job manager that will run the job.
- **Create a Job**: A job is created to hold a collection of tasks. The job exists on the job manager, but a job object in the local Matlab session represents that job.
- **Create Tasks**: While the job is in the pending state, tasks can be created to add to the job. Each task of a job can be represented by a task object in the local Matlab session.
- **Submits a Job to the Job Queue for Execution**: When the job has all its tasks defined, it is submitted to the queue in the job manager. The job manager distributes the job's tasks to the worker sessions for evaluation. When all of the workers are completed with the job's tasks, the job manager moves the job to the finished state.
- **Retrieve the Job's Results**: The resulting data from the evaluation of the job is available as a property value of each task object. All the steps above can be represented diagrammatically with Figure 7.5 below

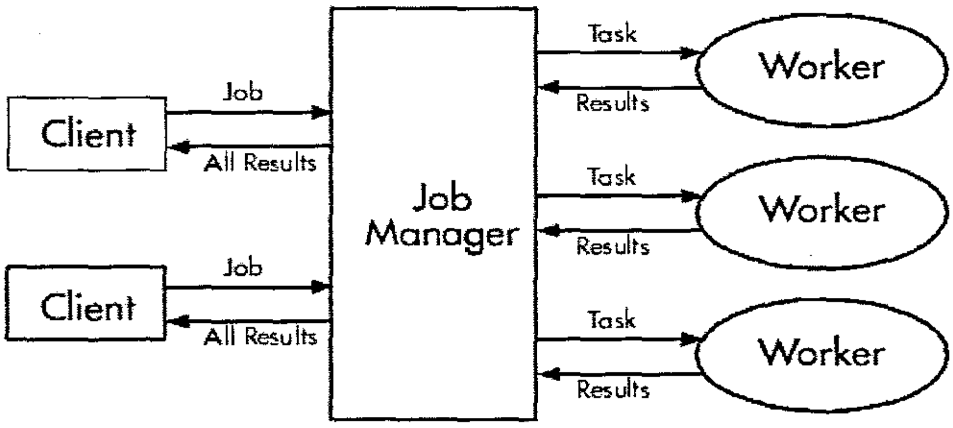


Figure 7.5: Client, job-manager and worker interaction

7.7.7 Code execution at the client and workers

The sequential code is executed at the client. If the code can be parallelizable, parallelizing function are called. After the scheduler or Mathworks job-manager sends data to workers. Results are collected from the workers at the client. Sequential code continues executing at client machine. This can be represented on Figure 7.6 below.

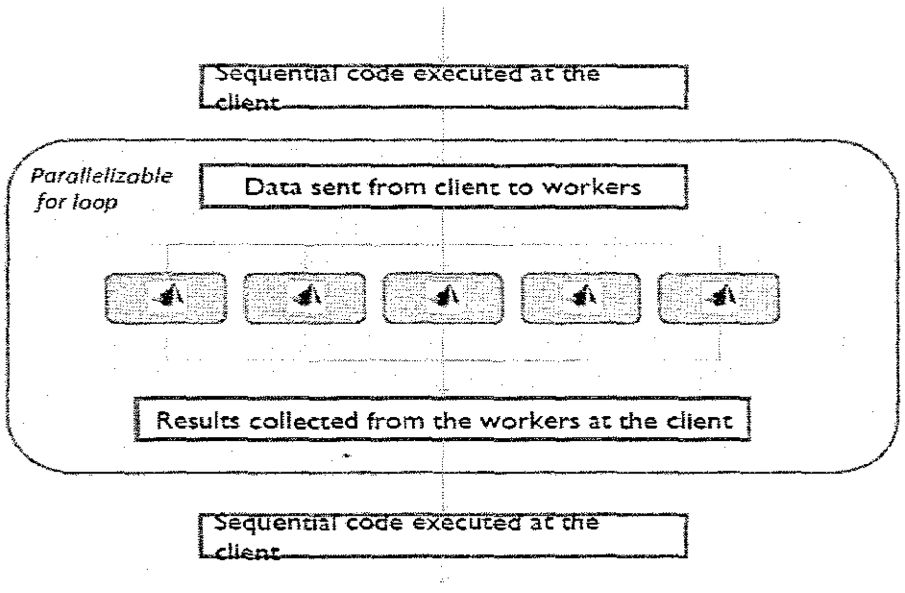

Sequential code executed at the client
Parallelizable for loop
Data sent from client to workers
Results collected from the workers at the client
Sequential code executed at the client

Figure 7.6: Code execution (adapted from Samuel et al, 2009)

7.7.8 Design of Load balancing Algorithm

It refers to the practice of allocating work among tasks so that all tasks are kept busy all the time. Load balancing is very important on designing parallel programs for performance reasons (R. Shah et al, 2007). Generally load balancing algorithm consists of two basic policies namely a transfer policy and a location policy. Location policy determines the under loaded processor. In other words it places corresponding nodes to/from which a node can send/receive workload to improve the cluster performance. Further, while balancing the load, types of information such number of jobs waiting in queue, job arrival rate, CPU processing rate, at each processor may be exchanged among the processors to improve system performance. Load can be classified as either static, dynamic and adaptive but the thesis uses static scheduling. The principle of load balancing can be presented in the following flowchart, Figure7.7.

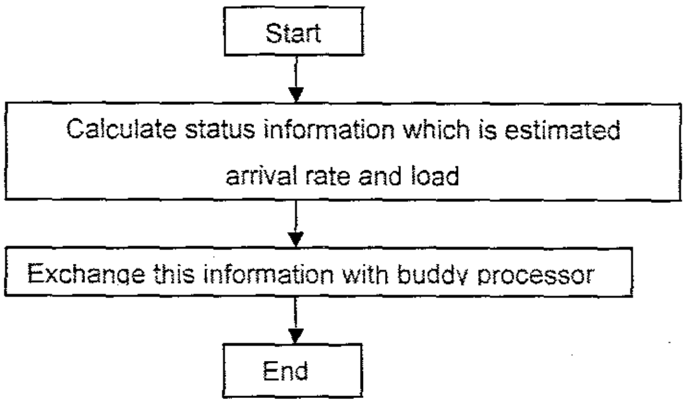

Start
Calculate status information which is estimated arrival rate and load
Exchange this information with buddy processor
End

Figure 7.7 A: Flow chart for load balancing
Processing by each processor i (Pi) on every status exchange

7.8 Program Development

The algorithms described in chapter 4, chapter 5 and chapter 6 are applied for organization of the calculations in two ways:

7.8.1 Sequential in one computer

This algorithm is used for calculation of the optimal state and control trajectories. Execution of a program resultss to a progression of calls to functions defined in different modules in a sequential way, called sequential composition. In sequential programming the program is run on a single computer meaning having single Central processing unit (CPU). Program is discretized in into series of instruction. Block diagram for sequential implementation is given on Figure7.8

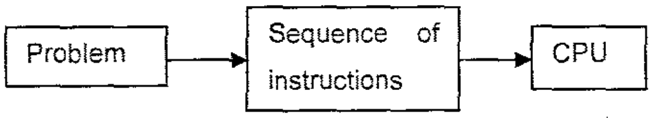


Figure 7.8: Sequential programming block diagram

In a parallel program that is created with the use of only sequential composition, each CPU executes the same program, which in turn performs a series of calls to separate program components. These program parts may communicate and coordinate, but they cannot generate new tasks. Hence, the whole computation progresses in serial from one parallel operation to the other. The sequential computation results are described in chapter4, 5 and 6.

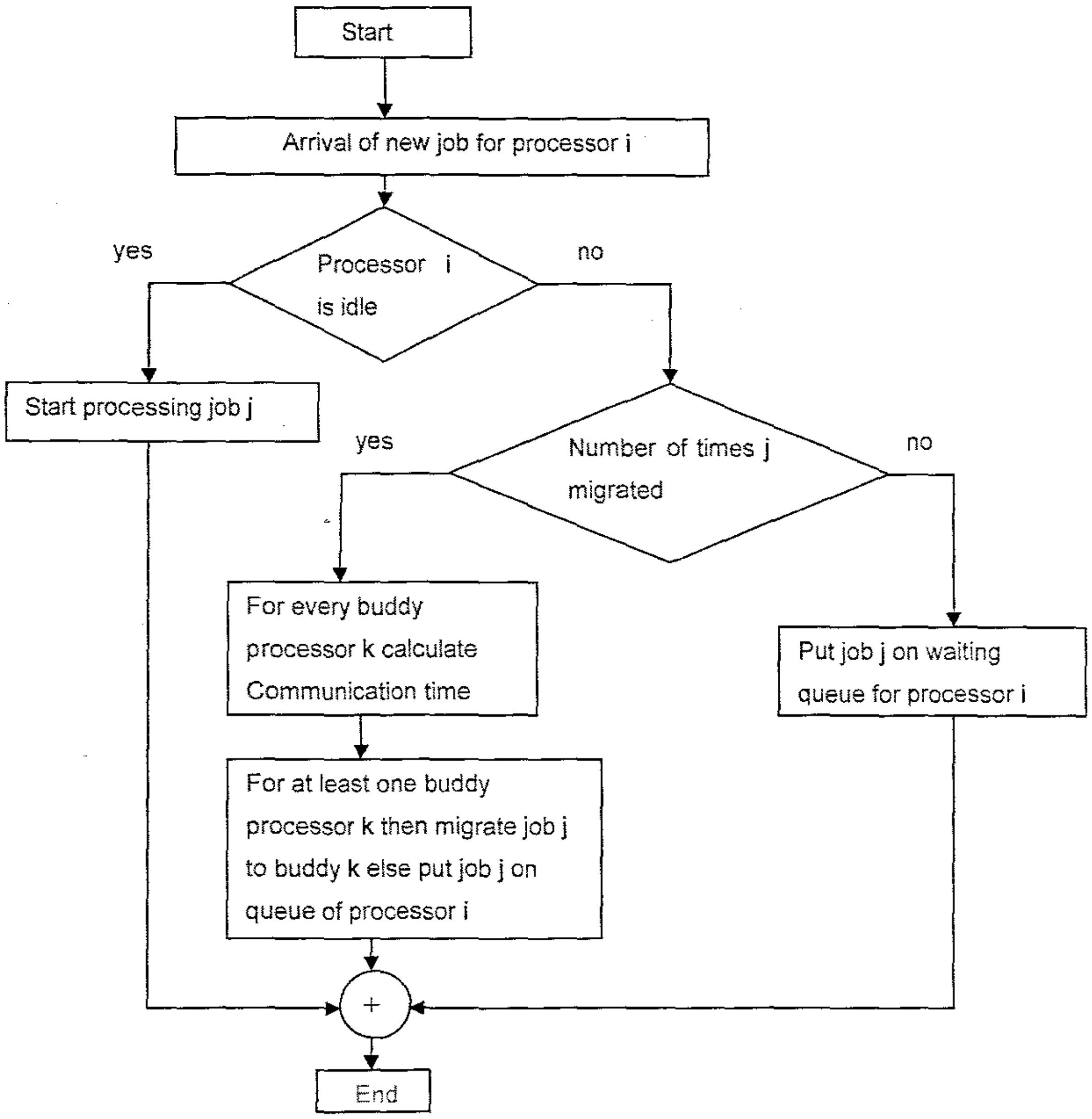

Start
Arrival of new job for processor i
yes
Processor i is idle
no
Start processing job j
yes
Number of times j migrated
no
For every buddy processor k calculate Communication time
Put job j on waiting queue for processor i
For at least one buddy processor k then migrate job j to buddy k else put job j on queue of processor i
+
End

Figure 7.7B: Flow chart for load balancing Processing by Pi on arrival of job j

7.8.2 Parallel, using the of computers

A parallel composition identifies which program components are to be processed in which division of the computer and how these components are to exchange data. In standard any program expressed as a parallel composition can be changed to a sequential composition that interleaves the execution of the range of program components appropriately. And hence the use of parallel composition can improve scalability and locality. Parallel programming can be achieved on one computer with multi-cores, in cluster of computers connected on one network. An example of parallel configuration of a cluster of computers is given on Figure7.9.

The algorithm for optimal control is programmed in both sequential and parallel way. The goal of parallelization is to obtain high performance and increased speed over the sequential program that solves the same problem. This is to ensure efficient load balancing among processors, reduction in communication overhead and synchronization. The parallel implementation and the results from the calculations are described in chapter 8.

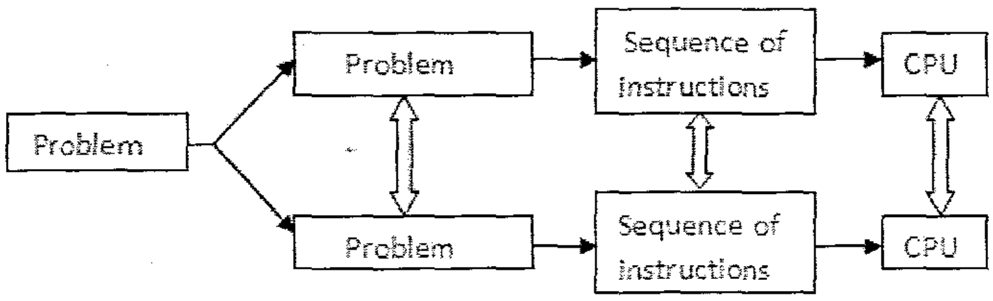


Figure 7.9: Parallel programming block diagram

multi-cores, in cluster of computers connected on one network. An example of parallel configuration of a cluster of computers is given on Figure7.9.

The algorithm for optimal control is programmed in both sequential and parallel way. The goal of parallelization is to obtain high performance and increased speed over the sequential program that solves the same problem. This is to ensure efficient load balancing among processors, reduction in communication overhead and synchronization. The parallel implementation and the results from the calculations are described in chapter 8.

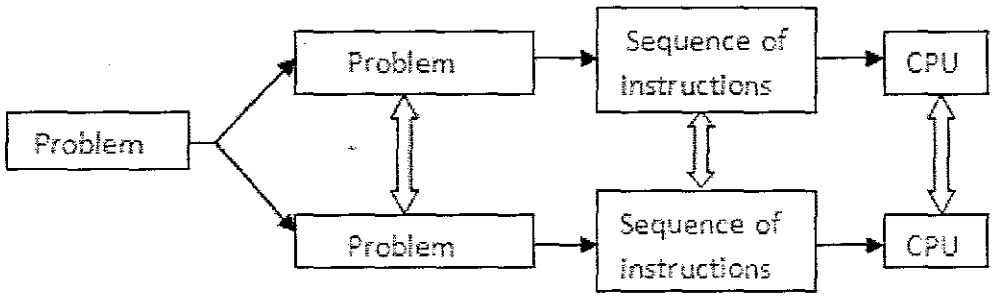


Figure 7.9: Parallel programming block diagram

7.9 Conclusion

A review of the developments of serial computers has revealed a lot of ways parallelism has been introduced into sequential computers. While the application of parallelism in sequential computers has been hidden from user program, it has been the key mechanism of delivering speed in computer from the architerature and system design perspective as opposed advance in solid-state electronic technology. Parallel applications are thus the next natural step in computational speed that can accompany and further the process in technology.

Although Mathworks hesitated with the introducing parallelism into Matlab, they finally completed a very good work. The Matlab engineers did their best to design the parallelization API in a Matlab-like, i.e. very user-friendly manner (including peculiar terminology). Thus, Parallel Matlab (PCT/MDCS) is an ideal environment for a smooth transition to the world of parallel scientific computing, especially for current Matlab users. The programs that are developed using PCT/MDCS in parallel way for batch, fed-batch and continuous fermentation process are given in appendices D, E and F

CHAPTER EIGHT
DESCRIPTION OF THE MATLAB PARALLEL PROGRAMS FOR OPTIMAL CONTROL PROBLEM OF FERMENTATION PROCESSES

8.1 Introduction

This chapter describes the programmes for parallel calculation of optimal state and control trajectories of batch, fedbatch and continuous fermentation processes.

The knowledge from chapter seven is used to implement the sequential algorithm of the sequential programs that are discussed and presented in chapter 4, chapter 5 and chapter 6 *in a parallel* way. *The algorithm* is implemented in parallel for the three fermentation processes i.e. batch, fed-batch and continuous, which is described in sections 8.2-8.4.

8.2 Matlab parallel program for optimal control of the batch fermentation process

The program *batch_parallel.m* transforms the sequential program for optimal control calculation of batch process using special functions and commands from the Parallel Computing Toolbox. The inputs and outputs are the same as for the sequential program. The *program subprob2_batch.m* is the program that is sent to the workers for parallel calculation of the optimal control and state trajectories *with the Filedependencies function from Parallel Computing Toolbox(PCT).* The definition of variables in a Matlab script-file *batch_parallel.m* is as follows:

global par T kd m pl n si K M1 eps1 mu l x s p el conmin conmax statemin statemax astate acon dt y0 x1 s1 p1 T1.

This variables are defined global since they have to be accessible at a shared memory which in this case is used as jobmanager. They are defined global in the script *batch_parallel* and *subprob2_batch.m.*

8.2.1 Starting parallel computation for batch fermentation process

The start of the coordinating procedure described in chapter4 is used for parallelization of the calculation of the state and control trajectories for parts of the optimization interval.

```
j=1
while j <= M2;   Second level iterations
jm = findResource('scheduler','configuration','jobmanagerconfig5')
pjob=createParallelJob(jm)
set(pjob,'MinimumNumberOfWorkers',numlabs)
set(pjob,'MaximumNumberOfWorkers',numlabs)
```

The function *subprog2_batch.m* is on the client path, but it has to be accesible to the workers.The FileDependencies property of the *pjob* is used to transfer this function to all the workers that are available on the computer cluster.

```
set(pjob,'FileDepandencies',{'subprog2_batch.m'})
```

Starting parallel computing is done by creating a task which is the same for every worker. The task has only one output with 4 output arguments and not input arguments to the function.

```
Task1=createTask(pjob, @subprob2_batch,4,{})
submit(pjob)
```

The function *subprog2_batch.m* is calculating the first level problem for an optimization interval with 256 periods, 32 machines, the time horizons for 8 steps and the last point for k = 257 is computed in the worker with index1.When the calculation completes, the output results are in the global variables x1, s1, p1, T1, and are transformed to the variables x, s, p, T, by the worker with index1

```
waitForState(pjob,'finished')
results=getAllOutputArguments(pjob)
```

The results are used for calculation of the improved values of the coordinating variables, which again are used for calculation of state and control variables until the convergence of the coordination process is achieved. The parallel version of the program is given in appendix D.1.

8.2.2 Function for parallel calculation of batch fermentation process

The part of the program implemented in a parallel way is described by the function file *subprog2_batch.m.*

function [*x,s,p,T*]=subprog2_batch

The same global parameters are declared.

```
global par T kd m pl n si K M1 eps1 mu l x s p el conmin conmax statemin statemax astate acon dt y0 x1 s1 p1 T1.
```

The *subprog2_batch.m* starts with calculation of the start*(a)* and stop*(b)* points of the optimization horizon for every worker.

```
dd=K/numlabs
a=(labindex-1)*dd+1
b=labindex*dd
```

Decomposition in time domain inside every subinterval is done

```
for k=a:b
i=1
```

The function determines the first level iterations number.

```
while i <= M1
```

Calculation of the expressions *Iv(k), k=0,K-1, v = x, s, p* is followed by calculation of the errors and the values of the state and control variables. The iterations continue until the state and control gradients are less than the error **eps1** or until the number of iterations becomes equal to M1. The program has to reach this point for every worker. The function *labBarrier* is used for this purpose: This function causes the workers that completed evaluation tasks earlier to wait for the ones that completed late.

labBarrier

The Parallel Computing Toolbox function *labBarrier* is used to synchronize the parallel calculations and to ensure that all workers have done the necessary calculations. Then the worker with labindex equal to 1 is forming the whole trajectory of state and control variables from the tasks using their global definition in the following way.

```
labBarrier
if labindex==1
x2=reshape(x1,1,numel(size(x1)));
s2=reshape(s1,1,numel(size(s1)));
p2=reshape(p1,1,numel(size(p1)));
T2=reshape(T1,1,numel(size(T1)))
x=x2([a:b,labindex])
s=s2([a:b,labindex])
p=p2([a:b,labindex])
T=T2([a:b,labindex])
```

The obtained trajectories are the output of the subprogram. They are used in the main program for calculation of the coordinating variables. The calculations stop when the number of iterations is reached or when the criteria for convergence of the coordinating procedure is satisfied. The program for calculation of the first-level subproblems in parallel is given in appendix D2.

8.2.3 Results from calculations with the Matlab parallel program for batch process

The optimal trajectories of state and control variables are given in figures8.1-8.6

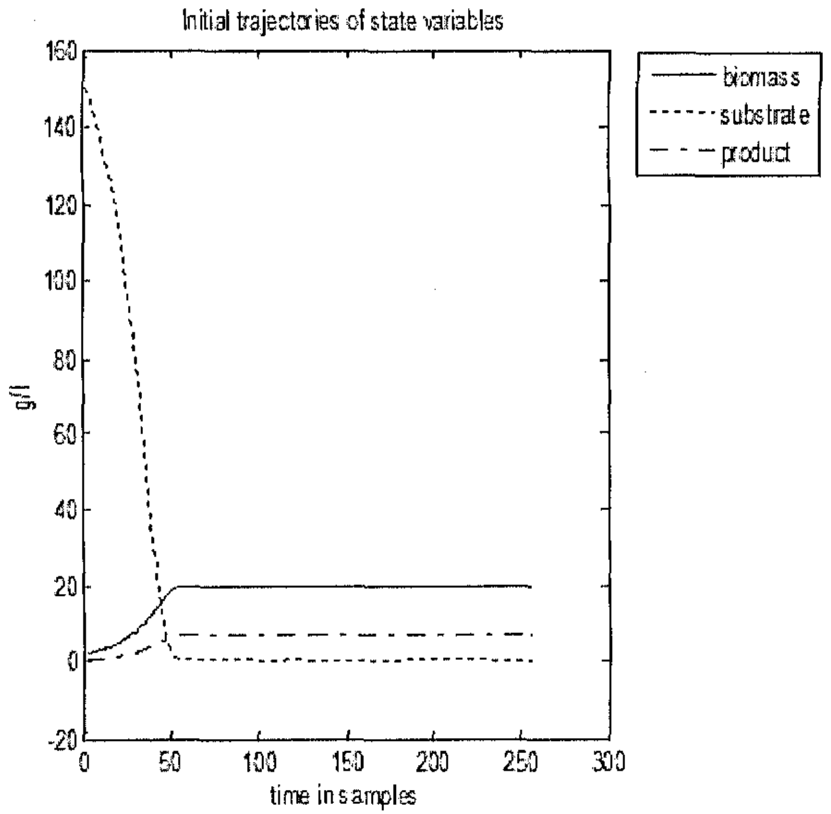


Figure 8.1: Initial trajectories of state variables for batch fermentation process

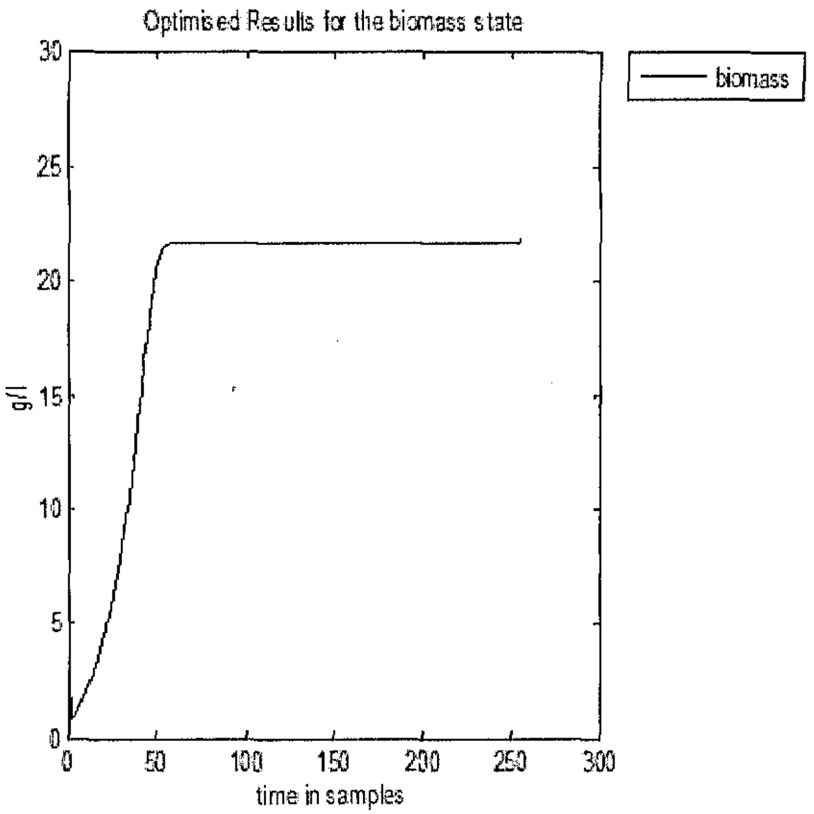


Figure 8.2: Optimal trajectory of biomass for batch fermentation process

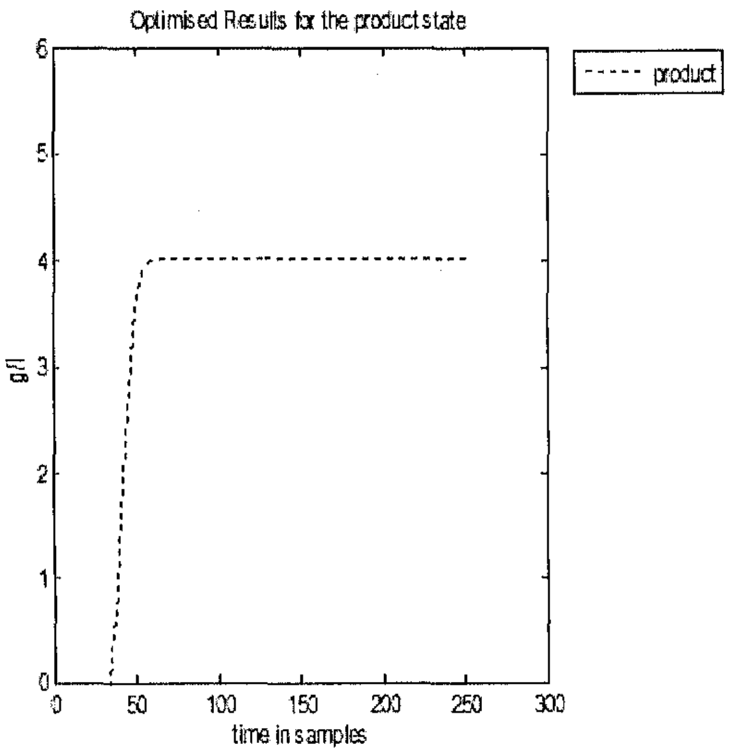


Figure8.3: Optimal trajectory of product for batch fermentation process

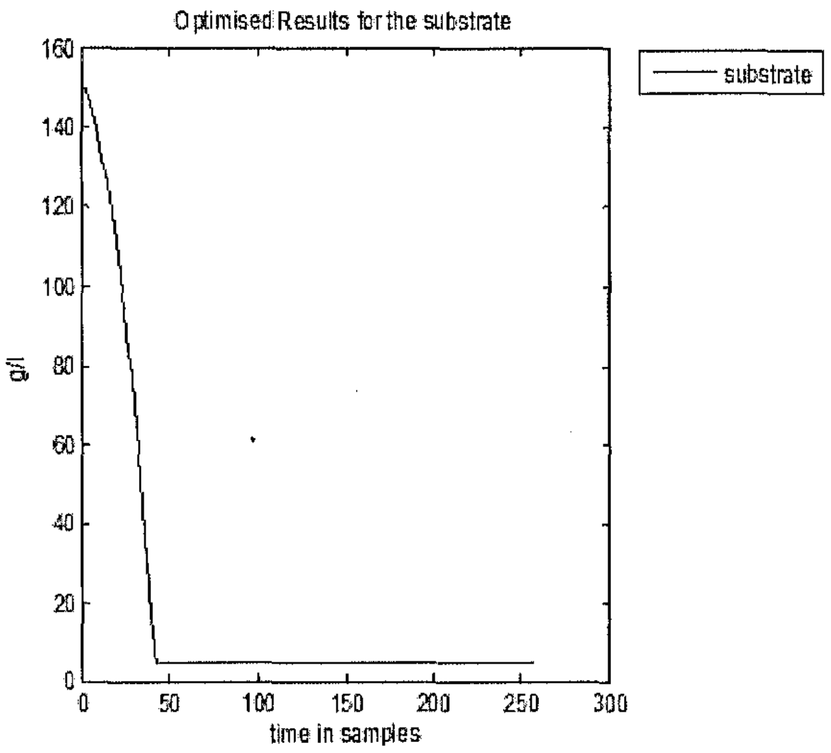


Figure 8.4: Optimal trajectory of substrate for batch fermentation process

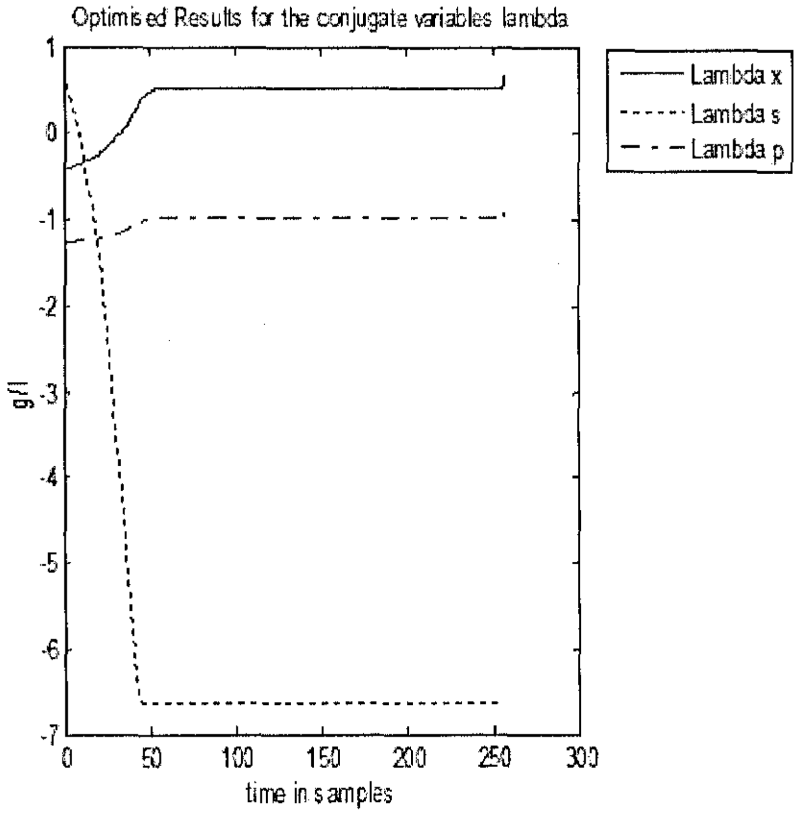


Figure 8.5: Optimal trajectories of conjugate variables for batch fermentation process

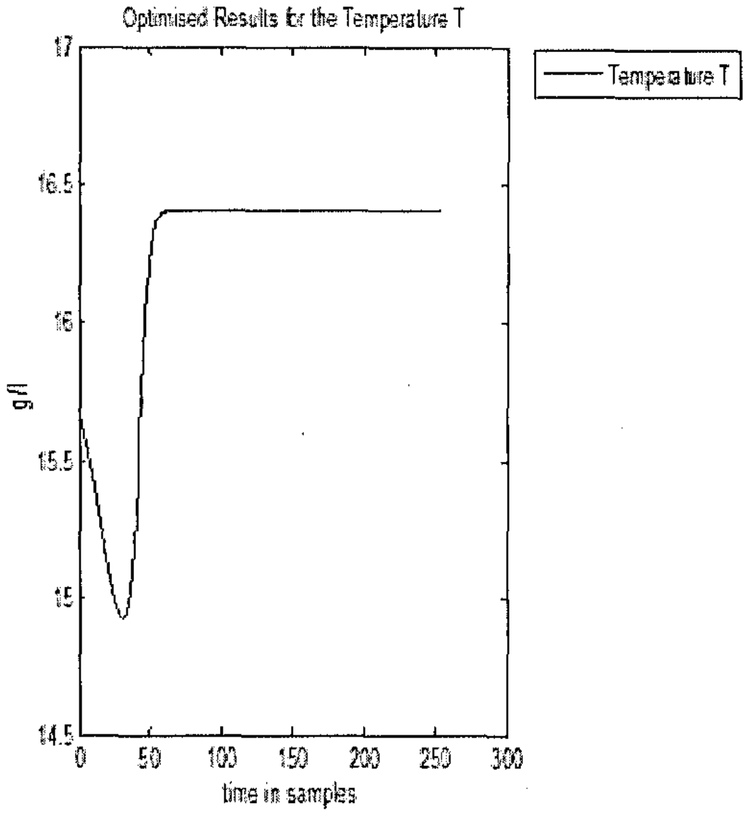


Figure 8.6: Optimal trajectory of temperature for batch fermentation process

8.3 Matlab parallel program for optimal control of fedbatch fermentation process

The program *fedbatch_parallel.m* transforms the sequential program for optimal control calculation of fed-batch process using special functions and commands from the Parallel Computing Toolbox. The inputs and outputs are the same as for the sequential program. The program *subprob2_fedbatch.m* is the program that is sent to the workers for parallel calculation of the optimal control and state trajectories with the *Filedependencies* function from the Parallel Computing Toolbox(PCT). The definition of variables in a Matlab script-file *batch_parallel.m* is as follows:

```
global par T F kd m pl n si K M1 eps1 mu l x s p v el conmin conmax statemin statemax astate acon dt y0 x1 s1 p1 v1 F1 T1.
```

This variables are defined global since they have to be accessible at a shared memory which in this case is used as jobmanager. They are defined global in the script *fedbatch_parallel* and *subprob2_fedbatch.m.*

8.3.1 Starting parallel computation for fed-batch fermentation process

The start of the coordinating procedure described in chapter4 is used for parallelization of the calculation of the state and control trajectories for parts of the optimization interval.

```
j=1
while j <= M2;   Second level iterations
jm = findResource('scheduler','configuration','jobmanagerconfig5')
pjob=createParallelJob(jm)
set(pjob,'MinimumNumberOfWorkers',numlabs)
set(pjob,'MaximumNumberOfWorkers',numlabs)
```

The function *subprog2_fedbatch.m* is on the client path, but it has to be accesible to the workers.The FileDependencies property of the *pjob* is used to transfer this function to all the workers that are available on the computer cluster .

```
set(pjob,'FileDepandencies',{'subprog2_fedbatch.m'})
```

Starting parallel computing is done by creating a task which is the same for every worker. The task has only one output with 6 output arguments and not input arguments to the function.

```
Task1=createTask(pjob, @subprob,6,{})
 submit(pjob)
```

The function *subprog2_fedbatch.m* is calculating the first level problem for an optimization interval with 256 periods, 32 machines, the time horizons for 8 steps and the last point for k = 257 is computed in the worker with index1. When the calculation completes, the output results are in the global variables x1, s1, p1,v1, T1,F1 and are transformed to the variables x, s, p,v, T,F by the worker with index1

```
waitForState(pjob,'finished')
```

```
results=getAllOutputArguments(pjob)
```

The results are used for calculation of the improved values of the coordinating variables, which again are used for calculation of state and control variables until the convergence of the coordination process is achieved. The parallel version of the program is given in appendix E.1.

8.3.2 Function for parallel calculation of fed-batch fermentation process

The part of the program implemented in a parallel way is described by the function file *subprog2.fedbatch.m*

```
function [x,s,p,v,T,F ]=subprog2_batch
```

The same global parameters are declared.

global par T F kd m pl n si K M1 eps1 mu l x s p v el conmin conmax statemin statemax astate acon dt y0 x1 s1 p1 v1 F1 T1.

The *subprog2_fedbatch.m* starts with calculation of the start*(a)* and stop*(b)* points of the optimization horizon for every worker. The number of points from the optimization interval for calculation of the optimal state and control variables is the same for every worker

```
dd=K/numlabs
a=(labindex-1)*dd+1
b=labindex*dd
```

Decomposition in time domain inside every subinterval is done

```
for k=a:b
i=1
```

The function determines the first level iterations number.

```
while i <= M1
```

Calculation of the expressions *lv(k), k=0,K-1, v = x, s, p,v* is followed by calculation of the errors and the values of the state and control variables. The iterations continue until the state and control gradients are less than the error eps1 or until the number of iterations becomes equal to M1. The program has to reach this point for every worker. The function *labBarrier* is used for this purpose:

labBarrier

The Parallel Computing Toolbox function *labBarrier* is used to synchronize the parallel calculations and to ensure that all workers have done the necessary calculations. Then the worker with *labindex* equal to 1 is forming the whole trajectory of state and control variables on the tasks of their global definition in the following way.

```
labBarrier

if labindex==1
x2=reshape(x1,1,numel(size(x1)));
s2=reshape(s1,1,numel(size(s1)));
p2=reshape(p1,1,numel(size(p1)));
v2=reshape(v1,1,numel(size(v1)))
T2=reshape(T1,1,numel(size(T1)))
F2=reshape(F1,1,numel(size(F1)));
x=x2([a:b,labindex])
s=s2([a:b,labindex])
p=p2([a:b,labindex])
v=v2([a:b,labindex])
T=T2([a:b,labindex])
F=F2([a:b,labindex])
```

The obtained trajectories are the output of the subprogram. They are used in the main program for calculation of the coordinating variables. The calculations stop when the number of iterations is reached or when the criteria for convergence of the coordinating procedure is satisfied. The program for calculation of the first-level subproblems in parallel is given in appendix E2.

8.3.3 Results from calculations with the Matlab parallel program for fed-batch process

The optimal trajectories of state and control variables are given in figures 8.7-8.14.

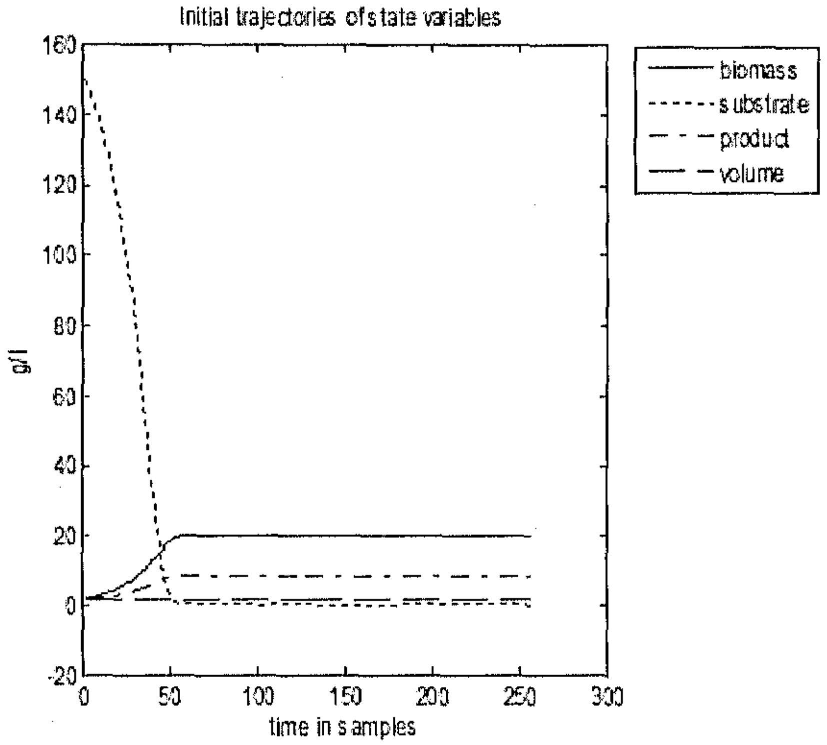


Figure 8.7: Initial trajectories of state variables for fed-batch fermentation process

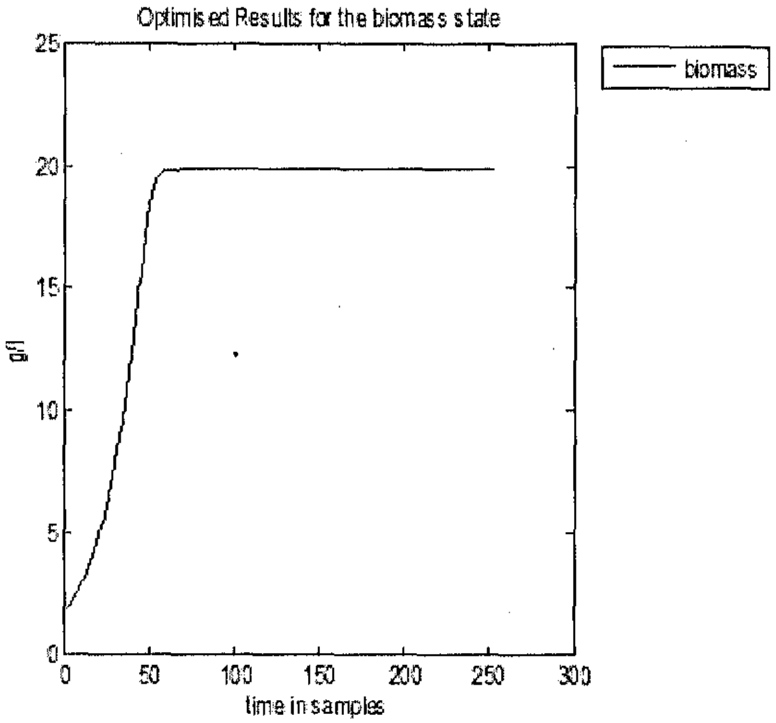


Figure 8.8: Optimal trajectory of biomass for fed-batch fermentation process

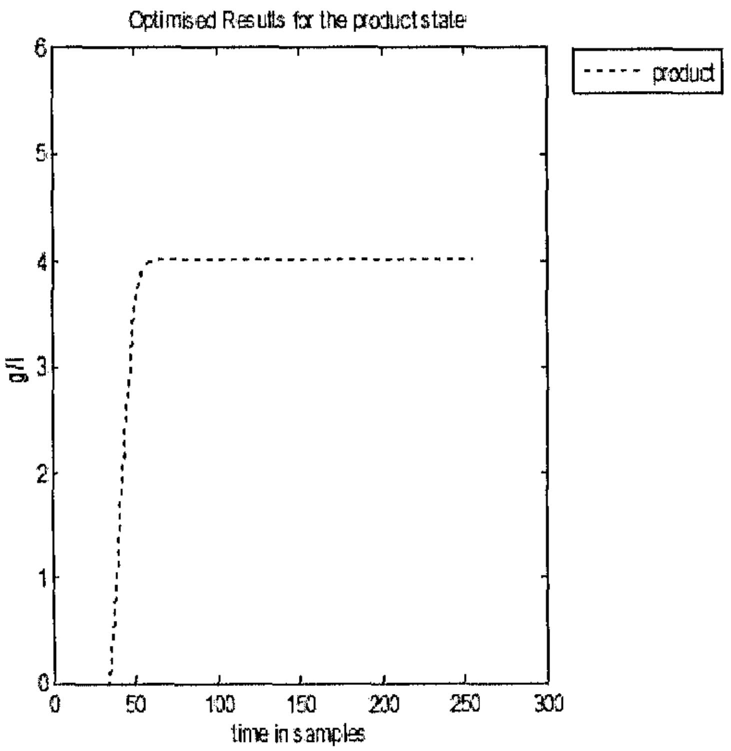


Figure 8.9: Optimal trajectory of product for fed-batch fermentation process

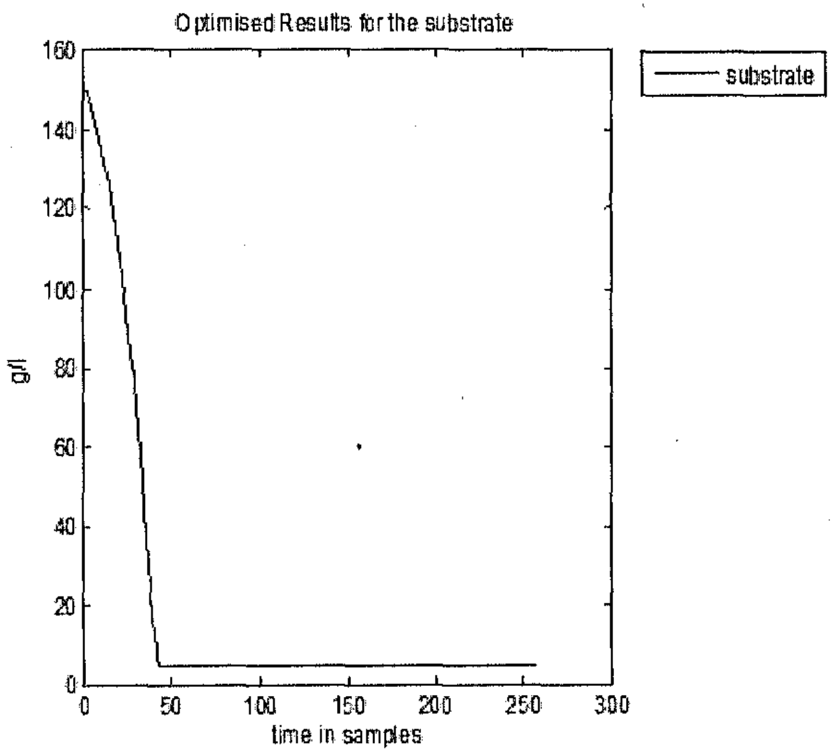


Figure 8.10: Optimal trajectory of substrate for fed-batch fermentation process

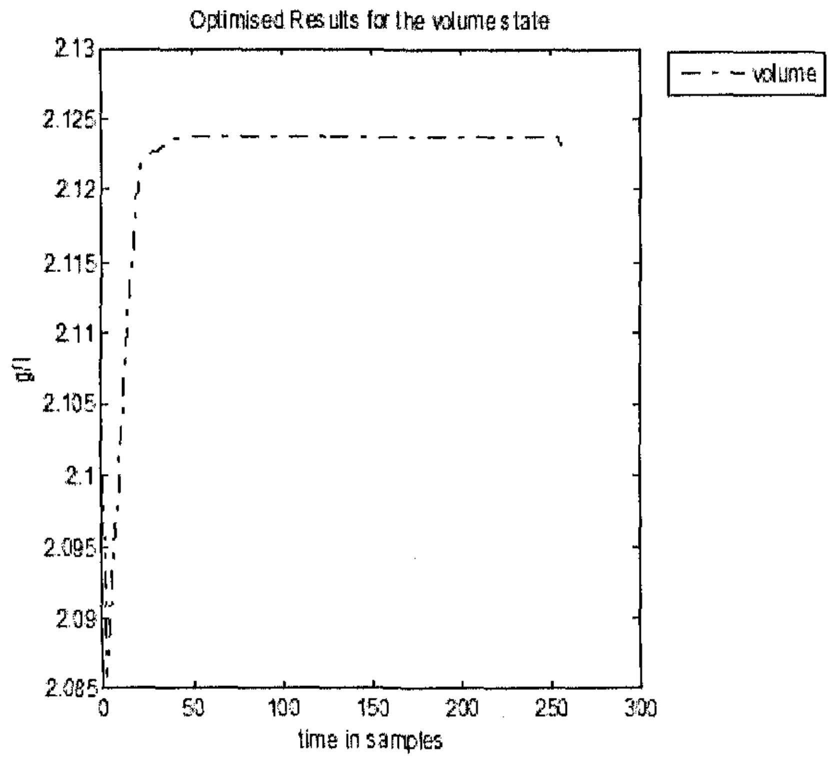


Figure 8.11: Optimal trajectory of volume for fed-batch fermentation process

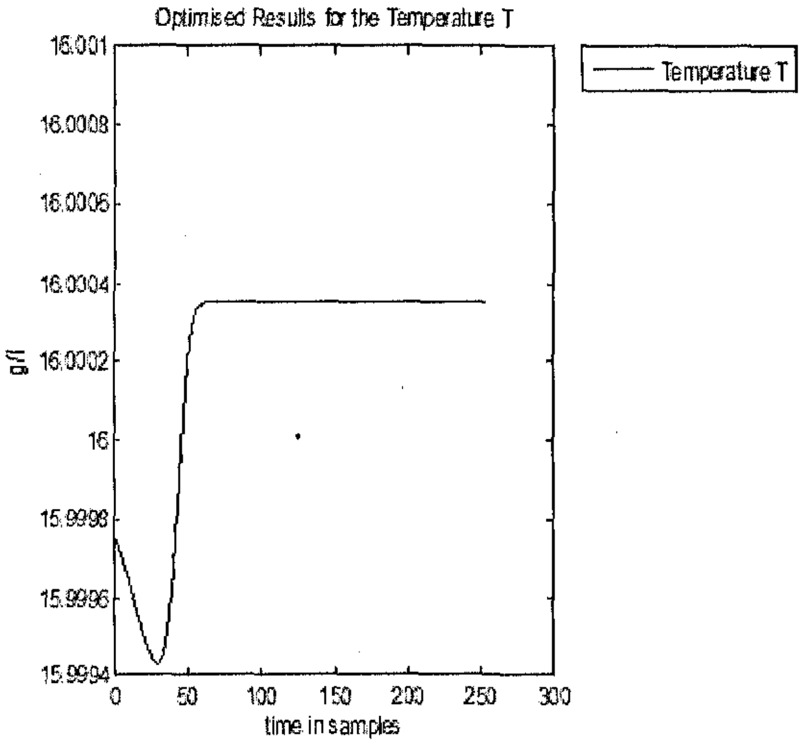


Figure 8.12: Optimal trajectory of temperature for fed-batch fermentation process

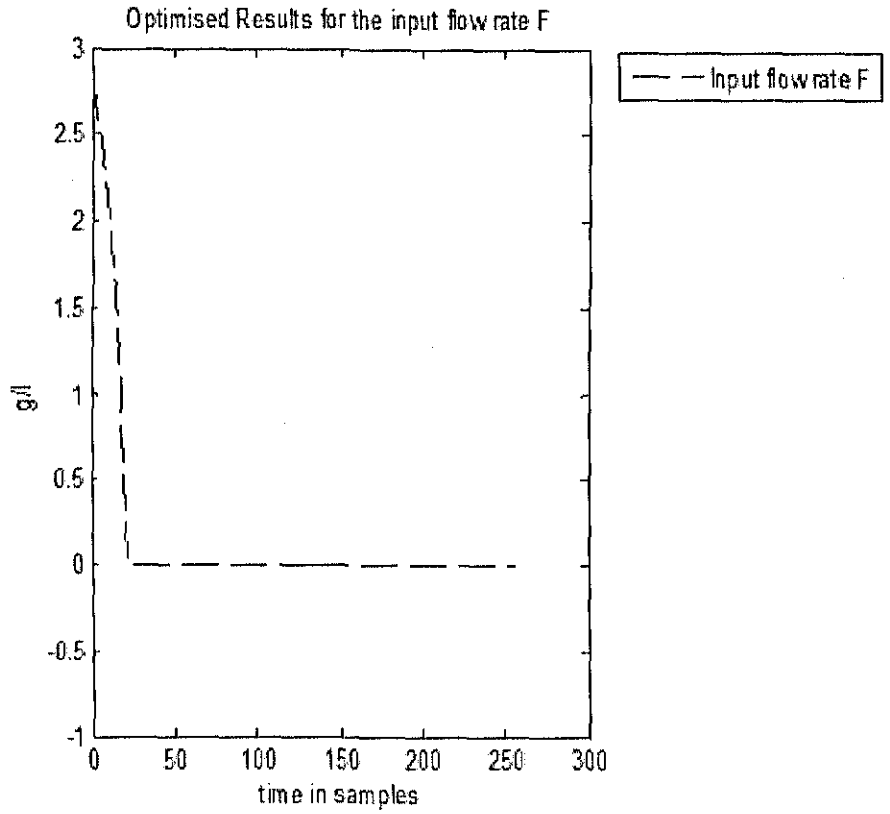


Figure 8.13: ***Optimal trajectory of input-flow rate for fed-batch fermentation process***

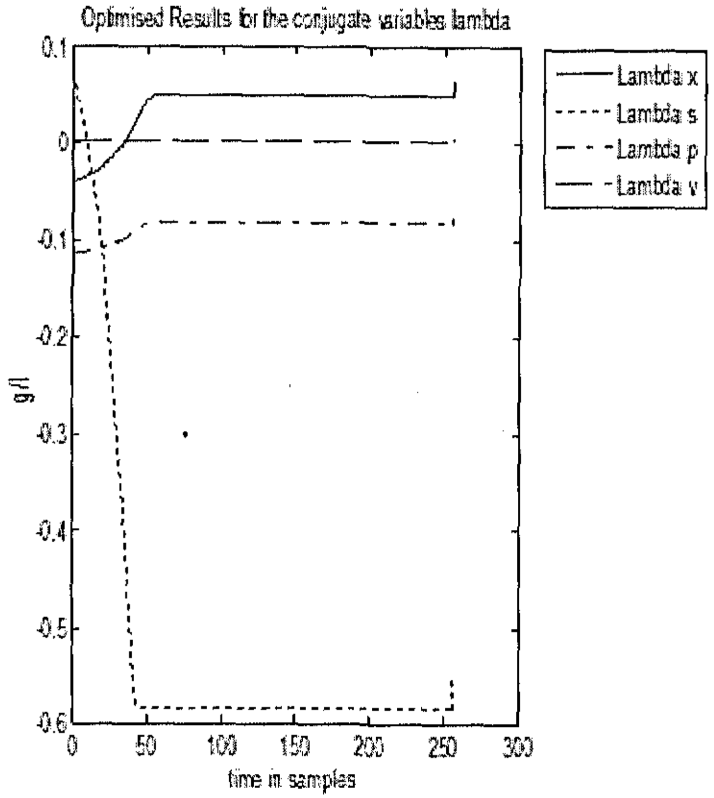


Figure 8.14: Optimal trajectories of conjugate variables for fed-batch fermentation process

8.4 Matlab parallel program for optimal control of continuous fermentation process

The program *continuous_parallel.m* transforms the sequential program for optimal control calculation of continuous process using special functions and commands from the Parallel Computing Toolbox. The inputs and outputs are the same as the sequential program. The program *subprob2_continuos.m* is the program that is sent to the workers for parallel calculation of the optimal control and state trajectories with the *Filedependencies* function from the Parallel Computing Toolbox(PCT). The definition of variables in a Matlab script-file *continuous_parallel.m* is as follows:

```
global par T D kd m pl n si K M1 eps1 mu l x s p el conmin conmax statemin statemax
astate acon dt y0 x1 s1 p1 D1 T1.
```

This variables are difined global since they have to be accessible at a shared memory which in this case is used as jobmanager. They are difined global in the script *continuous_parallel* and *subprob2_continuos.m.*

8.4.1 Starting parallel computation of continuous fermentation process

The start of the coordinating procedure described in chapter4 is used for parallelization of the calculation of the state and control trajectories for parts of the optimization interval.

```
j=1
while j <= M2;   Second level iterations
jm = findResource('scheduler','configuration','jobmanagerconfig5')
pjob=createParallelJob(jm)
set(pjob,'MinimumNumberOfWorkers',numlabs)
set(pjob,'MaximumNumberOfWorkers',numlabs)
```

The function *subprog2_continouos.m* is on the client path, but it has to be accesible to the workers.The FileDependencies property of the *pjob* is used to transfer this function all the workers that are available on the computer cluster .

```
set(pjob,'FileDepandencies',{'subpro2_continuous.m'})
```

Starting parallel computing is done by creating a task which is the same for every worker. The task has only one output with 5 output arguments and not input arguments to the function.

```
Task1=createTask(pjob, @subprob2_continuous,5,{})
submit(pjob)
```

The function *subprog2_continuous.m* is calculating the first level problem for an optimization interval with 256 periods, 32 machines, the time horizons for 8 steps and the last point for k = 257 is computed in the worker with index1. When the calculation completes, the output results are in the global variables x1, s1, p1, T1,D1 transformed to the variables x, s, p, T,D by the worker with index1

```
waitForState(pjob,'finished')
```

```
results=getAllOutputArguments(pjob)
```

The results are used for calculation of the improved values of the coordinating variables, which again are used for calculation of state and control variables until the convergence of the coordination process is achieved. The parallel version of the program is given in appendix D.1.

8.4.2 Function for parallel calculation of continuous fermentation process

The part of the program implemented in a parallel way is described by the function file *subprog2.continuous.m*

```
function [x,s,p,T,F ]=subprog2_continuous
```

The same global parameters are declared.

global par T D kd m pl n si K M1 eps1 mu l x s p el conmin conmax statemin statemax astate acon dt y0 x1 s1 p1 D1 T1.

The *subprog2_continouos.m* starts with calculation of the start and stop points of the optimization horizon for every worker.

```
dd=K/numlabs
a=(labindex-1)*dd+1
b=labindex*dd
```

Decomposition in time domain inside every subinterval is done

```
for k=a:b
i=1
```

The function determines the first level iterations number.

```
while i <= M1
```

Calculation of the expressions *lv(k), k=0,K-1, v = x, s, p* is followed by calculation of the errors and the values of the state and control variables. The iterations continue until the state and control gradients are less than the error eps1 or until the number of iterations becomes equal to M1. The program has to reach this point for every worker. The function *labBarrier* is used for this purpose:

labBarrier

The Parallel Computing Toolbox function *labBarrier* is used to synchronize the parallel calculations and to ensure that all workers have done the necessary calculations. Then the worker with labindex equal to 1 is forming the whole trajectory of state and control variables of the tasks using their global definition in the following way.

```
labBarrier

if labindex==1
x2=reshape(x1,1,numel(size(x1)));
s2=reshape(s1,1,numel(size(s1)));
p2=reshape(p1,1,numel(size(p1)));
T2=reshape(T1,1,numel(size(T1)))
D2=reshape(D1,1,numel(size(F1)));
x=x2([a:b,labindex])
s=s2([a:b,labindex])
p=p2([a:b,labindex])
T=T2([a:b,labindex])
D=D2([a:b,labindex])
```

The obtained trajectories are the output of the subprogram. They are used in the main program for calculation of the coordinating variables. The calculation stop when the number of iterations is reached or when the criteria for convergence of the coordinating procedure is satisfied. The program for calculation of the first-level sub-problem in parallel is given in appendix F2.

8.4.3 Results from calculations with the Matlab parallel program for continuous process

The optimal trajectories of state and control variables are given in figures 8.15-8.22.

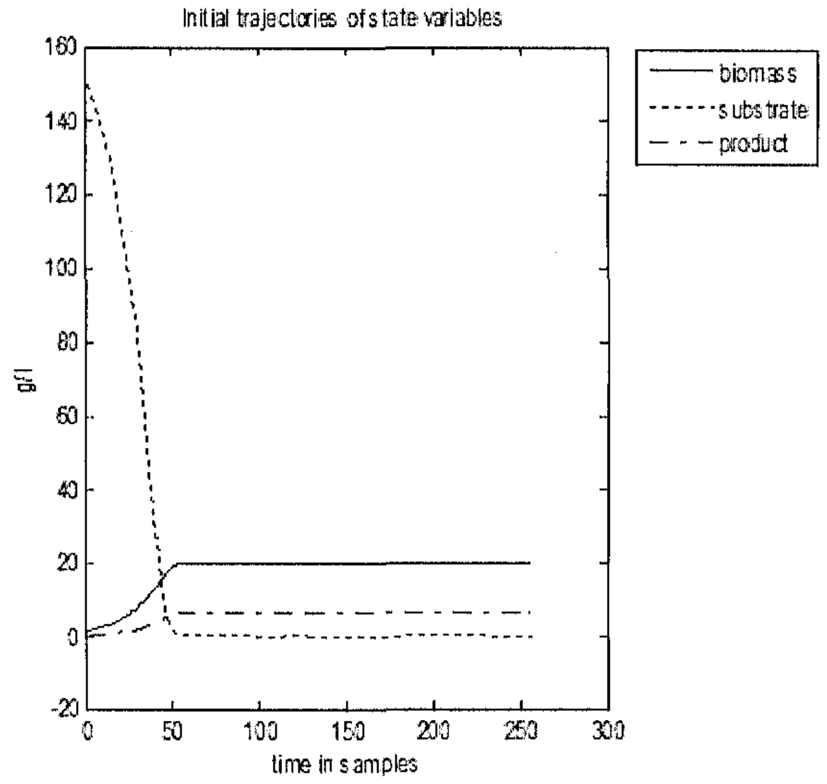


Figure 8.15: Initial trajectories of state variables for continuous fermentation process

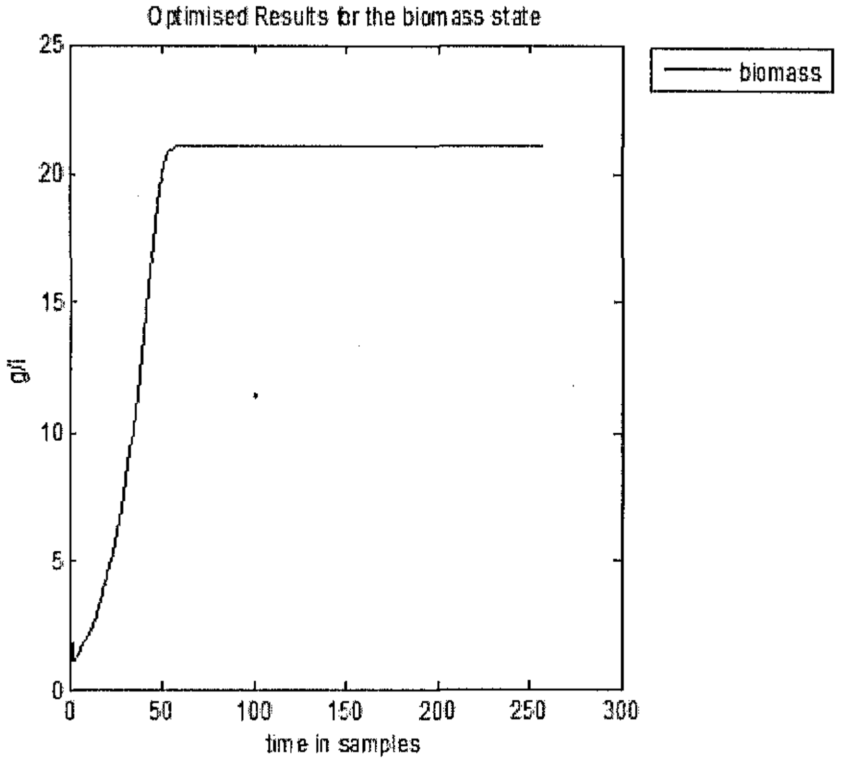


Figure 8.16: Optimal trajectory of biomass for continuous fermentation process

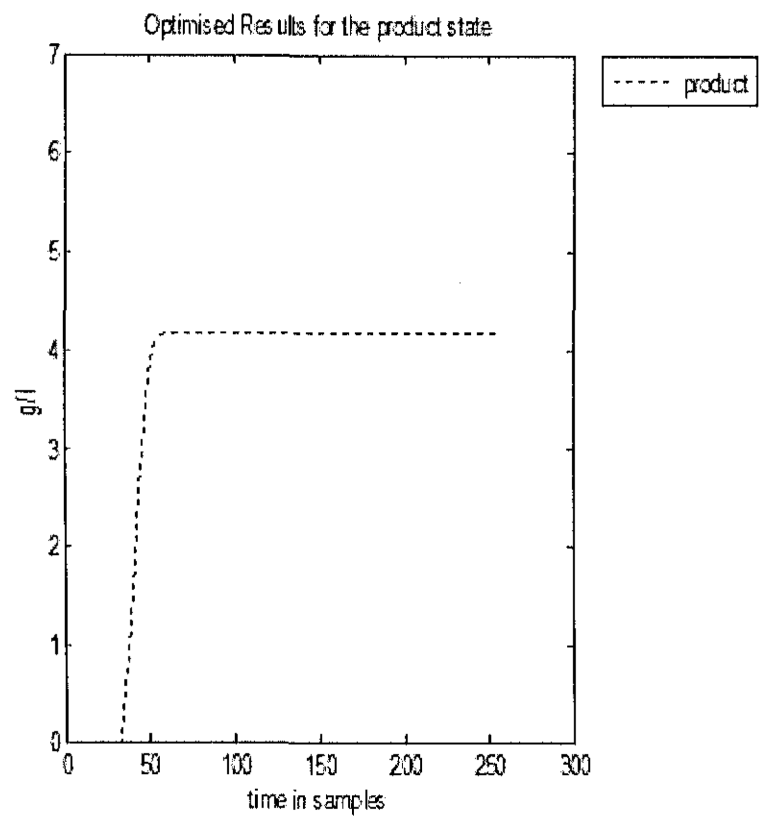


Figure 8.17: Initial trajectory of product for continuous fermentation process

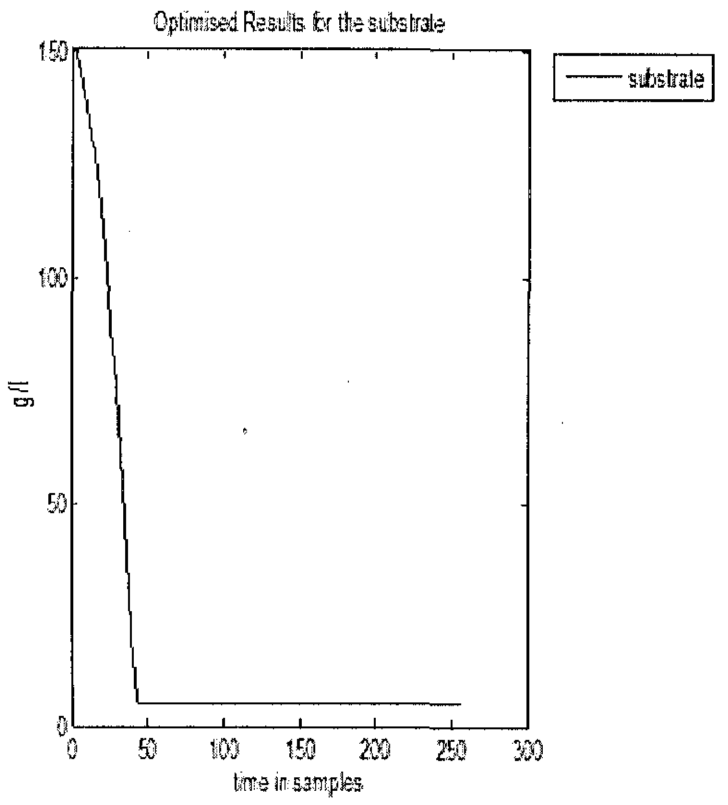


Figure 8.18: Optimal trajectory of substrate for continuous fermentation process

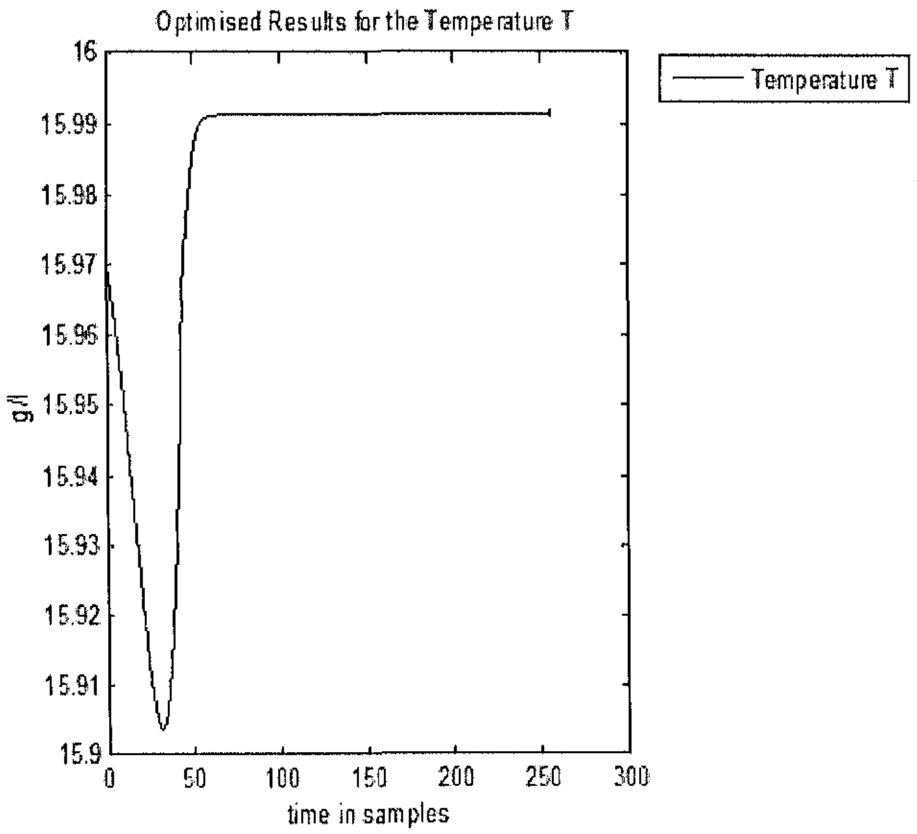


Figure 8.19: optimal trajectory of temperature for continuous fermentation process

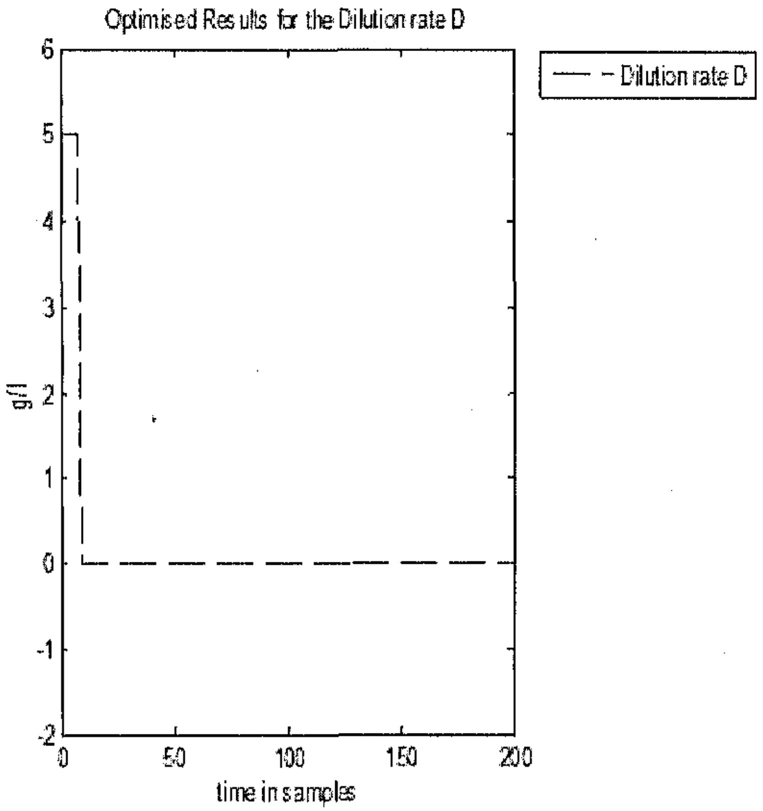


Figure 8.20: Optimal trajectory of Dilution rate for continuous fermentation process

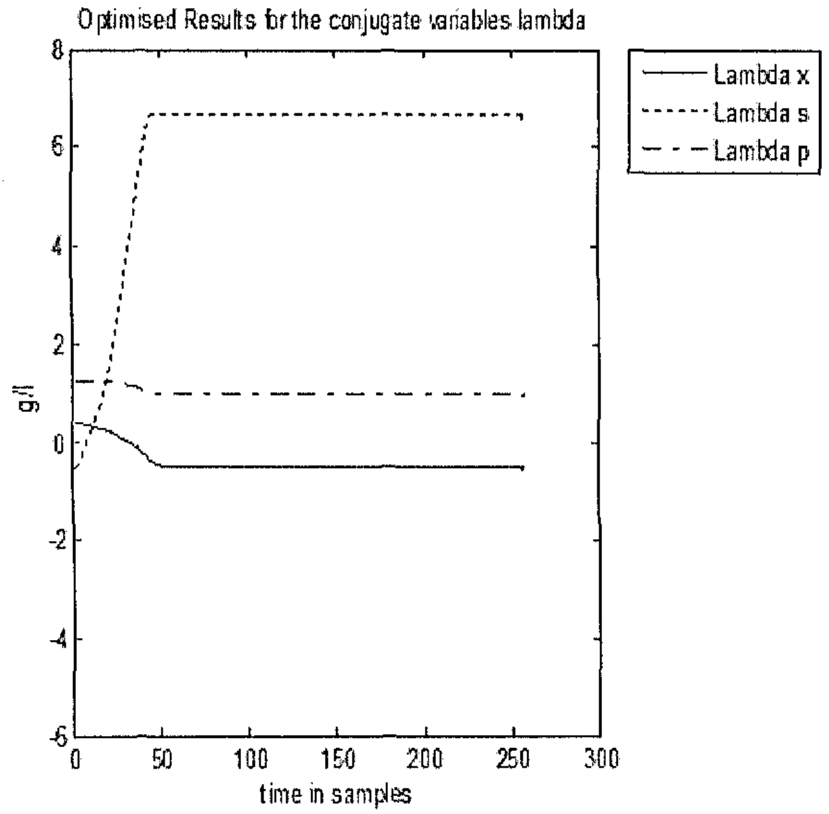


Figure 8.21: Optimal trajectories of conjugate variables for continuous fermentation process

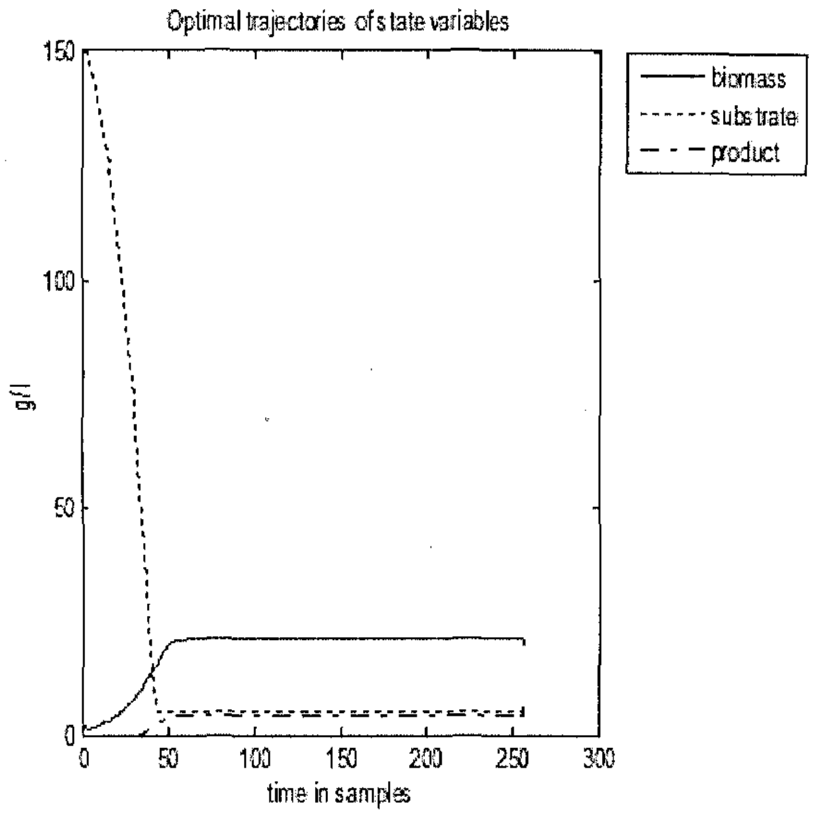


Figure 8.22: Optimal trajectories of state variables for continuous fermentation process

8.5 Comparison of the results from the sequential and parallel computing

The results of sequential program are compared with the results of parallel ones for calculation of optimal state and control of fermentation processes. The three fermentation processes are considered i.e. batch, fed-batch and continuous.

8.5.1 Comparison of the results from the sequential and parallel computing for batch fermentation process

The optimal state and control trajectories given from figure 4.3 -4.8 are compared with the optimal control and state trajectories of figure 8.1-8.6. It is clear that the results from the sequential calculation and the parallel calculation of optimal state and control trajectories are the same.

8.5.2 Comparison of the results from the sequential and parallel computing for fed-batch fermentation process

The optimal state and control trajectories given from figure 5.2 -5.9 are compared with the optimal control and state trajectories of figure 8.7-8.14. Again the results from the sequential calculation are the same with the ones from parallel calculation of optimal state and control trajectories.

8.5.3 Comparison of the results from the sequential and parallel computing for continuous fermentation process

The optimal state and control trajectories given from figure 6.2 -6.9 are compared with the optimal control and state trajectories of figure 8.15-8.22. It is clear that the results from the sequential calculation are the same with ones from the parallel calculation of optimal state and control trajectories.

8.6 Conclusion

This chapter describes the programmes for optimal control calculation in Parallel Computing Toolbox. The input and output variables of the programmes and the used functions and commands are described. The Results from calculations with the Matlab sequential program and the parallel calculations also are described. The results from the sequential and parallel computing are compared. Conclusion of the thesis is given in chapter nine.

CHAPTER NINE
CONCLUSION AND FUTURE RECOMMENDATIONS

9.1 Introduction

The purpose of this project is to develop methods, algorithms and programme for parallel solution of the problem for optimal control such that the process of calculation is characterized with reduced complexity and minimum time.

Fermentation, due to an inherent flexibility of the technology provides a valuable tool in biotechnology, not only to study microorganisms, but also to commercially produce valuable components. However, many requisites and variables should be taken into consideration in its implementation. Moreover, the control of the process is mandatory, which otherwise would render the mode of operation complexity useless. In addition to the complexity of the fermentation, the control mode and parameters to be controlled should also be analysed.

The control of fermentation processes can implicate many difficulties: low accuracy of on-line measurements of substrate concentrations, limited validity of the feed schedule under a variety of conditions and prediction of variations due to strain modification or change in the quality of the nutrient medium. These aspects point to the need of a fermentation process control strategy, which is model independent, identifies the optimal state on-line, incorporates a negative feedback control in the nutrients feeding system and contemplates saturation kinetic model, variable yield model, variations in feed substrate concentration and product inhibited fermentation. (Agrawal et al, 1994). The problem for formulation and solution of the typical optimal control problems of the fermentation processes using decomposition methods is considerd and solved in the thesis.

9.2 Aim and objectives

The research is focused on developing a decomposition method to solve the problem for optimal control of the fermentation processes for production of biomass and product. The project aim is divided into the following objectives:

- Developing a unique mathematical model of the fermentation processes by incorporating of physiochemical variables into the kinetic modes.

- Developing decomposition method for optimal control calculation of the three considered types of fermentation processes i.e. batch, fed-batch and continuous.
- Developing algorithms and programs in Matlab for sequential calculation of the optimal control problem.
- Developing algorithms and programs in Matlab for distributed-parallel calculation of the optimal trajectories.

9.3 Thesis delivarables

9.3.1 Development of the mass balance models for different types of fermentation processes

Based on mass balance principle, the models of batch ,fed-batch and continuous fermentation processes are derived in chapter two. The developed models incorporate the physiochemical variable temperature, in the biological mass balance equations. In this way: the control of the enzymes over the biological variables is utilised and opportunities for process optimisation are generated.

9.3.2 Development of a decomposition method for solution of the optimal control problem

The optimal control problems for the mass balance models are formulated based on the optimal control theory using criteria for maximum production of product for the batch, maximum biomass for the fed-batch process and minimum time for the continuous process.The method for solution is obtained based on the Lagrange's functional and decomposition coordination approach for time domain decomposition of the problem. Special coordinating vector is introduced in order to fully decompose the Lagrange's functional in time domain. . In this way the complexity of the problem is reduced and the solution of the nonlinear two-point boundary value problem is avoided.

9.3.3 Development of the algorithms and programs in Matlab for sequential calculation of the optimal control trajectories

The algorithms are developed for sequential calculation of control and state trajectories of the fermentation processes and are implemented in Matlab. The programs are given in appendix A,B and C and are presented in Table 9.1.

Table 9.1: Sequential programs for calculation of optimal state and control trajectories

Fermentation process	Main program	Sub-program	Time of calculation
Batch	Batch_seque.m	Modelbatch_func2.m	17.732429seconds
Fed-batch	Fedbatch_seque.m	Modelfedbatch_func2.m	585.295686seconds.
Continuous	Continuous_seque.m	Modelcontinuous_func2.m	27.945219 seconds.

9.3.4 Development of the algorithms and programs in Matlab for distributed-parallel calculation of the optimal control trajectories

Decomposition method allows naturally application of the parallel computing cluster of computers by which the sub-problems on the optimal control problem are solved in parallel on the first level of the two level calculating structure. The algorithms developed for parallel calculation of control and state trajectories of the fermentation processes are implemented in Matlab. The programs are given in appendix D,E and F and are presented in Table 9.2.

Table 9.2: Parallel programs for calculation of optimal state and control trajectories

Fermentation process	Main program	Parallel Sub-program	Model Sub-program	Time of calculation
Batch	batch_parallel.m	Subprob2_batch.m	Modelbatch_func2.m	17.732429seconds
Fed-batch	Fedbatch_parallel.m	Subprob2_fedbatch.m	Modelfedbatch_func2.m	585.295686seconds
Continuous	Continuous_parallel.m	Subprob2_continuous.m	Modelcontinuous_func2.m	27.945219 seconds.

9.4 Application of results

The developed approach for modeling and for optimal control problem solution can be applied with small modifications to some industrial processes with similar characteristics, as:

- Wasterwater purification plants.
- Beer industry.
- Food industry.
- Mining industry for extraction of metal ions.
- Sugar and pharmaceutical industry.
- Control purposes in education.
- As an application plant for different control strategies

9.5 Future research

The future developments can be decomposed in the following way:

- Testing of the two-layer control system on the working plant
- Testing of the optimisation program for different experiments
- Testing of the developed application on the bigger scale fermentor
- Application of the developed methods and programmes for different batch, fed-batch and continuous processes.

9.6 Publication

Mbangeni, L. & Tzoneva, R. 2010. Development of methods for parallel computation of the solution of the problem for optimal control of batch fermentation process. *Submitted to SAIEE research journal*

Mbangeni, L. & Tzoneva, R. 2010. Parallel computation of the problem for maximum production of biomass in a fed-batch fermentation process. *Submitted to control and automation journal*

Mbangeni, L. & Tzoneva, R. 2010. Development of a decomposition method and algorithm for parallel calculation of minimum time control of a continuous fermentation process using cluster of computers. *Submitted to SAIEE research journal*

BIBLIOGRAPHY

Agrawal P., Koshy G., Ramseier M., 1989: An algorithm for operating a fed-batch fermentor at optimum specific growth rate. *Biotechnology and Bioengineering*, vol. 33, pp. 115-125.

Ahmed s. Fawzy, Oliver r. Hinton, 1984: Optimal and hierarchical control of fermentation processes using a microcomputer: *A survey, IEEE, IEEE TRANS. SYST. Man Cyber. vol.* smc-14, no. 1, pp. 162-165.

Aiba, S. (1979). Review of process control and optimization in fermentation. *Biotechnology. Bioeng. Symp.* 9: pp269-281.

Angel L., Decegama. 1989: The technology of parallel processing, *NY Prentice Hall.*

Andrés, B. (1996). Modelling, Simulation, and Optimal Control of and Industrial Beer Fermentation Process. (inSpanish). Doctoral Thesis. *University Complutense of Madrid. Spain.*

Andres-Toro, B., Giron-Sierra, J.M., Lopez-Orozco, J.A., Fernandez-Conde, C.,(1997a). Optimization of Batch Fermentation process by Genetic Algorithms. *Proc. IEEE International Conference on System, Man, and Cybernetics, 'Computational Cybernetics and Simulation'*, p.392-397.

Banga J.R, Eva Balsa-Canto, Carmen G Moles and Alonso (2003). Dynamic Optimization of bioprocesses: *Review, process Engineering group*, pp257-265.

Banga J R, Seider W D (1996). Global optimization of chemical processes using stochastic algorithms. *In: State of the Art in Global Optimization, C.A. Floudas and P. M. Pardalos (eds.), Kluwer Academic Pub., Dordrecht, The Netherlands*, pp 563–583.

Banga, J. R., Alonso, A. A. and Singh, R. P. (1997), stochastic dynamic optimization of batch and semi continuous bioprocesses. *Biotechnology. Prog.* 13: pp326-335.

Banga, J.R., Alonso, A.A., Singh, R.P., 1994. Stochastic optimal control of fed-batch bioreactors. *Presented at AIChE Annual Meeting, San Francisco.*

Banga, J.R., Alonso, A.A., Moles, C.G., and Balsa-Canto, E. (2002). Efficient and robust numerical strategies for the optimal control of non-linear bio-processes. In *Proceedings of the Mediterranean Conference on Control and Automation (MED2002)*, (eds. J.S. Sentievio and M. Attrans), IEEE CSS, Lisbon, Portugal.

Beyer, H-G. (1996), "Toward a theory of evolution strategies: self-adaptation", *Evolutionary Computation*, Vol. 3 No.3, pp.311-47.

Beyer, H-G., Schwefel, H-P. (2002), "Evolution strategies", *Natural Computing*, Vol. 1 pp.3-52.

Bibila T.A and Robinson D.K (1995).Biotechnology programming 11.

Bishop, B.F. and S.J.Lorbert (1990). The needs for sensors in bacterial and yeast fermentations; *in Sensors in bioprocess control (Ed. Twork, J.V. and M.Yacynych), Marcel Dekker*, Inc., New York.

Brain Pumphrey, Christian Julien, 1996: An Introduction to fermentation.

Bryson A and Ho Y-C (1975). Applied Optimal control. *New York hemisphere publishing corporation.*

Carrasco E.F and Banga J.R (1997). Dynamic optimization of batch reactors using adaptive stochastic algorithms, *Industrial & engineering chemistry research*, pp2252-2261.

Casey, G.P. and W.M.Ingledew (1986). Ethanol tolerance in yeasts. *CRC Crit. Rev. Microbiol.*13: pp219-280.

Chancelier J-P, Renaud A (1994). Daily Generation Scheduling: Decomposition Methods to Solve the Hydraulic Problems, *International Journal of Electrical Power and Energy System*, 16, pp175-181.

Chang, Y.K. and H.C.Lim. (1990). Fast inferential adaptive optimization of a continuous yeast culture based on carbon dioxide evolution rate. *Biotechnology. Bioeng.* 35, pp8-14.

Chang, S.C., Chang, T.S., and LuH, P.B.,(1989). A hierarchical decomposition for large-scale optimal control problems with parallel processing structure. *Automatica,* Vol 25, pp77-86.

Charles F.Van Loan, 2000: Introduction to scientific computing, *NY Prentice Hall.*

Chen, C. and Hwang, C. (1990), Optimal control computation for differential-algebraic process systems with general constraints. *Chem. Eng. Comm.* 97, pp9-26.

Choy R, Edelman A (2003). Parallel MATLAB: Doing it Right, Computer Science AI laboratory, *Massachusetts Institute of Technology*, Clarke, D.J., Calder, M.R., Carr, R.J.G., Coleman, B.C.B. (1985). Moody, S.C. and T.A.Collinge, The development and application of biosensing devices for bioreactor monitoring and control. *Biosensors,* 1: pp 213-320.

Culler DE, Singh JP, Gupta A (1998). Parallel computer architecture: *a hardware/ software approach. Los Altos, CA: Morgan Kaufmann.*

Cuthrell, J.E. and Biegler, L.T. (1989). Simultaneous Optimization and Solution Methods for Batch Reactor Control Profiles. *Comput. Chem. Engng.*, 13, pp49-62.

Dadebo, S. A. and Mcauley, K. B. (1995), Dynamic optimization of constrained chemical engineering problems using dynamic programming. *Comp. Chem. Engng.* 19: pp513-525.

David B. Fogel, *Evolutionary Computation*: The Fossil Record, *Wiley-IEEE Press*, 1998.

D'Amore, T. and G.G.Stewart (1987). Ethanol tolerance of yeast, *Enzyme Microbiol. Technol.*, 9, pp322-330.

Dahhou B., Chamilothoris G. Roux (1991a): Adaptive predictive control of a continuous fermentation process. *International journal of Adaptive Control and Signal processing*, 18, vol.5, pp351-362.

Dairaku K., E. Izumuto, H. Mourikawa, S. Shioyta and T. Takamatsu, An advance micro computer control system in a baker's yeast fed-batch culture using a tubing method, *Journal of Fermentation Technology* **61** (1983) (2), pp189–196.

DARPA. High productivity computing systems (HPCS) program. http://www.highproductivity.org.

Dehdari P., A.Ohadi, L.Edara, K. Wesner, 2005: Fermentation technology Abstract.

DeRose, L., Gallivan, K., Gallopoulos, E., Marsolf, B., Padua, D(1995). FALCON: a MATLAB interactive restructuring compiler. *In: Languages and Compilers for Parallel Computing*, pp. 269–288. Springer-Verlag, New York

Dekkers, R.M. and M.H.Voetter (1986). Adaptive control of baker's yeast fermentation, *IFAC proceedings series*, 10, pp103-110.

Desineni Subbram Naidu (1985), Optimal Control System.

Dobry, D.D. and J.L.Jost (1977). Computer applications to fermentation operations; *In Annual Reports on Fermentation Processes (vol.1), (Ed.) D.Perlman, Academic Press, New York.*

Dochan D., Bastin G., (1990). Adaptive control of fed-batch bioreactor, *Chemical Engineering Comm,* vol 87, pp67-85.

Dongarra J. J., J. R. Bunch, C. B. Moler, and G. W. Stewart, 1992, *LINPACK User's Guide*. Philadelphia, PA: SIAM.

Di Pillo G., S. Lucidi. 2001. An augmented Lagrangian function with improved exactness properties. *SIAM J. Optimal.* 12, pp376-406.

El-Mansi E.M.T, Bryce C.F.A, Demain A.L and Allman A.R. (2006). Fermentation Microbiology and Biotechnology (second edition). *CRC Press, Taylor & Francis Group,* ISBN 0-8493-5334-3.

Engasser, J.M., Marc, I., Moll, M., Duteurtre, B. (1981). *Proc. EBC Congress*, pp579-583.

Engasser, J.M (1988). Bioreactor engineering: The design and optimization of reactors with living cells, *Chemical Engineering Science*, 43:pp1739-1748.

Estrada-Flores S., Merts I., De Ketelaere B., Lammertyn J., (2006). Development and validation of grey-box models for refrigeration applications: A review of key concepts. *International journal of refrigeration*, 29, pp931-946

Eungdamrong N., Iyengar R. (2004). Modeling cell signaling networks, *Biology of the cell*, 96, pp355-362.

Esposito W.R and Floudas (2000). Deterministic Global Optimization in Nonlinear Optimal Control Problems, *Department of Chemical Engineering* , Princeton University , Princeton , USA,17,pp97-126.

Eungdamrong N.J. and Iyengar R. (2004). Modeling Cell Signaling Networks: *Review, Department of Pharmacology and Biological Chemistry Mount Sinai School of Medicine* New York NY 10029 USA, Biology of the Cell 96, pp355–362.

Everett, H. (1963), Generalized Lagrange multiplier method for solving problems of optimum allocation of resources, *Operations Research* 11, pp399-417

Feng W, Manocha D (2007). High-performance computing using accelerators. *Parallel Computing* , vol 33:pp645-648.

Floudas C.A (2000). Deterministic global optimization: theory, Methods and Applications, *Kluwer Academic Publishers.*

Fogel L.J., Owens A.J., and Walsh M.J.(1966) Artificial Intelligence through Simulated Evolution, John Wiley, NY.

Galvanauskas V., Simutis R., Volk N. and Lubbert A., 1998. Model based design of a biochemical cultivation process, *Institut fur Bioverfahrenstechnik und*

Galvanauskas V., R. Simutis, N. Volk and A. Lübbert (1998). Model-based design of a biochemical cultivation process, *Bioprocess Eng.*, 18 pp227–234.

Gao W, Kemao Q, Wang H, Lin F and Soon Seah H(2009). Parallel computing for fringe pattern processing: A multicore CPU approach in MATLAB environment, *School of computer Engineering university of Singapore*

Gee, D.A., Fred Ramirez, W. (1994). "A Flavour Model for Beer Fermentation". *J. Inst. Brew.*, 100, pp321-329.

Gerson, D.F., Kole, M.M., Ozum, B. and M.N.Oguztoreli (1988). Substrate concentration control in bioreactors, *Biotechnology and Genetic Engineering Reviews*, 6:pp68-150.

Goesmann, A., Linke, B., Rupp, O., Krause, L., Bartels, D., Doudrup,

M., McHardy, A.C., Wilke, A., P¨uhler, A., Meyer, F., 2003. Building a BRIDGE for the integration of heterogeneous data from functional genomics into a platform for systems biology. *J.Biotechnol.* 106, pp157–167

Goldberg, D. E., (1989). "Genetic Algorithms in Search, Optimization and Machine Learning," *Addison-Wesley*

Grossmann I.E (1996). Global optimization in Engineering design, *Kluwer Academic* Dordrecht Boston.

Guus, C., Boender, E., and Romeijn, H.E. (1995). Stochastic methods. In *Handbook of global optimization* (eds. R. Horst and P.M. Pardalos), *Kluwer Academic Publishers*, Dordrecht, the Netherlands, pp829-869.

Guidoboni, G.E (1984). Continuous fermentation systems for alcohol production, *Enzyme Microb. Technol.* 6: pp194-200.

Hamer, IW. and C.B.Richenberg, (1988). On-line optimizing control of a packed-bed immobilized-cell reactor, *AIChE,* 34: pp626-632.

Haisong Chen, 2005: Methods and algorithm for optimal of fed batch fermentation process (master's dissertation).

Holland J.H (1975). Adaptation in natural and artificial systems: An introduction analysis with applications to biology, *control and artificial intelligence*, University of Michigan.

Hough, JS. Briggs, DE. Stevens.(1971). Malting and Brew.Sc.. Chapman & Hall.

Humphrey, A.E (1990). Problems and challenges in the production and processing of biologically active materials; in Frontiers *in Bio processing* (Ed. Sikdar, S.K., Bier, M. and P.Todd), *CRC Press*, Inc., Boca Raton, Florida.

Honda H. and T. Kobayashi (2004).Industrial application of fuzzy control in bioprocesses, *Adv. in Biochem. Eng. Biotech.* 87, pp151–171.

Hollingsworth, J., Liu, K., Pauca, P (1996).: Parallel Toolbox for Matlab PT v. 1.00: Manual and Reference Pages. Wake Forest University URL:

www.mthcsc.wfu.edu/pt/pt.html

Jayaraman, V.K., Kulkarni, B.D., Gupta, K., Rajesh, J. and Kusumaker, H.S., 2001. *Dynamic optimization of fed-batch bioreactors using the ant algorithm. Biotechnology Progress* 17, pp. 81–88.

James S., R. Legge and H. Budman (2002). Comparative study of black-box and hybrid estimation methods in fed-batch fermentation, *Journal of Process Control* 12 (1), pp113–121.

Jacobson, P (1992). Kågström, B., Rännar, M.: Algorithm development for distributed memory multicomputers using CONLAB. *Sci. Prog.* 1, pp185–203

Jorgensen K., Naes T. (2004). A design and analysis strategy for situations with uncontrolled raw material variation. *Jounarl of Chemometrics*, 18, pp45-52.

Jones D. M., Watton J., Brown K.J. (2007). Comparison of black, white and grey-box models to predict ultimate tensile strength of high-strength hot rolled coils at the port Talhot strip mills. Proceedings of the institution of Mechanical Engineers. *Jounarl of materials and design applications*, 221, pp1-9.

Johnson, A (1987). *The control of fed-batch fermentation processes-a survey, Automatica*, 23:pp691-705.

Jones, R.P (1989). Biological principles for the effects of ethanol, Enzyme Microb. *Techno.*11: pp130-153.

Julian E.H.(1989). Modelling of batch and fed-batch ethanol fermentation. Master thesis

Karakuzu C., Turker M. and Ozturk S (2006). Modeling, on-line state estimation and fuzzy control of production scale fed-batch baker's yeast fermentation, *control engineering practice* 14, pp959-974..

Karim, M.N. and G.Traugh (1987). Data acquisition and control of a continuous fermentation unit, *Journal of Industrial Microbiology*, 2, pp305-317.

Klein, M.T., Hou, G., Quann, R.J., Wei, W., Liao, K.H., Yang, R.S.H., Campain, J.A., Mazure, K.M.A., Broadbelt, L.J., 2002. BioMOL: a computer-assisted biological modeling tool for complex chemical mixtures and biological processes at the molecular level. Environ. *Health Perspect.* 110, pp1025–1029.

Kiviharju K. (2006). Optimization and modeling of bacterial processes; Dissertation for the degree of Doctor, *Department of Chemical Technology*, Helsinki University of Technology.

Kiviharju K., Salonen K, Leisola M. and Eekainen (2007). Kinetics of Bifidobacterium longum ATCC 15707 growth, Helsinki University of Technology, *Laboratory of Bioprocess Engineering, Process Biochemistry,* 42, pp1140–1145.

Kriz J. A(1983). Queuing analysis of a symmetric multiprocessor with shared memories and buses. *Proc IEE Computing Digital Tech IEEE*, vol 130: pp83–89.

Kronsjo T (1969). Decomposition of a Large Nonlinear Convex Separable Economic System in the Dual Direction. *Economics of Planning* 9: pp71-94.

Kronsjo, T (1969). Decomposition of a Nonlinear Convex Separable Economic System in Primal and Dual Directions. In: Optimization, Fletcher, R. (Ed.), *Academic Press*, London, pp85-97.

Lasdon L and Schoeffler J (1965). A Multi-Level Technique for Optimization. *Proceedings of the 1965 Joint Automatic Control Conference,* Troy, New York.

Lasdon L (1968). Duality and Decomposition in Mathematical Programming. *IEEE Transactions on Systems Science and Cybernetics*, 4: pp86-100.

Lasdon L. (1970). Optimization Theory of Large Systems, MacMillian, London.

Laluce, C. (1991).Crit. Rev. Biotechnology., 11: pp149-161.

Lee J. and Ramirez W.F., 1994. Optimal fed-batch control of induced foreign protein production by recombinant bacteria, *AIChE J.* 40, pp899–907.

Leigh, J.R. and M.Thoma (1986) Editorial: Control in bio processing, *IEE proceedings,* vol.133, pt-D, No.5:pp193.

Lemerle, C., Di Ventura, B., Serrano, L., 2005. Space as the final frontier in stochastic simulations of biological systems. *FEBS Lett.* 579, pp1789–1794.

Lim, H.C., Tayeb, Y.J., Modak, J.M. and Bonte, P., 1986. Computational algorithms for optimal feed rates for a class of fed-batch fermentation: Numerical results for penicillin and cell mass production. *Biotechnology and Bioengineering* 28, pp. 1408–1420.

Luus R., (1993). Optimal control by dynamic programming using systematic reduction in *grid size. pp995-1013.*

Luus, R., 1993. Application of dynamic programming to differential-algebraic process systems. *Computers and Chemical Engineering* **17** , pp. 373–377.

Luus R. and Hennessy D. (1999). Optimization of fed-batch reactors by the Luus-Jakola optimization procedure, *Ind. Eng. Chem. Res.* 38**,** pp1948–1955.

Maurer M., Rittmann B. (2004). Formulation of the CBC-model for modeling the contaminants and footprints in natural attenuation *of BTEX. Biodegration*, 15, pp419-434.

Maiorella, B.L (1985). Ethanol. In: M. Moo-Young (ed.), Comprehensive Biotechnology, vol.3. Pergamon, Oxford.

Maurer, M, Rittmann, B. E (2004). Formulation of the CBC-model for modeling the contaminants and footprints in natural attenuation of BTEX, *Biodegradation.* Vol 15. No 6. pp419-434.

Message Passing Interface Forum. MPI: A message-passing interface standard. Technical Report UT-CS-94- 230, 1994.

MATLAB Distributed Computing Engine® For Use with MATLAB, 2005

http://www.mathworks.com/access/helpdesk/help/pdf_doc/mdce/mdce.pdf .

https://tagteamdbserver.mathworks.com/ttserverroot/Download/29492_91263v01_DCT_datasheet.pdf .

Michael L. Shuler, Fikret Kargi (1992), Bioprocess Engineering Basic concepts.

Mitchel D., Von Meien O., Krieger N. And Dalsenter F.(2004). A review of recent developments in modeling of microbial growth kinetics and intraparticle phenomena in solide-state fermentation, *Biochem. Eng*, 17, pp15-17.

Michalewicz Z (1996). Genetic Algorithms + data structures = Evolution programs springler-verlag.

Michalewicz Z., Janikow C.Z. and Krawczyk J.B. (1992). A modified genetic algorithm for optimal control problems, *Computers Math. Applic.* **23**, pp83–94.

Mikler R, Kramer W, Doblholf-Dier O and Bayer K (1995). Strategies for optimal dissolved oxygen control. *Prepr of the 6th Int Conf on computer appl in Biotechnology*, pp315-318.

Mkondweni N.S and Tzoneva R, Development of a mathematic model of the fed batch fermentation process for the production of yeast. *Journal of Engineering design and Technology.*

Mkondweni N.S. Development of a mathematic model of the fed batch fermentation process for the production of yeast. (Masters Dissertation)

Montague, G.A. (1989).Morris, A.J.; Ward, A.C. Fermentation monitoring and control: a perspective. *Biotechnology and Genetic Engineering Reviews*, 7: pp147-188.

Moler, C (1995). Why isn't There a Parallel MATLAB? The *MathWorks Newsletter*, Spring

Murtagh, J.E (1986).*Process Biochem.* pp 61-65.

Nielsen J. and Villadsen J.(1994). Bioreaction engineering principles, Plenum press, New York.

Nielsen J., J. Villadsen (1992). Modeling of microbial kinetics, *chemEng. Sci.*, 47, pp4225-4270.

Njodzi Z., 2001: Studies on the fed-batch propagation of brewer's yeast in high gravity *wort*, (master's dissertation).

O'Callaghan D., Cunningham P., (2005). Modern process control techniques in the production of dried milk products-A review. Lait, 85, pp335-342

Ogata K., (1990): *Modern Control Engineering*, second edition, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Panuganti, R., Baskaran, M.M., Hudak, D., Krishnamurthy, A., Nieplocha, J., Rountev, A., et al (2006). GAMMA: Global Arrays Meets MATLAB. *Technical Report OSU-CISRC-1/06-TR15*, The Ohio State University, January

Papamichail I and Adjiman C.S (2002). A Rigorous Global Optimization Algorithm for Problems with Ordinary Differential Equations, *Centre for Process Systems Engineering, Department of Chemical Engineering and Chemical Technology*, Imperial College of Science, Technology and Medicine, London,24,pp1-33.

Park, S. and Ramirez, W.F., 1988. Optimal production of secreted protein in fed-batch reactors. *AIChE J.* **34** 9, pp. 1550–1558.

Pamment, N.B (1989). Overall kinetics and mathematical modelling of ethanol inhibition in yeasts; in Alcohol toxicity in yeasts and bacteria (Ed. van Uden, N.), *CRC Press, Boca Raton*, Florida.

Peter R. (2008): Predictive modeling of diary manufacturing processes, *International Diary Jounarl*, pp741-753.

Pollock, J.R.A. (1979). Brewing Science. *Academic Press*.

Pomerleau, Y. and M.Perrier (1990). Estimation of multiple growth rates in bioprocesses, AIChE, 36: pp207-215.

Pomerleau, Y. and M.Perrier (1992). Estimation of multiple specific growth rates: Experimental validation, *AIChE*, 38: pp1751-1760.

Popchev, T. & Tsoneva, R. 1990: Synthesis of Two-Layer Control of Interconnected Systems with Time Delays in the Presence of Slowly Varying Disturbance. *Proceedings for the, Tallinn: XI World Congress IFAC*, vol. 2, pp. 25-31.

Pushpavanam S, Rao S and Khan I (1999). Optimization of a biochemical fed-batch reactor using sequential quadratic programming " *Industrial & Engineering Chemistry Research*, 38, pp1998-2004.

Queinnec, I.; Dahhou, B.; Roux, G.; Goma, G (1991). Pourciel, J.B. Estimation and control of a continuous alcoholic fermentation process. J. *Ferment. Bioeng*, 72, pp285-290.

Rajkiran P., Muthu Manikandan B., Ashok K Nieplocha3 J., Rountev1 A and Sadayappan1 P (2007). *An Efficient Distributed Shared Memory Toolbox for MATLAB*

Reinikainen, P., Olkku, J., Markkanen, P. and A.Holmberg (1986). Determination of pH and temperature profiles for the growth of E.coli, *IFAC proceedings series*, 10: pp219-224.

Reaktionstechnik, Martin-Luther-Universitat, *Bioprocess Engineering*, 18, pp227-234.

Reuss, M. (1986). Computer control of bioreactors: Present limits and challenges of the future in Chemical Process Control, Ed. Morari, M. and T.J.McAvoy, *Elsevier*, Amsterdam.

Rhinehart, R.R., and J.B. Riggs, (1990). "Process Control through Nonlinear Modeling," Control, pp86-90.

Rodriguez-Acosta F., Regalado C.M. and Torres N.V. (1999). Nonlinear optimization of biotechnological processes by stochastic algorithms: application to the maximization of the production rate of ethanol, glycerol and carbohydrates by *Saccharomyces cerevisiae*, *J. Biotechnology.* 68, pp15–28.

Rolf, M.J. and H.C.Lim, (1982).Computer control of fermentation processes, Enzyme Microb.*Technol* 4: pp370-380.

Rolf, M.J. and H.C.Lim (1985). Experimental adaptive on-line optimization of cellular productivity of a continuous baker's yeast culture, *Biotechnology. Bioeng.* 27: pp1236-1245.

Rolf, M.J. and H.C.Lim (1984). Adaptive on-line optimization for continuous bioreactors, *Chem Eng. Commun.* 29: pp229-255.

Robert M, Paindavoine M, Gorria P (1992). *An edge detection ASIC for real time defect* detection. In: *Proceedings of Fifth annual IEEE international ASIC conference and exhibit.* pp193–196 (IEEE)

Rolf, M.J. and H.C.Lim (1985a). Experimental adaptive on-line optimization of cellular productivity of a continuous baker's yeast culture, *Biotechnology. Bioeng.* 27: pp1236-1245.

Ryhiner, G., Dunn, IJ. Heinzle, E. and S.Rohani (1992). Adaptive on-line optimal control of bioreactors: application to anaerobic degradation, *Journal of Biotechnology*, 22: pp89-106.

Sarkar D. and Modak J.M (2004). Optimization of fed-batch bioreactors using genetic algorithms: multiple control variableas, *Computer and chemical Engineering* 28, pp789-798.

Schwefel, H-P. (1995), *Evolution and Optimum Seeking*, Wiley, New York, NY.

Schoeffler J (1971). Static Multi-Level Systems. In: Optimization Methods for Large-Scale Systems with Applications, Wismer, D.A. (Ed.), McGraw-Hill, New York.

Scragg, A.H (1988) Biotechnology; in Biotechnology for engineers (Ed. Scragg, A.H.) Ellis Horwood Limited, West Sussex, U.K.

Seywald H., Kumar R.R. and Deshpande S.M. (1995).Genetic Algorithm Approach for Optimal Control Problems with Linearly Appearing Controls, *J. of Guidance, Control, and Dynamics* 18 (1) pp177--182.

Semones, G.E. and H.C.Lim (1989). Experimental multivariable adaptive optimization of the steady-state cellular productivity of a continuous baker's yeast culture, Biotechnology. Bioeng. 33, pp16-25.

Shama, G (1988). Developments in bioreactors for fuel ethanol production, *Process Biochemistry*, page138.

Sharma G, Martin J(2008). Matlab: A Language for Parallel Computing, *The Mathworks Ltd*, Cambridge Business Park, pp37:3-36

Shen, G., Cordeiro, O. and Vinson D.(1995). Model identification of batch processes for process control. *Proc of the American control conference*,3, pp1751-1755.

Shi, Z., Shimizu, K., Watanabe, N. and T.Kobayashi (1989). Adaptive on-line optimizing control of bioreactor systems, *Biotechnology. Bioeng.* 33: pp999-1009.

Singer, *A.B*, *Bok*, *J.K* and *Barton*, *P.I* (*2001*). Convex Underestimators for Variational and. Optimal Control Problems. *Computer Aided Chemical Engineering* 9, pp767-772.

Simutis, R. and Lübbert, A., 1997. A comparative study on random search algorithms for biotechnical process optimization. *J. Biotechnology.* **52**, pp. 245–25.

Smith F.G, Dimenna R.A (2004). Simulation of the batch chemical process using parallel computing with PVM and Speedup. *Computers and Chemical Engineering*, Savannah River Technology Center, Westinghouse Savannah River ,pp1649-1659.

Sonnleitner B., Kappeli, O. (1986). "Growth of Saccharomyces Cerevisiae Is Controlled by Its Limited Respiratory Capacity: Formulation and Verification of a Hypothesis". *Biotechnology. Bioeng.* 28, pp927-937.

Springham, D.G. (1991). The established industries; in Biotechnology: the science and the business, *Harwood Academic Publishers*, New York

Staniskis, J.(1984). Optimal control of biotechnological processes. Vilnjus: Mocslas, Russia.

Tang, J., LuH, P. B., and Change, T.S.,(1991). A parallel algorithm for long horizon optimal control problems using the mixed coordination method. *International Journal of control*, vol 53, pp1395-1412

Tamura, H. 1975. Decentralized Optimization for Distributed Lag Models of Discrete Systems, *Automatica*, vol. 11, pp. 593-602.

Tebbani S., Dumur D. and Hafidi G (2008). Open-loop optimization and trajectory tracking of the fed-batch bioreactor, *chemical Engineering Processing* 47, pp1933-1941

Tenney, R.I. (1985). " Rationale of the Brewery Fermentation". *ASBC* J., 43, 2, pp57-60.

Tholudur and W.F. Ramirez, Optimization of fed-batch biorectors using neural network parameter function models, *Biotechnology. Prog.* **12** (1996), pp. 302–309

Torn A, Ali M and Viitanen S (1999). A Numerical Comparison of Some Modified Controlled Random Search Algorithms, *Department of Computer Science*, Åbo Akademi University, Finland, 11, pp377-385.

Tzoneva, T. Patarinska and I. Popchev. 1998: Optimal control of continuous fermentation processes, Decomposition methods for start-up, steady-state and minimum time problem. *Bioprocess Eng* 13, 189-196.

Vassiliadis V.S (1993). Computational solution of dynamic optimization with general differential algebraic constraints, PhD thesis Imperial College University of London.

Villadsen, J (1989). Simulation of biochemical reactions, *Computers chem. Engng.* 13: pp 385-395

Vigie, P., Goma, G., Renaud, P.Y., Chamilothoris, G (1990). Dahhou, B. and J.B.Pourciel. Adaptive predictive control of a multistage fermentation process, *Biotechnology. Bioeng.* 35, pp217-223.

Vincent Y.B, Julie B.L, Margaret M.W, and Georges M.F(2005). Augmented Lagrangian coordination for decomposed design problems. *6th World Congresses of structural and Multidisciplinary optimization*, department of Mechanical Engineering Clemson

University.

Volesky B. (1992). Modeling and Optimization of Fermentation Processes. *ISBN* 0444895884.

Wang, N.S. and G.Stephanopoulos (1984). A new approach to bioprocess identification and modelling, *Biotechnology. Bioeng. Symp.* 14, pp635-656.

Warren, R.K., Hill, G.A. and D.G.Macdonald (1992). Simulation and optimization of a cell recycle system, Applied Biochemistry and Biotechnology, 34/35: pp585-595.

Warren, R.K (1992). Cell recycle of S.cerevisiae for ethanol fermentation, Ph.D. Thesis, University of Saskatchewan, Canada.

Weigand, W.A (1978). Computer applications to fermentation processes; In Annual Reports on Fermentation Processes (Ed. D.Perlman) Academic Press, New York.

Williams, D. (1990). Problems of measurement and control in biotechnological processes; in Chemical engineering problems in biotechnology (E. Winkler, M.A.), Elsevier Science Publishers Ltd., Essex, U.K.

Williams, D., Yousefpour, P. and B.H.Swanik (1984). On-line adaptive control of a fermentation process. IEE proceedings. vol.131, pp117-124.

Winkler, M.A (1988). Optimization and time-profiling in fermentation processes, in: (Ed.) M.A.Bushell, Progress in Industrial Microbiology. *Elsevier Science Publishers*, Netherlands.

Wu, W.T., Chen, K.C. and H.W.Chiou (1985). On-line optimal control for fed-batch culture of baker's yeast production, *Biotechnology. Bioeng.* 27: pp756-760.

Xiaohong G and Peter B.L(1992). The new parallel algorithm for optimal control problems of interconnected systems, *International Journal*, vol 56 no 6,pp1275-1297.

Yamashita, Y. and M. Shima (1997).Numerical computational method using genetic algorithm for the optimal control problem with terminal constraints and free parameters," Nonlinear Analysis, 30, pp2285-2290.

Zhang Q, Berry M.W, Lamb B.T and Samuel T(2009). Parallel Nonnegative Tensor Factorization Algorithm for Mining Global Climate data. *Proceedings of the international Conference on Computational Science(ICCS 2009)* GeoComputing workshop, Baton Rouge, LA, Lecture Notes in Computer(LNCS) 5545, Berlin,pp405-415

Zhang, H. and Lennox, B. (2003). Integrated condition monitoring and control of fed-batch fermentation processes, *Journal of Process Control*, **14 (1)**, pp41-pp50

Zhihua .Xiong and Jie Zhang (2004). Modeling and optimal control of fed-batch processes using a novel control affine feed-forward neural network, Centre for process Analytics and control technology, school of *Chemical Engineering and advanced materials*, university of Newcastle, Neurocomputing,61, pp317-337.

APPENDICES

APPENDIX A: MATLAB SEQUENTIAL PROGRAM FOR CALCULATION OF CONTROL AND STATE FOR BATCH FERMENTATION PROCESS

A.1: MATLAB script file - batch_seque.m

```
% M-File: batch_seque.m
%==========================M-FILE DESCRIPTION========================
% Calculation of state and control trajectories for batch fermentation
% process

clear all% clear workspace
tic; % start the clock
%========================= Set Initial values========================

M1 = 100; M2 = 50; %Max number of iterations of the first and second
levels
K = 256; %Number of steps in optimization horizon
ax = 0.001; as = 0.001; ap = 0.001; %Steps of the gradient procedures
for calculation of the State variables
aT = 0.05; %Steps of the gradient procedures for calculation of the
Control variables
alx = 0.001; als = 0.001; alp = 0.001; %Step of the gradient procedures
for calculation of the Conjugate variables
adx = 0.001; ads = 0.001; adp = 0.001; %Steps of the gradient procedures
for calculation of the Interconnections in time
eps1=0.001; %tolerances for stop of the calculation of the first level
eps2=0.001; %tolerances for stop of the calculation of the second levels
%===================================================================

%===========Initial values of the penalty coefficients==============
mx=0.0010;
ms=0.0010;
mp=0.0010;
%===================================================================

%=====Initial trajectory values of the conjugate variables==========
lx = zeros (1, K);
lp = zeros (1, K);
ls = zeros (1, K);
for k = 1: K
        lx (k) =0.001;
        ls (k) =0.005;
        lp (k) =0.002;
end
%===================================================================

%====Initial trajectories of the control variables for calculation of===
T=16.0;
%===================================================================
======================Model co-efficients==========================
a1 = 0.0004; a2 = 0.0001; a3 = 0.0002;
b1 = 0.2; b2 = 0.03; b3 = 0.01;
c1 = 2; c2 = 0.3; c3 = 0.021;
d1 = 0.0040; d2 = 0.006; d3 = 0.001;
par = [a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3];
kd = 0.00008; % death constant
pl = 50; % inhibition factor
m = 0.11; % Maintenance co-efficient
n = 0.52; % Power co-efficient
si= 266.6; % Input substrate
dt = 0.25; % delta t
%=======================================================================

%================Initial values of state variables=====================
x0=1.83;
s0=150.0;
p0= 0;
y0=[x0 s0 p0]';
%=======================================================================

%==============Constraints of control variable=========================
Tmin = 10;
Tmax = 20;
%=======================================================================
===
%=============== Constraints of state variables=======================
xmin = 0.83;
xmax =100;
smin = 5;
smax = 300;
pmin = 0;
pmax = 100;
%=======================================================================

%=====Calculation of the initial trajectories of state variables=======
t=1: K+1;
[t, y] =ode15s (@ (t, y) modelbatch_func2 (t, y, par T, kd, m, pl, n,
si), t, y0);
x=(y (:, 1))';
s=(y(:,2))';
p=(y(:,3))';
figure(1)
plot(t,x,'-k',t,s,':k',t,p,'-.k'),
title('Initial trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','Location','NorthEastOutside')
%=======================================================================

%Calculation of the initial trajectories of the interconnections in time
domain================================================================
for k = 1:K
    dx(k) = 10.0;
    ds(k) = 138;
    dp(k) = 25.0;
end
```

```
%=======================================================================
%====Initial trajectories of the control variables for calculation of===
T=16.0*ones(1,K);
%=======================================================================

%====Calculation of the initial trajectories of Error_Lambda_v_(k)======
 for k=1:K
    elx(k) = dx(k)-x(k+1);
    els(k) = ds(k)-s(k+1);
    elp(k) = dp(k)-p(k+1);
 end
%=======================================================================

%================START OF THE COORDINATING PROCEDURE===================
j=1;
while j <= M2;%Second level iterations

for k=1:K %Decomposition in time domain
    i=1;
while i <= M1 %First level iterations

%=========PreAllocation of variables for faster execution==============
dmmaxdT = zeros(1,K);
dKsdT =  zeros(1,K);
dYxsdT = zeros(1,K);
dYpxdT = zeros(1,K);
 mmax  = zeros(1,K);
 Ks    = zeros(1,K);
 Yxs   = zeros(1,K);
 Ypx   = zeros(1,K);
 ff    = zeros(1,K);
 fx    = zeros(1,K);
 fs    = zeros(1,K);
 fp    = zeros(1,K);
 f1    = zeros(1,K);
 f2    = zeros(1,K);
 eex   = zeros(1,K);
 ees   = zeros(1,K);
 eep   = zeros(1,K);
 ex    = zeros(1,K);
 es    = zeros(1,K);
 ep    = zeros(1,K);
 eT    = zeros(1,K);
%=======================================================================

%CALCULATION OF THE DERIVATIVES OF THE KINETIC COEFFICIENTS TOWARDS THE
CONTROL VARIABLES

%=================For the temperature==================================
dmmaxdT(k) = par(2)+2*par(3)*T(k); %Equations 5.10
dKsdT(k)=par(5)+2*par(6)*T(k);
dYxsdT(k)=par(8)+2*par(9)*T(k);
dYpxdT(k)=par(11)+2*par(12)*T(k);
%=======================================================================

%============Calculation of the kinetic oefficients====================
mmax(k)=par(1)+par(2)*T(k)+par(3)*(T(k)^2); % Equation 2.72
```

```
Ks(k)=par(4)+par(5)*T(k)+par(6)*(T(k)^2);    % Equation 2.73
Yxs(k)=par(7)+par(8)*T(k)+par(9)*(T(k)^2);   % Equation 2.74
Ypx(k)=par(10)+par(11)*T(k)+par(12)*(T(k)^2); % Equation 2.75
%==================================================================


%=============Calculation of the functions fx,fs,fp,f1 and f2==========
pk=p(k);
ff(k)=dt*mmax(k)  *((1-pk/pl)^n);
fx(k)=ff(k)*s(k)/(Ks(k) + s(k));     % Equation 4.20
fs(k)=ff(k)*Ks(k)  * x(k)/((Ks(k)  + s(k))^2); %Equation 4.22
fp(k)=dt*mmax(k)*((1-p(k)/pl)^(n-1))*x(k)*n*s(k)/(pl*(Ks(k)  + s(k))); %
Equation 4.21
ff1(k)=ff(k)*x(k)*s(k)/(Ks(k) + s(k)); %Equation 4.23
f1(k) = ff1(k)/mmax(k)
f2(k)=ff1(k)/(Ks(k) + s(k)); %Equation 4.24
%==================================================================

%==Calculation of the expressions ee_v_(k)=lambda(k)m_v*E_lambda_v_(k)==
eex(k)=lx(k)-mx*elx(k);
ees(k)=ls(k)-ms*els(k);
eep(k)=lp(k)-mp*elp(k);
%==================================================================
%========Calculation of the gradients for state variables=============
ex(k)=(1+fx(k)-dt*kd)*eex(k)-((fx(k)/Yxs(k))+ dt*m)*ees(k)+ fx(k)*Ypx(k)
* eep(k);%Equation 4.25
es(k)=fs(k)*eex(k)+(1-fs(k)/Yxs(k))*ees(k)+Ypx(k)*
f1(k)*eep(k);%Equation 4.26
ep(k)=-fp(k)*eex(k)+(fp(k)/Yxs(k))*ees(k)+(1+fp(k))*eep(k);%Equation
4.27
%==================================================================

%========Calculation of the gradients for control variables===========
eT(k) = (f1(k)*dmmaxdT(k)+f2(k)*dKsdT(k))*eex(k) + ((-
f1(k)/(Yxs(k))^2)*(dmmaxdT(k)*Yxs(k)-mmax(k)*dYxsdT(k))+
(f2(k)/Yxs(k))*dKsdT(k)) *ees(k) +
(f1(k)*(dYpxdT(k)*mmax(k)+Ypx(k)*dmmaxdT(k))-
Ypx(k)*f2(k)*dKsdT(k))*eep(k);
%Equation 4.28
n1=[ex(k) es(k) ep(k) eT(k)]';
%==================================================================

%================Calculation norm of gradients========================
nn = norm(n1);
%Check the condition for end of the iterations on the first level
if nn <=eps1
break
%==================================================================
%====Calculation of the new values of the state and control
variables=====
elseif k == 1
        x(k)=x0;
        s(k)=s0;
        p(k)=p0;
        T(k) = T(k) + aT*eT(k);
else
    x(k) = x(k) + ax*ex(k);
    s(k) = s(k) + as*es(k);
    p(k) = p(k) + ap*ep(k);
    T(k) = T(k) + aT*eT(k);
end
%=====================================================================
%========================Check against constraints===================
if x(k) < xmin
    x(k) = xmin;
elseif x(k) > xmax
    x(k) = xmax;
end
if s(k) < smin
    s(k) = smin;
elseif s(k) > smax
    s(k) = smax;
end
if p(k) < pmin
    p(k) = pmin;
elseif p(k) > pmax
    p(k) = pmax;
end
if T(k) < Tmin
    T(k) = Tmin;
elseif T(k) > Tmax
   T(k) = Tmax;
end
%=====================================================================

i=i+1;
end %End of the iterations on the first level
end %end of the iteration for k=1,K

%CALCULATION ON THE SECOND LEVEL
%Calculation of the final value of x(k)=x(K+1)=xK1;sK1,pK1

xK1=x(K)+f1(K)-dt*kd*x(K);
sK1=s(K)-f1(K)/Yxs(K)-dt*m*x(K);
pK1=p(K)+f1(K)*Ypx(K);
if xK1 <= xmin
    xK1 = xmin;
elseif xK1 >=xmax
    xK1 = xmax;
end
if sK1 <= smin
    sK1 = smin;
elseif sK1 >= smax
    sK1 = smax;
end
if pK1 <= pmin
    pK1 = pmin;
elseif pK1 >= pmax
    pK1=pmax;
end
x(K+1)=xK1;
p(K+1)=pK1;
s(K+1)=sK1;
%=====================================================================
%============Calculation of the gradient elv, where v = x,s,p=========
for k=1:K
    elx(k) = dx(k) - x(k+1);
    els(k) = ds(k) - s(k+1);
    elp(k) = dp(k) - p(k+1);
end
%====================================================================

nn1=norm(elx)+ norm(els)+ norm(elp);

if nn1<=eps2
    break

else
%====Calculation of the coordinating variables lv and dv, where v =
x,s,p==
for k =1:K
    lx(k) = lx(k) - alx*elx(k);
    ls(k) = ls(k) - als*els(k);
    lp(k) = lp(k) - alp*elp(k);

    edx(k) = -lx(k)-mx*elx(k);
    eds(k) = -ls(k)-ms*els(k);
    edp(k) = -lp(k)-mp*elp(k);

    dx(k) = dx(k) + adx*edx(k);
    ds(k) = ds(k) + ads*eds(k);
    dp(k) = dp(k) + adp*edp(k);
end
%====================================================================
===
end% End for the norm
j = j+1;
end% End of second level iteration
toc% End the clock

%==============================Plot Results===========================
figure(2)
t=1:K+1;
plot(t,x,'-k')
title('Optimised Results for the biomass state')
ylabel('g/l')
xlabel('time in samples')
legend('biomass','Location','NorthEastOutside')
figure(3)
plot(t,p,':k')
title('Optimised Results for the product state')
ylabel('g/l')
xlabel('time in samples')
legend('product','Location','NorthEastOutside')
figure(4)
title('Optimised Results for the substrate')
ylabel('g/l')
xlabel('time in samples')
plot(s,'-k')
```

```
legend('substrate','Location','NorthEastOutside')
t1=1:K;
figure(5)
plot(t1,T,'-k')
title('Optimised Results for the Temperature T')
ylabel('g/l');
xlabel('time in samples');
legend('Temperature T','Location','NorthEastOutside')
figure(6)
plot(t1,lx,'-k',t1,ls,':k',t1,lp,'-.k')
title('Optimised Results for the conjugate variables lambda')
ylabel('g/l');
xlabel('time in samples');
legend('Lambda x','Lambda s','Lambda p','Location','NorthEastOutside')
%==========================================================================
```

A.2: MATLAB script file – modelbatch_func2.m

```
M-File: modelbatch_func2.m
%==========================M-FILE DESCRIPTION ============================
%The function is used to calculate the trajectories of the derivatives
of the state space vector

function ydot= modelbatch_func2 (t, y, par, T, kd, m, pl, n, si)

%=========================Model_variables==================================

ydot(1)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))-(kd*y(1)));
%Equation 2.78
ydot(2)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))/-
(par(7)+par(8)*T+par(9)*(T*T))-m*y(1));%Equation 2.79
ydot(3)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T))+y(2))*(par(10)+p
ar(11)*T+par(12)*(T*T));%Equation 2.80
ydot= [ydot (1) ydot (2) ydot (3)]';
%==========================================================================
```

APPENDIX B: MATLAB SEQUENTIAL PROGRAM FOR CALCULATION OF CONTROL AND STATE FOR FED-BATCH FERMENTATION PROCESS

B.1: MATLAB script file - fedbatch_seque.m

```
% M-File: fedbatch_seque.m
%========================M-FILE DESCRIPTION ==========================
%Calculation   of   state   and   control   trajectories   for   fed-batch
fermentation
% process
clear all% clear workspace
tic; % start the clock
%=====================Set Initial values==============================
M1 = 30; M2 = 10; %Max number of iterations of the first and second
levels
K = 256; %Number of steps in optimization horizon
ax = 0.001; as = 0.001; ap = 0.001; av = 0.001; %Steps of the gradient
procedures for calculation of the State variables
aT = 0.05; aF = 0.05; %Steps of the gradient procedures for calculation
of the Control variables
alx = 0.001; als = 0.001; alp = 0.001; alv = 0.001; %Step of the
gradient procedures for calculation of the Conjugate variables
adx = 0.001; ads = 0.001; adp = 0.001; adv = 0.001; %Steps of the
gradient procedures for calculation of the Interconnections in time
eps1=0.001; %tolerances for stop of the calculation of the first levels
eps2=0.001; %tolerances for stop of the calculation of the second levels
%====================================================================



%===========Initial values of the penalty coefficients================
mx=0.0010;
ms=0.0010;
mp=0.0010;
mv=0.0010;
%====================================================================

%========Initial trajectories for the conjugate variables==============
for k = 1: K
        lx (k) =0.001;
        ls (k) =0.005;
        lp (k) =0.002;
        lv (k) =0.003;
 end
%====================================================================

%=============Initial trajectories of the control variables============
T=16.0;
F=0;
%====================================================================

%========================Model coefficients==========================
a1 = 0.0004; a2 = 0.0001; a3 = 0.0002;
b1 = 0.2; b2 = 0.03; b3 = 0.01;
c1 = 2; c2 = 0.3; c3 = 0.021;
d1 = 0.0040; d2 = 0.006; d3 = 0.001;
```

```
par = [a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3];
kd = 0.00008; % death constant
pl = 50; % inhibition factor
m = 0.11; % Maintenance co-efficient
n = 0.52; % Power co-efficient
si= 266.6; % Input substrate
dt = 2.5; % delta t
%==========================================================================

%===================tial values of state variables===================
x0=1.83;
s0=150.0;
p0=2.1;
v0=2.1;
y0=[x0 s0 p0 v0]';
%==========================================================================
====================Constraints of control variables===================
Tmin = 10;
Fmin = 0;
Tmax = 20;
Fmax = 5.0;
%==========================================================================

==================Constraints of State variables=====================
vmin = 2;
vmax = 4;
pmin = 2.0;
pmax = 50;
smin = 20;
smax = 300;
xmin = 0.83;
xmax =55;
%==========================================================================
=======Calculation of the initial trajectories of state variables=======
t=1: K+1;
[t, y] =ode15s(@(t,y) modelfedbatch_func2(t,y, par, T,F, kd, m, pl, n,
si),t,y0);
x=(y (:, 1))';
s=(y(:,2))';
p=(y(:,3))';
v=(y(:,4))';
figure(1)
plot(t,x,'-k',t,s,':k',t,p,'-.k',t,v,'--k'),
title('Initial trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','volume','Location','NorthEastOut
side')
%==========================================================================

% Calculation of the initial trajectories of the interconnections in
time
domain===================================================================
===
for k = 1:K
dx(k) = 10.0;
ds(k) = 138;
```

```
dp(k) = 25.0;
dv(k) = 2.0;
end
%======================================================================
%============Initial trajectories of the control variables============
T=16.0*ones (1, K);
F=5.0*ones (1, K);
%======================================================================
%=======Calculation of the initial trajectories of Error_Lambda_v_(k)===
 for k=1:K
 elx(k) = dx(k)-x(k+1);
 els(k) = ds(k)-s(k+1);
 elp(k) = dp(k)-p(k+1);
 elv(k) = dv(k)-v(k+1);
 end
%======================================================================

%==================START OF THE COORDINATING PROCEDURE=================
j=1;
while j <= M2;%Second level iterations

for k=1:K %Decomposition in time domain
i=1;
while i <= M1 %First level iterations

%=========PreAllocation of variables for faster execution==============
dmmaxdT = zeros(1,K);
dKsdT =  zeros(1,K);
dYxsdT = zeros(1,K);
dYpxdT = zeros(1,K);
 mmax   = zeros(1,K);
 Ks     = zeros(1,K);
 Yxs    = zeros(1,K);
 Ypx    = zeros(1,K);
 ff     = zeros(1,K);
 fx     = zeros(1,K);
 fs     = zeros(1,K);
 fp     = zeros(1,K);
 f1     = zeros(1,K);
 f2     = zeros(1,K);
 eex    = zeros(1,K);
 ees    = zeros(1,K);
 eep    = zeros(1,K);
 ex     = zeros(1,K);
 es     = zeros(1,K);
 ep     = zeros(1,K);
 eT     = zeros(1,K);
 eF     = zeros(1,K);
%======================================================================
%CALCULATION OF THE DERIVATIVES OF THE KINETIC COEFFICIENTS TOWARDS THE
CONTROL VARIABLES

%=====================For the temperature=============================
dmmaxdT(k) = par(2)+2*par(3)*T(k); %Equations 5.10
dKsdT(k)=par(5)+2*par(6)*T(k);
dYxsdT(k)=par(8)+2*par(9)*T(k);
dYpxdT(k)=par(11)+2*par(12)*T(k);
```

```
%================================================================

%==============Calculation of the kinetic coefficients================
mmax(k)=par(1)+par(2)*T(k)+par(3)*(T(k)^2); % Equation 2.72
Ks(k)=par(4)+par(5)*T(k)+par(6)*(T(k)^2);    % Equation 2.73
Yxs(k)=par(7)+par(8)*T(k)+par(9)*(T(k)^2);  % Equation 2.74
Ypx(k)=par(10)+par(11)*T(k)+par(12)*(T(k)^2); % Equation 2.75
%================================================================

%======Calculation of the expressions fv where v = x,s,p,v and f1 and
f2===
pk=p(k);
ff(k)=dt*mmax(k) *((1-pk/pl)^n);
fx(k)=ff(k)*s(k)/(Ks(k) + s(k));  % Equation 5.25
fs(k)=ff(k)*Ks(k) * x(k)/((Ks(k) + s(k))^2); %Equation 5.27
fp(k)=dt*mmax(k)*((1-p(k)/pl)^(n-1))*x(k)*n*s(k)/(pl*(Ks(k)  + s(k))); %
Equation 5.26
fv(k)=dt*F(k)/(v(k)^2); %Equation 5.29
ff1(k)=ff(k)*x(k)*s(k)/(Ks(k) + s(k)); %Equation 5.28
f2(k)=ff1(k)/(Ks(k) + s(k)); % Equation 5.30
f1(k) = ff1(k)/mmax(k)
%================================================================

%=Calculation of the expressions ee_v_(k)=lambda(k)m_v*E_lambda_v_(k)===
eex(k)=lx(k)-mx*elx(k);
ees(k)=ls(k)-ms*els(k);
eep(k)=lp(k)-mp*elp(k);
eev(k)=lv(k)-mv*elv(k);
%================================================================

%==========Calculation of the gradients for state variables===========
ex(k)=(1+fx(k)-dt*kd-dt*F(k)/v(k))*eex(k)+((fx(k)/Yxs(k))- dt*m)*ees(k)+
fx(k)*Ypx(k)  * eep(k);%Equation 5.31
es(k)=fs(k)*eex(k)+(1-fs(k)/Yxs(k)-dt*F(k)/v(k))*ees(k)+Ypx(k)*
fs(k)*eep(k);%Equation 5.32
ep(k)=-fp(k)*eex(k)+(fp(k)/Yxs(k))*ees(k)+(1-fp(k)-
dt*F(k)/v(k))*eep(k);%Equation 5.33
ev(k)=fv(k)*x(k)*eex(k)-(fv(k)*(si-s(k))*ees(k))+(fv(k)*p(k)*eep(k))+
eev(k);
%================================================================

%===========Calculation of the gradients for control variables=========
eF(k)=-dt*x(k)/v(k)*eex(k)+(dt*(si-s(k))/v(k))*ees(k)-
(dt*p(k)/v(k))*eep(k)+dt*eev(k);%Equation 5.34
eT(k) = (f1(k)*dmmaxdT(k)+f2(k)*dKsdT(k))*eex(k)  + ((-
f1(k)/(Yxs(k))^2)*(dmmaxdT(k)*Yxs(k)-mmax(k)*dYxsdT(k))+
(f2(k)/Yxs(k))*dKsdT(k))  *ees(k)  +
(f1(k)*(dYpxdT(k)*mmax(k)+Ypx(k)*dmmaxdT(k))-
Ypx(k)*f2(k)*dKsdT(k))*eep(k);
%Equation 5.35
n1=[ex(k) es(k) ep(k) ev(k) eF(k) eT(k)]';
%================================================================

nn = norm(n1); %Calculating norm of gradients
if nn <=eps1; %Check the condition for end of the iterations on the
first level
break
```

```
%CALCULATION ON THE SECOND LEVEL
%Calculation of the final value of x(k)=x(K+1)=xK1;sK1,pK1,vK1

xK1=x(K)+f1(K)-dt*kd*x(K)-dt*F(K)*x(K)/v(K);
sK1=s(K)-f1(K)/Yxs(K)-dt*m*x(K)+dt*F(K)*(si-s(K))/v(K);
pK1=p(K)+f1(K)*Ypx(K)-dt*F(K)*p(K)/v(K);
vK1=v(K)+dt*F(K);
if xK1 <= xmin
    xK1 = xmin;
elseif xK1 >=xmax
    xK1 = xmax;
end
if sK1 <= smin
    sK1 = smin;
elseif sK1 >= smax
    sK1 = smax;
end
if pK1 <= pmin
    pK1 = pmin;
elseif pK1 >= pmax
pK1=pmax;
end
if vK1 <= vmin
    vK1 = vmin;
elseif vK1 >= vmax
    vK1 = vmax;
end
x(K+1)=xK1;
p(K+1)=pK1;
s(K+1)=sK1;
v(K+1)=vK1;
%=======================================================================
%=======Calculation of the gradients of the conjugate variables elv
where v = x,s,p,v====
for k=1:K
elx(k) = dx(k) - x(k+1);
els(k) = ds(k) - s(k+1);
elp(k) = dp(k) - p(k+1);
elv(k) = dv(k) - v(k+1);
end
%=======================================================================
%Calculation of the norm of the gradients of the conjugate variables
nn1=norm(elx)+ norm(els)+ norm(elp)+ norm(elv); % adding the norm of
errors

if nn1<=eps2% check condition for termination of the second level
iteration
break

else
%=Calculation of the improved values of the coordinating variables and
the interconnection in time domain lv and dv, where v = x,s,p,v
for k =1:K
lx(k) = lx(k) - alx*elx(k);
ls(k) = ls(k) - als*els(k);
lp(k) = lp(k) - alp*elp(k);
```

```
%CALCULATION ON THE SECOND LEVEL
%Calculation of the final value of x(k)=x(K+1)=xK1;sK1,pK1,vK1

xK1=x(K)+f1(K)-dt*kd*x(K)-dt*F(K)*x(K)/v(K);
sK1=s(K)-f1(K)/Yxs(K)-dt*m*x(K)+dt*F(K)*(si-s(K))/v(K);
pK1=p(K)+f1(K)*Ypx(K)-dt*F(K)*p(K)/v(K);
vK1=v(K)+dt*F(K);
if xK1 <= xmin
    xK1 = xmin;
elseif xK1 >=xmax
    xK1 = xmax;
end
if sK1 <= smin
    sK1 = smin;
elseif sK1 >= smax
    sK1 = smax;
end
if pK1 <= pmin
    pK1 = pmin;
elseif pK1 >= pmax
pK1=pmax;
end
if vK1 <= vmin
    vK1 = vmin;
elseif vK1 >= vmax
    vK1 = vmax;
end
x(K+1)=xK1;
p(K+1)=pK1;
s(K+1)=sK1;
v(K+1)=vK1;
%==========================================================================
%=======Calculation of the gradients of the conjugate variables elv
where v = x,s,p,v====
for k=1:K
elx(k) = dx(k) - x(k+1);
els(k) = ds(k) - s(k+1);
elp(k) = dp(k) - p(k+1);
elv(k) = dv(k) - v(k+1);
end
%==========================================================================
%Calculation of the norm of the gradients of the conjugate variables
nn1=norm(elx)+ norm(els)+ norm(elp)+ norm(elv); % adding the norm of
errors

if nn1<=eps2% check condition for termination of the second level
iteration
break

else
%=Calculation of the improved values of the coordinating variables and
the interconnection in time domain lv and dv, where v = x,s,p,v
for k =1:K
lx(k) = lx(k) - alx*elx(k);
ls(k) = ls(k) - als*els(k);
lp(k) = lp(k) - alp*elp(k);
```

```
lv(k) = lv(k) - alv*elv(k);

dx(k) = dx(k) + adx*(-lx-mx*elx(k));
ds(k) = ds(k) + ads*(-ls-ms*els(k));
dp(k) = dp(k) + adp*(-lp-mp*elp(k));
dv(k) = dv(k) + adv*(-lp-mp*elp(k));
end
%=============================================================================
end% End for the norm
j=j+1;
end% End for second level iteration
toc; %End the clock
=========================================END=========================
%=================================Plot Results==========================
figure(2)
t=1:K+1;
plot(t,x,'-k')
title('Optimised Results for the biomass state')
ylabel('g/l');
xlabel('time in samples');
legend('biomass','Location','NorthEastOutside')
figure(3)
plot(t,p,':k')
title('Optimised Results for the product state')
ylabel('g/l');
xlabel('time in samples');
legend('product','Location','NorthEastOutside')
figure(4)
title('Optimised Results for the substrate')
ylabel('g/l');
xlabel('time in samples');
plot(s,'-k')
legend('substrate','Location','NorthEastOutside')
figure(5)
plot(t,v,'-.k')
title('Optimised Results for the volume state')
ylabel('g/l');
xlabel('time in samples');
legend('volume','Location','NorthEastOutside')
t1=1:K;
figure(6)
plot(t1,T,'-k')
title('Optimised Results for the Temperature T')
ylabel('g/l');
xlabel('time in samples');
legend('Temperature T','Location','NorthEastOutside')
figure(7)
plot(t1,F,'--k')
title('Optimised Results for the input flow rate F')
ylabel('g/l');
xlabel('time in samples');
legend('Input flow rate F','Location','NorthEastOutside')
figure(8)
plot(t1,lx,'-k',t1,ls,':k',t1,lp,'-.k',t1,lv,'--k')
title('Optimised Results for the conjugate variables lambda')
ylabel('g/l');
xlabel('time in samples');
```

```
legend('Lambda x','Lambda s','Lambda p','Lambda
v','Location','NorthEastOutside')
%=====================================================================
```

B.2: MATLAB script file – modelfedbatch_func2.m

```
M-File: modelfedbatch_func2.m
%=======================M-FILE DESCRIPTION =======================
%The function is used to calculate the trajectories of the derivatives
of the state space vector

function ydot=modelfedbatch_func2(t,y,par, T, F, kd, m, pl, n, si)

%========================Model_variables==========================
ydot(1)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))-(kd*y(1))-
F*y(1)/y(4)); %Equation 2.84
ydot(2)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))/-
(par(7)+par(8)*T+par(9)*(T*T))-m*y(1)-F*(si-y(2))/y(4));%Equation 2.85
ydot(3)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T))+y(2))*(par(10)+p
ar(11)*T+par(12)*(T*T))-F*y(3)/y(4);%Equation 2.86
ydot(4)=F; %Equation 2.87
ydot=[ydot(1) ydot(2) ydot(3) ydot(4)]';
%=====================================================================
```

APPENDIX C: MATLAB SEQUENTIAL PROGRAM FOR CALCULATION OF CONTROL AND STATE FOR CONTINUOS FERMENTATION PROCESS

C.1: MATLAB script file – continuous_seque.m

```
% M-File: continuous_seque.m
%========================M-FILE DESCRIPTION ==========================
%Calculation of state and control trajectories for continuous
fermentation process

clear all% clear workspace
tic; % start the clock
%======================Set Initial values============================
M1 = 100; M2 = 50; %Max number of iterations of the first and second
levels
K = 256; %Number of steps in the optimization horizon
ax = 0.001; as = 0.001; ap = 0.001; %Steps of the gradient procedures
for calculation of the State variables
aT = 0.05; aD = 0.05; %Steps of the gradient procedures for calculation
of the Control variables
alx = 0.001; als = 0.001; alp = 0.001; %Step of the gradient procedures
for calculation of the Conjugate variables
adx = 0.001; ads = 0.001; adp = 0.001; %Steps of the gradient procedures
for calculation of the Interconnections in time
eps1=0.001; %tolerances for stop of the calculation of the first levels
eps2=0.001; %tolerances for stop of the calculation of the second levels
%====================================================================

%=============Initial values of the penalty coefficients=============
mx=0.0010;
ms=0.0010;
mp=0.0010;
%====================================================================
dt = 0.01967; %Initial value of the sampling period

%==========Initial trajectories for the conjugate variables==========
for k = 1: K+1
lx (k) =0.001;
ls (k) =0.005;
lp (k) =0.002;
end
%====================================================================

%========Initial trajectories of the control variables===============
T = 16;
D = 0;
%====================================================================

%======================Model co-efficients===========================
a1 = 0.0004; a2 = 0.0001; a3 = 0.0002;
b1 = 0.2; b2 = 0.03; b3 = 0.01;
c1 = 2; c2 = 0.3; c3 = 0.021;
d1 = 0.0040; d2 = 0.006; d3 = 0.001;
par = [a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3];
kd = 0.00008; % death constant
p1 = 50; % inhibition factor
```

```
m = 0.11; % Maintenance co-efficient
n = 0.52; % Power co-efficient
si= 266.6; % Input substrate
Ks = 0.5; %Initial value of Monod constant for calculation of dtmin and
dtmax
mmax = 0.5; %Initial value of maximum growth rate for calculation of
dtmin and dtmax
Yxs = 0.172; Initial value of stochiometry coefficient for biomass
growth for calculation of dtmin and dtmax
Ypx = 0.421; Initial value of stochiometry coefficient for biomass
growth for calculation of dtmin and dtmax
%=========================================================================

%=================Initial values of state variables===================
x0=1.83;
s0=150.0;
p0= 0;
y0=[x0 s0 p0]';
%=========================================================================

%================steady state values of state variables================
xbar = 100*ones (1, K+1);
sbar = 30*ones (1, K+1);
pbar = 100*ones (1, K+1);
%=========================================================================

%===================Constraints of control variables==================
Tmin = 10;
Dmin = 0;
Tmax = 25;
Dmax = 5;
%=========================================================================

%====================Constraints of State variables===================
xmin = 0.83;
xmax = 100;
smin = 5;
smax = 300;
pmin = 0;
pmax = 100;
%=========================================================================
%=================Calculation of delta min and max====================
dtmax = 0.01*(Ks + smax)/ ((mmax-kd-Dmax)*smax - Ks*(kd + Dmax));
dtmin = 0.01*(Ks + smin)/ ((mmax-kd-Dmin)*smin - Ks*(kd + Dmin));
%=========================================================================

%==Calculation of the initial trajectories of state variables==========
t=1: K+1;
[t, y] =ode15s (@ (t, y) modelcontinuos_func2 (t, y, par, T, D, kd, m,
pl, n, si), t, y0);
x=(y (:, 1))';
s=(y(:,2))';
p=(y(:,3))';
t = 1:K+1;
figure(1)
t = 1:K+1;
plot(t,x,'-k',t,s,':k',t,p,'-.k')
```

```
title('Initial trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','Location','NorthEastOutside')
%=======================================================================

% Calculation of the initial trajectories of the interconnections in
time
domain=================================================================
===
for k = 1:K
dx(k) = 10.0;
ds(k) = 138;
dp(k) = 25.0;
end
%=======================================================================
%=========Initial trajectories of the control variables=================
T = 16*ones (1, K);
D = 1.25*ones (1, K);
%=======================================================================

%====Calculation of the initial trajectories of el_v_(k,v=s,x,p=========
 for k=1:K
 elx(k) = dx(k)-x(k+1);
 els(k) = ds(k)-s(k+1);
 elp(k) = dp(k)-p(k+1);
 end
%=======================================================================

 %==========Calculation of elx(K) at the last point===================
 elx(K) = x(K+1)- xbar(K+1);
 els(k) = s(K+1)- sbar(K+1);
 elp(k) = p(K+1)- pbar(K+1);
%=======================================================================

%=============Start of the coordinating procedure=====================
j=1
while j <= M2;%Second level iterations

for k=1:K %Decomposition in time domain
i=1;
while i <= M1 %First level iterations

%========PreAllocation of variables for faster execution==============
dmmaxdT = zeros(1,K);
dKsdT =  zeros(1,K);
dYxsdT = zeros(1,K);
dYpxdT = zeros(1,K);
 mmax  = zeros(1,K);
 Ks    = zeros(1,K);
 Yxs   = zeros(1,K);
 Ypx   = zeros(1,K);
 ff    = zeros(1,K);
 fx    = zeros(1,K);
 fs    = zeros(1,K);
 fp    = zeros(1,K);
 f1    = zeros(1,K);
 f2     = zeros(1,K);
 eex    = zeros(1,K);
 ees    = zeros(1,K);
 eep    = zeros(1,K);
 ex     = zeros(1,K);
 es     = zeros(1,K);
 ep     = zeros(1,K);
 eT     = zeros(1,K);
%=====================================================================

%CALCULATION OF THE DERIVATIVES OF THE KINETIC COEFFICIENTS TOWARDS THE
CONTROL VARIABLES

%=======================For the temperature==========================
dmmaxdT(k) = par(2)+2*par(3)*T(k); %Equations 5.10
dKsdT(k)=par(5)+2*par(6)*T(k);
dYxsdT(k)=par(8)+2*par(9)*T(k);
dYpxdT(k)=par(11)+2*par(12)*T(k);
%=====================================================================

%===========Calculation of the kinetic coefficients===================
mmax(k)=par(1)+par(2)*T(k)+par(3)*(T(k)^2); % Equation 2.72
Ks(k)=par(4)+par(5)*T(k)+par(6)*(T(k)^2);    % Equation 2.73
Yxs(k)=par(7)+par(8)*T(k)+par(9)*(T(k)^2);   % Equation 2.74
Ypx(k)=par(10)+par(11)*T(k)+par(12)*(T(k)^2); % Equation 2.75
%=====================================================================

%=======Calculation of the expressions from equation (6.38)-(6.42=======
pk=p(k);
ff(k)=dt*mmax(k) *((1-pk/pl)^n);
ff2(k)= dt*((1-pk/pl)^n);
fx(k)=ff(k)*s(k)/(Ks(k) + s(k));  % Equation 6.38
fs(k)=ff(k)*Ks(k) * x(k)/((Ks(k) + s(k))^2); %Equation 6.40
fp(k)=-dt*mmax(k) * Ypx(k) *((1-p(k)/pl)^(n-1))*n*x(k)*s(k)/(pl*(Ks(k) +
s(k))); % Equation 6.39
f1(k)=ff2(k)*x(k)*s(k)/(Ks(k) + s(k)); %Equation 6.41
f2(k)=-f1(k)*mmax(k)/(Ks(k) + s(k)); % Equation 6.42
%=====================================================================

%===Calculation of the expressions ee_v_(k)=lambda(k)-m_v*E_lambda_v_(k)
eex(k)=lx(k)-mx*elx(k);
ees(k)=ls(k)-ms*els(k);
eep(k)=lp(k)-mp*elp(k);
%=====================================================================

%==========Calculation of the gradients for state variables============
ex(k)=(1+fx(k)-dt*kd+dt*D(k))*eex(k)+((fx(k)/Yxs(k))-dt*m)*ees(k)+
fx(k)*Ypx(k) * eep(k);%Equation 6.43
es(k)=fs(k)*eex(k)+(1-fs(k)/Yxs(k)-dt*D(k))*ees(k)+Ypx(k)*
fs(k)*eep(k);%Equation 6.44
ep(k)=fp(k)*eex(k)-(fp(k)/Yxs(k))*ees(k)+(1+fp(k)-
dt*D(k))*eep(k);%Equation 6.45
%=====================================================================
%============Calculation of the gradients for control variables========
eD(k) =-dt*x(k)*eex(k)+(dt*(si-s(k))*ees(k))-(dt*p(k)*eep(k)); %Equation
6.47
```

```
eT(k) = (f1(k)*dmmaxdT(k)+f2(k)*dKsdT(k))*eex(k) + ((-
f1(k)/(Yxs(k))^2)*(dmmaxdT(k)*Yxs(k)-mmax(k)*dYxsdT(k))+
(f2(k)/Yxs(k))*dKsdT(k)) *ees(k) +
(f1(k)*(dYpxdT(k)*mmax(k)+Ypx(k)*dmmaxdT(k))-
Ypx(k)*f2(k)*dKsdT(k))*eep(k);%Equation 6.46
n1=[ex(k) es(k) ep(k) eD(k) eT(k)]';
%==============================================================
nn = norm(n1); %Calculation norm of gradients
if nn <=eps1%Check the condition for end of the iterations on the first
level
break
%==========Calculation of the new values of the state and control
variables
elseif k == 1
x(k)=x0;
s(k)=s0;
p(k)=p0;
T(k) = T(k) - aT*eT(k);
D(k) = D(k) - aD*eD(k);
else
x(k) = x(k) - ax*ex(k);
s(k) = s(k) - as*es(k);
p(k) = p(k) - ap*ep(k);
T(k) = T(k) - aT*eT(k);
D(k) = D(k) - aD*eD(k);
end
%==============================================================

%========================Check against constraints===================
if x(k) < xmin
    x(k) = xmin;
elseif x(k) > xmax
    x(k) = xmax;
end
if s(k) < smin
    s(k) = smin;
elseif s(k) > smax
    s(k) = smax;
end
if p(k) < pmin
    p(k) = pmin;
elseif p(k) > pmax
    p(k) = pmax;
end
if D(k) < Dmin
    D(k) = Dmin;
elseif D(k) > Dmax
    D(k) = Dmax;
end
if T(k) < Tmin
    T(k) = Tmin;
elseif T(k) > Tmax
   T(k) = Tmax;
End
%==============================================================

i=i+1;
```

```
end %End of the iterations on the first level
end %end of the iteration for k=1,K

%CALCULATION ON THE SECOND LEVEL
%Calculation of the final value of x(k)=x(K+1)=xK1;sK1,pK1

%==========calculation of the gradients at the final point=============
K1 = K+1;
ex(K1)= lx(K1)+ mx*(x(K1)-xbar(K1));
es(K1)= ls(K1)+ ms*(s(K1)-sbar(K1));
ep(K1)= lp(K1)+ mp*(p(K1)-pbar(K1));
%=====================================================================

%===============Calculation of the final value of state===============
xK1 = x(K1) - alx*ex(K1);
sK1 = s(K1) - als*es(K1);
pK1 = p(K1) - alp*ep(K1);
%=====================================================================

%===============Check the constraints at the final point===============
if xK1 <= xmin
    xK1 = xmin;
elseif xK1 >=xmax
    xK1 = xmax;
end
if sK1 <= smin
    sK1 = smin;
elseif sK1 >= smax
    sK1 = smax;
end
if pK1 <= pmin
    pK1 = pmin;
elseif pK1 >= pmax
    pK1=pmax;
end
%=====================================================================
x(K+1)=xK1;
p(K+1)=pK1;
s(K+1)=sK1;

%Calculation on the coordinating level
%Calculation of the new values of the gradients of the coordinating
%variables  using  the  values  of  the  calculated  state  and  control
variables
for k=1:K

elx(K+1)  = x(K+1)-xbar(K+1);
els(K+1)  = s(K+1)-sbar(K+1);
elp(K+1)  = p(K+1)-pbar(K+1);
end
nn1=norm(elx)+ norm(els)+ norm(elp);

if nn1<=eps2
    break
else
% ==========Calculation of the improved values of lv, v = x,s,p=========
for k =1:K+1
```

```
lx(k) = lx(k) + alx*elx(k);
ls(k) = ls(k) + als*els(k);
lp(k) = lp(k) + alp*elp(k);
end
%===========================================================================

%=====Calculation of the improved values of the sampling period========
mmax(K)=par(1)+par(2)*T(K)+par(3)*(T(K)^2); % Equation 2.72
Ks(K)=par(4)+par(5)*T(K)+par(6)*(T(K)^2);   % Equation 2.73
Yxs(K)=par(7)+par(8)*T(K)+par(9)*(T(K)^2);  % Equation 2.74
Ypx(K)=par(10)+par(11)*T(K)+par(12)*(T(K)^2); % Equation 2.75
eex(K)=lx(K)+ mx*(x(K)-xbar(K));
ees(K)=ls(K)+ ms*(s(K)-sbar(K));
eep(K)=lp(K)+ mp*(p(K)-pbar(K));
ff1(K) = mmax(K) *((1-pk/pl)^n);
f3(K) = ff1(K)*x(K)*s(K)/(Ks(K) + s(K));% Equation 6.24
dt =  (-K -(f3(K)-(kd*x(K)-D(K)*x(K))*eex(K)+(f3(K)/Yxs(K)+ m*x(K)-
D(K)*(si-s(K))*ees(K)-(Ypx(K)*f3(K)-D(K)*p(K))*eep(K))/(mx*(f3(K)-
kd*x(K)-D(K)*x(K))^2 + ms*(f3(K)/Yxs(K)+ m*x(K)-D(K)*(si-s(K))^2+
mp*(Ypx(K)*f3(K))-D(K)*p(K))^2);
% Equation 6.63
%===========================================================================
%=====Check the constraints for the improved value of dt===============
if dt <= dtmin
   dt = dtmin;
elseif dt >=dtmax
    dt = dtmax;
end
for k = 1:K
    dx(k) = dx(k) - adx*(-mx*elx(k)-lx(k));
    ds(k) = ds(k) - ads*(-ms*els(k)-ls(k));
    dp(k) = dp(k) - adp*(-mp*elp(k)-lp(k));
end
end% End for the norm
j = j+1;
end % End of the second level iteration
toc; %end of the clock
%======================================END=============================
%=============================Plot Results============================
figure(2)
t=1:K+1;
plot(t,x,'-k')
title('Optimised Results for the biomass state')
ylabel('g/l');
xlabel('time in samples');
legend('biomass','Location','NorthEastOutside')
figure(3)
plot(t,p,':k')
title('Optimised Results for the product state')
ylabel('g/l');
xlabel('time in samples');
legend('product','Location','NorthEastOutside')
figure(4)
plot(t,s,'-k')
title('Optimised Results for the substrate')
ylabel('g/l');
xlabel('time in samples');
```

```
legend('substrate','Location','NorthEastOutside')
t1=1:K;
figure(5)
plot(t1,T,'-k')
title('Optimised Results for the Temperature T')
ylabel('g/l');
xlabel('time in samples');
legend('Temperature T','Location','NorthEastOutside')
t1=1:K;
figure(6)
plot(t1,D,'--k')
title('Optimised Results for the Dilution rate D')
ylabel('g/l');
xlabel('time in samples');
legend('Dilution rate D','Location','NorthEastOutside')
figure(7)
t1=1:K+1;
plot(t1,lx,'-k',t1,ls,':k',t1,lp,'-.k')
title('Optimised Results for the conjugate variables lambda')
ylabel('g/l');
xlabel('time in samples');
legend('Lambda x','Lambda s','Lambda p','Location','NorthEastOutside')
figure(8)
t = 1:K+1;
plot(t,x,'-k',t,s,':k',t,p,'-.k')
title('Optimal trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','Location','NorthEastOutside')

%===========================================================================
```

C.2: MATLAB script file – modelcontinuous_func2.m

```
%M-File: modelcontinuous_func2.m
%========================M-FILE DESCRIPTION ==========================
%The function is used to calculate the trajectories of the derivatives
of the state space vector
function ydot= modelcontinuos_func2(t,y, par, T, D, kd, m, pl, n, si)
%==============================Model_variables=========================
ydot(1)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))-(kd*y(1))-
D*y(1)); %Equation 2.93
ydot(2)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))/-
(par(7)+par(8)*T+par(9)*(T*T))-m*y(1)+ D*(si-y(2)));%Equation 2.94
ydot(3)=(par(1)+par(2)*T+par(3)*(T*T))*(par(10)+par(11)*T+par(12)*(T*T))
*((1-(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T))+y(2))-
(D*y(3));%Equation 2.95
ydot=[ydot(1) ydot(2) ydot(3)]';
%===========================================================================
```

APPENDIX D: MATLAB PARALLEL PROGRAM FOR CALCULATION OF CONTROL AND STATE FOR BATCH FERMENTATION PROCESS

D.1: MATLAB script file – parallel_batch.m

```
% M-File: batch_parallel.m
%=========================M-FILE DESCRIPTION ==========================
% Calculation of state and control trajectories for batch fermentation
% process

clear all% clear workspace
%=========================Set Initial values===========================

global par T kd m pl n si K M1 M2 eps1 mu l x s p el conmin conmax
statemin statemax astate acon dt y0 x1 s1 p1 T1

M1 = 100; M2 = 50; %Max number of iterations of the first and second
levels
K = 256; %Number of steps in optimization horizon
ax = 0.001; as = 0.001; ap = 0.001; %Steps of the gradient procedures
for calculation of the State variables
aT = 0.05; %Steps of the gradient procedures for calculation of the
Control variables
alx = 0.001; als = 0.001; alp = 0.001; %Step of the gradient procedures
for calculation of the Conjugate variables
adx = 0.001; ads = 0.001; adp = 0.001; %Steps of the gradient procedures
for calculation of the Interconnections in time
eps1=0.001; %tolerances for stop of the calculation of the first level
eps2=0.001; %tolerances for stop of the calculation of the second levels
%=====================================================================

%==========Initial values of the penalty coefficients==================
mx=0.0010;
ms=0.0010;
mp=0.0010;
%=====================================================================


%===Initial trajectory values of the conjugate variables===============
for k = 1: K
lx (k) =0.001;
ls (k) =0.005;
lp (k) =0.002;
end
%=====================================================================

%===Initial trajectories of the control variables for calculation of===
T=16.0;
%=====================================================================

%=======================Model co-efficients===========================
a1 = 0.0004; a2 = 0.0001; a3 = 0.0002;
b1 = 0.2; b2 = 0.03; b3 = 0.01;
c1 = 2; c2 = 0.3; c3 = 0.021;
d1 = 0.0040; d2 = 0.006; d3 = 0.001;
```

```
par = [a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3];
kd = 0.00008; % death constant
pl = 50; % inhibition factor
m = 0.11; % Maintenance co-efficient
n = 0.52; % Power co-efficient
si= 266.6; % Input substrate
dt = 0.25; % delta t
%============================================================================

%==============Initial values of state variables=====================
x0=1.83;
s0=150.0;
p0= 0;
y0=[x0 s0 p0]';
%============================================================================

%============Constraints of control variable=========================
Tmin = 10;
Tmax = 20;
%============================================================================

%==============Constraints of state variables========================
xmin = 0.83;
xmax =100;
smin = 5;
smax = 300;
pmin = 0;
pmax = 100;
%============================================================================

%===Calculation of the initial trajectories of state variables=========
t=1: K+1;
[t, y] =ode15s (@ (t, y) modelbatch_func2(t, y, par T, kd, m, pl, n,
si), t, y0);
x=(y (:, 1))';
s=(y(:,2))';
p=(y(:,3))';
figure(1)
plot(t,x,'-k',t,s,':k',t,p,'-.k'),
title('Initial trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','Location','NorthEastOutside')
%============================================================================

%Calculation of the initial trajectories of the interconnections in time
domain=============================================================
for k = 1:K
    dx(k) = 10.0;
    ds(k) = 138;
    dp(k) = 25.0;
end
%============================================================================
 %===Initial trajectories of the control variables for calculation
of====
T=16.0*ones(1,K);
%============================================================================
%====Calculation of the initial trajectories of Error_Lambda_v_(k)======
 for k=1:K
    elx(k) = dx(k)-x(k+1);
    els(k) = ds(k)-s(k+1);
    elp(k) = dp(k)-p(k+1);
 end
%======================================================================
%================START OF THE COORDINATING PROCEDURE===================
j=1;
while j <= M2;%Second level iterations
if matlabpool('size')> 0 %Checking if there is a pool of workers open
   matlabpool('close')% Sending a close signal to workers if there is a
pool open
end
matlabpool open jobmanagerconfig5%Opening 32 matlab workers from the
cluster
t0 = clock; % start of the clock on second level calculation
j=1
while j <= M2;%Second level iterations
diary parallelfedbatch %opening blank page for printing results

jm = findResource('scheduler','configuration','jobmanagerconfig5')
disp(get(jm))%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% check the status of
the jobmanager
timingStart = tic;
start = tic;
pjob1=createParallelJob(jm,'Configuration','jobmanagerconfig5'); %create
parallel job from the jobmanager
get(pjob1) % Monitor the status of pjob1

%==========How long it takes to create a job===========================
times.jobCreateTime = toc(start)
description.jobCreateTime = 'Job creation time'
%======================================================================

Tasks = findTask(pjob1)
[pending running finished] = findTask(pjob1)
set(pjob1,'Configuration','jobmanagerconfig5')
set(pjob1,'MinimumNumberOfWorkers',32);
set(pjob1,'MaximumNumberOfWorkers',32);
set(pjob1,'FileDependencies',{'subprob2_batch.m'})
get(jm)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Notice how jobmanager
is changing
disp(get(pjob1))
disp('busy workers')
disp(get(jm, 'NumberOfbusyWorkers'))

%====================STARTING PARALLEL COMPUTING=======================
%Create task- only one with 4 output arguments  and not input arguments,
%measure how long that takes and measure how long it takes to submit the
%job to the cluster
task=createTask(pjob1, @subprob2_batch,4,{})
get(pjob1,'task')

%==========How long it takes to create a task==========================
times.taskCreateTime = toc(start)
```

```
description.taskCreateTime = 'Task creation time'
%=====================================================================
submit(pjob1)
%===========How long it takes to submit a job=========================
times.submitTime = toc(start)
description.submitTime = 'Job submission time'
%=====================================================================

%Once the job has been submitted, we hope all its tasks execute in
%parallel. Measure how long it takes for all the tasks to start and to
run
%to completion.
waitForState(pjob1,'finished')
times.jobWaitTime = toc(start)
description.pjobWaitTime = 'Job wait time'
get(jm)
%Tasks have now completed, so we are again executing a code in the
matlab
%client. Measure how long it takes to retrieve all the job results
results=getAllOutputArguments(pjob1)
times.resultsTime = toc(start)
description.resultsTime = 'Results retrieval time'
%=====================================================================
%============Verify that the job ran without errors===================
errmsgs = get(pjob1.task, {'ErrorMessage'})
nonempty = ~cellfun(@isempty, errmsgs)
celldisp(errmsgs(nonempty))
%=====================================================================

%Measure the total time elapsed from creating the job up to the results
times.totalTime = toc(start)
operationtime = etime(clock,t0) %operation time for completion of
parallel computing
destroy(pjob1)
matlapool close% close matlapool of workers

%Calculation of the improved values of the coordinating variables
%Calculation of the errors e_lambda_V (k)

for (k=1:K)
elx(k) = dx(k) - x(k+1);
els(k) = ds(k) - s(k+1);
elp(k) = dp(k) - p(k+1);

end
el=[elx;els;elp ];

%==============Calculation of the norm of the errors=================
nn1=norm(elx)+ norm(els)+ norm(elp);
%=====================================================================
%Check of the convergence of the coordinating procedure for the end of
the
%calculations
if nn1<=eps2
    break
else
for (k =1:K)
```

```
lx(k) = lx(k) - alx*elx(k);
ls(k) = ls(k) - als*els(k);
lp(k) = lp(k) - alp*elp(k);

edx(k) = -lx(k)-mx*elx(k);
eds(k) = -ls(k)-ms*els(k);
edp(k) = -lp(k)-mp*elp(k);

dx(k) = dx(k) + adx*edx(k);
ds(k) = ds(k) + ads*eds(k);
dp(k) = dp(k) + adp*edp(k);
end
end
l=[lx;ls;lp];
j=j+1
end
%================================Plot Results========================
figure(2)
t=1:K+1;
plot(t,x,'-k')
title('Optimised Results for the biomass state')
ylabel('g/l')
xlabel('time in samples')
legend('biomass','Location','NorthEastOutside')
figure(3)
plot(t,p,':k')
title('Optimised Results for the product state')
ylabel('g/l')
xlabel('time in samples')
legend('product','Location','NorthEastOutside')
figure(4)
title('Optimised Results for the substrate')
ylabel('g/l')
xlabel('time in samples')
plot(s,'-k')
legend('substrate','Location','NorthEastOutside')
t1=1:K;
figure(5)
plot(t1,T,'-k')
title('Optimised Results for the Temperature T')
ylabel('g/l');
xlabel('time in samples');
legend('Temperature T','Location','NorthEastOutside')
figure(6)
plot(t1,lx,'-k',t1,ls,':k',t1,lp,'-.k')
title('Optimised Results for the conjugate variables lambda')
ylabel('g/l');
xlabel('time in samples');
legend('Lambda x','Lambda s','Lambda p','Location','NorthEastOutside')
%====================================================================
```

D.2: MATLAB script file – subprob2_batch.m

```
M-File: subprob2_batch.m
%========================M-FILE DESCRIPTION ========================
%The function is used to calculate the optimal trajectories of the state
and control by different number of workers in parallel

function [x,s,p,T ]=subprob2_batch

global par T kd m pl n si  K M1 M2 eps1 mu l x s p el conmin conmax
statemin statemax astate acon dt y0 x1 s1 p1 T1

%======Calculation of the initial and final point of the subintervals===
dd=K/numlabs
a=(labindex-1)*dd+1
b=labindex*dd
%=====================================================================
for (k=a:b) %Decomposition in time domain
    i=1
    while i <= M1 %First level iterations

%=========PreAllocation of variables for faster execution=============
dmmaxdT = zeros(1,K);
dKsdT =  zeros(1,K);
dYxsdT = zeros(1,K);
dYpxdT = zeros(1,K);
 mmax  = zeros(1,K);
 Ks    = zeros(1,K);
 Yxs   = zeros(1,K);
 Ypx   = zeros(1,K);
 ff    = zeros(1,K);
 fx    = zeros(1,K);
 fs    = zeros(1,K);
 fp    = zeros(1,K);
 f1    = zeros(1,K);
 f2    = zeros(1,K);
 eex   = zeros(1,K);
 ees   = zeros(1,K);
 eep   = zeros(1,K);
 ex    = zeros(1,K);
 es    = zeros(1,K);
 ep    = zeros(1,K);
 eT    = zeros(1,K);
%=====================================================================

%=Calculation of the parameters derivatives torwards the temperature====
dmmaxdT(k) = par(2)+2*par(3)*T(k); %Equations 5.10
dKsdT(k)=par(5)+2*par(6)*T(k);
dYxsdT(k)=par(8)+2*par(9)*T(k);
dYpxdT(k)=par(11)+2*par(12)*T(k);
%=====================================================================
%============Calculation of the kinetic coefficients==================
mmax(k)=par(1)+par(2)*T(k)+par(3)*(T(k)^2);   % Equation 2.72
Ks(k)=par(4)+par(5)*T(k)+par(6)*(T(k)^2);     % Equation 2.73
Yxs(k)=par(7)+par(8)*T(k)+par(9)*(T(k)^2);    % Equation 2.74
Ypx(k)=par(10)+par(11)*T(k)+par(12)*(T(k)^2);%Equation2.75
%=====================================================================

%Calculation of the expressions lambda_v_(k), k=0,K-1, v = x,s,p and
lamda1(k), lamda2(k)
ff(k)=dt*mmax(k) *((1-p(k)/pl)^n);
```

```
fx(k)=ff(k)*s(k)/(Ks(k) + s(k));     % Equation 4.20
fs(k)=ff(k)*Ks(k)*x(k)/((Ks(k) + s(k))^2); %Equation 4.22
fp(k)=dt*mmax(k)*((1-p(k)/pl)^(n-1))*x(k)*s(k)/(pl*(Ks(k)+s(k)));%
Equation 4.21
ff1(k)=ff(k)*x(k)*s(k)/(Ks(k) + s(k)); %Equation 4.23
f2(k)=ff1(k)/((Ks(k) + s(k)); % Equation 4.24
f1(k) = ff1(k)/mmax(k)
%=========================================================================

%Calculation of the expressions ee_v_(k)=lambda(k)-m_v*E_lambda_v_(k)
eex(k)=l(1,k)-mu(1)*el(1,k);
ees(k)=l(2,k)-mu(2)*el(2,k);
eep(k)=l(3,k)-mu(3)*el(3,k);
%=========================================================================

%=======Calculation of the gradients for state variables===============
ex(k)=(1+fx(k)-dt*kd)*eex(k)+((fx(k)/Yxs(k))-
dt*m)*ees(k)+dt*fx(k)*Ypx(k) * eep(k);%Equation 4.25
es(k)=fs(k)*eex(k)+(1-fs(k)/Yxs(k))*ees(k)+Ypx(k)*
f1(k)*eep(k);%Equation 4.26
ep(k)=-fp(k)*eex(k)+(fp(k)/Yxs(k))*ees(k)+(1-fp(k))*eep(k);%Equation
4.27
%=========================================================================

%==========Calculation of the gradients for control variables==========
eT(k) = (f1(k)*dmmaxdT(k)+f2(k)*dKsdT(k))*eex(k) + ((-
f1(k)/(Yxs(k))^2)*(dmmaxdT(k)*Yxs(k)-mmax(k)*dYxsdT(k))+
(f2(k)/Yxs(k))*dKsdT(k)) *ees(k) +
(f1(k)*(dYpxdT(k)*mmax(k)+Ypx(k)*dmmaxdT(k))-
Ypx(k)*f2(k)*dKsdT(k))*eep(k);
%Equation 4.28
n1=[ex(k) es(k) ep(k) eT(k)]';
%=========================================================================

nn = norm(n1) %Calculation norm of gradients
if nn <=eps1%Check the condition for end of the iterations on the first
level
break
%Calculation of the new values of the state and control variables
elseif k == 1
x1(1,k,labindex)=y0(1);
s1(1,k,labindex)=y0(2);
p1(1,k,labindex)=y0(3);
T1(1,k,labindex) = T(k) + acon(1)*eT(k);
else
x1(1,k,labindex) = x(k) + astate(1)*ex(k);
s1(1,k,labindex) = s(k) + astate(2)*es(k);
p1(1,k,labindex) = p(k) + astate(3)*ep(k);
T1(1,k,labindex) = T(k) + acon(1)*eT(k)
end
i=i+1
end

%======================Check against constraints=====================
if x1(1,k,labindex) < statemin(1)
x1(1,k,labindex) = statemin(1);
elseif x1(1,k,labindex) > statemax(1)
```

```
x1(1,k,labindex) = statemax(1);
end
if s1(1,k,labindex) <statemin(2)
s1(1,k,labindex) = statemin(2);
elseif s1(1,k,labindex) > statemax(2)
s1(1,k,labindex) = statemax(2);
end
if p1(1,k,labindex) < statemin(3)
p1(1,k,labindex) = statemin(3);
elseif p1(1,k,labindex) > statemax(3)
p1(1,k,labindex) = statemax(3);
end
if T1(1,k,labindex) < conmin(1)
T1(1,k,labindex) = conmin(1);
elseif T1(1,k,labindex) > conmax(1)
T1(1,k,labindex) = conmax(1);
end
end  %end of the iterations in the subproblem time interval (a,b)
%================================================================================
labBarrier
if labindex==1
x2=reshape(x1,1,numel(size(x1)));
s2=reshape(s1,1,numel(size(s1)));
p2=reshape(p1,1,numel(size(p1)));
T2=reshape(T1,1,numel(size(T1)))
x=x2([a:b,labindex])
s=s2([a:b,labindex])
p=p2([a:b,labindex])
T=T2([a:b,labindex])

%======Calculation of the final value of x(k)=x(K+1)=xK1;sK1,pK1========
xK1=x(K)+f1(K)-dt*kd*x(K);
sK1=s(K)-f1(K)/Yxs(K)-dt*m*x(K);
pK1=p(K)+f1(K)*Ypx(K);
if xK1 <= statemin(1)
xK1 = statemin(1);
elseif xK1 >=statemax(1)
xK1 = statemax(1);
end
x(49)=xK1
if sK1 <= statemin(2)
    sK1 = statemin(2);
elseif sK1 >= statemax(2)
    sK1 = statemax(2);
end
s(49)=sK1
 if pK1 <= statemin(3)
pK1 = statemin(3);
elseif pK1 >= statemax(3)
    pK1=statemax(3)
end
 p(49)=pK1
end
%================================================================================
```

D.3: MATLAB script file – modelbatch_func2.m

```
M-File: modelbatch_func2.m
```

```
%========================M-FILE DESCRIPTION =========================
%The function is used to calculate the trajectories of the derivatives
of the state space vector

function ydot= modelbatch_func2 (t, y, par, T, kd, m, pl, n, si)

%==========================Model_variables============================

ydot(1)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))-(kd*y(1)));
%Equation 2.78
ydot(2)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))/-
(par(7)+par(8)*T+par(9)*(T*T))-m*y(1));%Equation 2.79
ydot(3)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T))+y(2))*(par(10)+p
ar(11)*T+par(12)*(T*T));%Equation 2.80
ydot= [ydot (1) ydot (2) ydot (3)]';
%=====================================================================
```

APPENDIX E: MATLAB PARALLEL PROGRAM FOR CALCULATION OF CONTROL AND STATE FOR FED-BATCH FERMENTATION PROCESS

E.1: MATLAB script file –fedbatch_ parallel.m

```
M-File: fedbatch_parallel.m
%========================M-FILE DESCRIPTION ===========================
%Calculation of state and control trajectories for fedbatch fermentation
% process

clear all% clear workspace
tic; % start the clock
global par kd m pl n si x1 s1 p1 v1 T1 F1 K M1 M2 eps1 mu l x s p v T F
el astate acon dt y0 el elx els elp conmin conmax statemin statemax
%=====================Set Initial values==============================
M1 = 5; M2 = 5; %Max number of iterations of the first and second levels
K = 48; %Number of steps in optimization horizon
ax = 0.001; as = 0.001; ap = 0.001; av = 0.001; %Steps of the gradient
procedures for calculation of the State variables
aT = 0.05; aF = 0.05; %Steps of the gradient procedures for calculation
of the Control variables
alx = 0.001; als = 0.001; alp = 0.001; alv = 0.001; %Step of the
gradient procedures for calculation of the Conjugate variables
adx = 0.001; ads = 0.001; adp = 0.001; adv = 0.001; %Steps of the
gradient procedures for calculation of the Interconnections in time
eps1=0.001; %tolerances for stop of the calculation of the first levels
eps2=0.001; %tolerances for stop of the calculation of the second levels
%====================================================================

%==========Initial values of the penalty coefficients================
mx=0.0010;
ms=0.0010;
mp=0.0010;
mv=0.0010;
%====================================================================

%========Initial trajectories for the conjugate variables=============
for k = 1: K
       lx (k) =0.001;
       ls (k) =0.005;
       lp (k) =0.002;
       lv (k) =0.003;
 end
%====================================================================

%===========Initial trajectories of the control variables===========
T=16.0;
F=0;
%====================================================================

%=====================Model co-efficients===========================
a1 = 0.0004; a2 = 0.0001; a3 = 0.0002;
b1 = 0.2; b2 = 0.03; b3 = 0.01;
c1 = 2; c2 = 0.3; c3 = 0.021;
d1 = 0.0040; d2 = 0.006; d3 = 0.001;
par = [a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3];
```

```
kd = 0.00008; % death constant
pl = 50; % inhibition factor
m = 0.11; % Maintenance co-efficient
n = 0.52; % Power co-efficient
si= 266.6; % Input substrate
dt = 2.5; % delta t
%=========================================================================

%=================Initial values of state variables=====================
x0=1.83;
s0=150.0;
p0=2.1;
v0=2.1;
y0=[x0 s0 p0 v0]';
%=========================================================================

%==================Constraints of control variables===================
Tmin = 10;
Fmin = 0;
Tmax = 20;
Fmax = 5.0;
%=========================================================================

%===================Constraints of State variables====================
vmin = 2;
vmax = 4;
pmin = 2.0;
pmax = 50;
smin = 20;
smax = 300;
xmin = 0.83;
xmax =55;
%=========================================================================

%======Calculation of the initial trajectories of state variables=======
t=1: K+1;
[t, y] =ode15s(@(t,y) modelfedbatch_func2(t,y, par, T,F, kd, m, pl, n,
si),t,y0);
x=(y (:, 1))';
s=(y(:,2))';
p=(y(:,3))';
v=(y(:,4))';
figure(1)
plot(t,x,'-k',t,s,':k',t,p,'-.k',t,v,'--k'),
title('Initial trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','volume','Location','NorthEastOut
side')
%=========================================================================

%Calculation of the initial trajectories of the interconnections in time
domain=================================================================
for k = 1:K
    dx(k) = 10.0;
    ds(k) = 138;
    dp(k) = 25.0;
    dv(k) = 2.0;
end
%============Initial trajectories of the control variables============
T=16.0*ones (1, K);
F=5.0*ones (1, K);
%=====================================================================

%=====================================================================
%=======Calculation of the initial trajectories of Error_Lambda_v_(k)===
 for k=1:K
    elx(k) = dx(k)-x(k+1);
    els(k) = ds(k)-s(k+1);
    elp(k) = dp(k)-p(k+1);
    elv(k) = dv(k)-v(k+1);
 end
%=====================================================================

%================START OF THE COORDINATING PROCEDURE================
j=1;
while j <= M2;%Second level iterations
if matlabpool('size')> 0 %Checking if there is a pool of workers open
   matlabpool('close')% Sending a close signal to workers if there is a
pool open
end
matlabpool open jobmanagerconfig5%Opening 32 matlab workers from the
cluster
t0 = clock; % start of the clock on second level calculation
j=1
while j <= M2;%Second level iterations
diary parallelfedbatch %opening blank page for printing results

jm = findResource('scheduler','configuration','jobmanagerconfig5')
disp(get(jm))%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% check the status of
the jobmanager

timingStart = tic;
start = tic;
pjob2=createParallelJob(jm,'Configuration','jobmanagerconfig5'); %create
parallel job from the jobmanager
get(pjob2) % Monitor the status of pjob1

%==========How long it takes to create a job==========================
times.jobCreateTime = toc(start)
description.jobCreateTime = 'Job creation time'
%=====================================================================

Tasks = findTask(pjob2)
[pending running finished] = findTask(pjob1)
set(pjob1,'Configuration','jobmanagerconfig5')
set(pjob,'MinimumNumberOfWorkers',32);
set(pjob,'MaximumNumberOfWorkers',32);
set(pjob1,'FileDependencies',{'subprob2_fedbatch.m'})
get(jm)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Notice how jobmanager
is changing
disp(get(pjob2))
disp('busy workers')
disp(get(jm, 'NumberOfbusyWorkers'))
```

```
%===================STARTING PARALLEL COMPUTING=====================

%Create task- only one with 4 output arguments  and not input arguments,
%measure how long that takes and measure how long it takes to submit the
%job to the cluster
task=createTask(pjob2, @subprob2_fedbatch,6,{})
get(pjob2,'task')

%==========How long it takes to create a task========================
times.taskCreateTime = toc(start)
description.taskCreateTime = 'Task creation time'
%==================================================================
submit(pjob2)

%===========How long it takes to submit a job========================
times.submitTime = toc(start)
description.submitTime = 'Job submission time'
%==================================================================

%Once the job has been submitted, we hope all its tasks execute in
%parallel. Measure how long it takes for all the tasks to start and to
run
%to completion.
waitForState(pjob2,'finished')
times.jobWaitTime = toc(start)
description.pjobWaitTime = 'Job wait time'
get(jm)

%Tasks have now completed, so we are again executing a code in the
matlab
%client. Measure how long it takes to retrieve all the job results
results=getAllOutputArguments(pjob2)
times.resultsTime = toc(start)
description.resultsTime = 'Results retrieval time'
%==================================================================

%===========Verify that the job ran without errors==================
errmsgs = get(pjob2.task, {'ErrorMessage'})
nonempty = ~cellfun(@isempty, errmsgs)
celldisp(errmsgs(nonempty))
%==================================================================

%Measure the total time elapsed from creating the job up to the results
times.totalTime = toc(start)
operationtime  =  etime(clock,t0) %operation  time  for  completion  of
parallel computing
destroy(pjob2)
matlapool close% close matlapool of workers

%Calculation of the improved values of the coordinating variables
%Calculation of the errors e_lambda_V (k)

for (k=1:K)
elx(k) = dx(k) - x(k+1);
els(k) = ds(k) - s(k+1);
elp(k) = dp(k) - p(k+1);
```

```
elv(k) = dv(k) - v(k+1);
end
el=[elx;els;elp;elv];
nn1=norm(elx)+ norm(els)+ norm(elp)+ norm(elv); %Calculation of the norm
of the error
%Check of the convergence of the coordinating procedure for the end of
the
%calculations
if nn1<=eps2
    break
else
%Calculation of the improved values of the coordinating variables and
%interconnection in time domain
for (k =1:K)
lx(k) = lx(k) - alx*elx(k);
ls(k) = ls(k) - als*els(k);
lp(k) = lp(k) - alp*elp(k);
lv(k) = lv(k) - alv*elv(k);

dx(k) = dx(k) + adx*(-lx-mx*elx(k));
ds(k) = ds(k) + ads*(-ls-ms*els(k));
dp(k) = dp(k) + adp*(-lp-mp*elp(k));
dv(k) = dv(k) + adv*(-lp-mp*elp(k));
end
end
l=[lx;ls;lp;lv];
j=j+1
end
%==============================Plot Results============================
figure(2)
t=1:K+1;
plot(t,x,'-k')
title('Optimised Results for the biomass state')
ylabel('g/l');
xlabel('time in samples');
legend('biomass','Location','NorthEastOutside')
figure(3)
plot(t,p,':k')
title('Optimised Results for the product state')
ylabel('g/l');
xlabel('time in samples');
legend('product','Location','NorthEastOutside')
figure(4)
title('Optimised Results for the substrate')
ylabel('g/l');
xlabel('time in samples');
plot(s,'-k')
legend('substrate','Location','NorthEastOutside')
figure(5)
plot(t,v,'-.k')
title('Optimised Results for the volume state')
ylabel('g/l');
xlabel('time in samples');
legend('volume','Location','NorthEastOutside')
t1=1:K;
figure(6)
plot(t1,T,'-k')
```

```
title('Optimised Results for the Temperature T')
ylabel('g/l');
xlabel('time in samples');
legend('Temperature T','Location','NorthEastOutside')
figure(7)
plot(t1,F,'--k')
title('Optimised Results for the input flow rate F')
ylabel('g/l');
xlabel('time in samples');
legend('Input flow rate F','Location','NorthEastOutside')
figure(8)
plot(t1,lx,'-k',t1,ls,':k',t1,lp,'-.k',t1,lv,'--k')
title('Optimised Results for the conjugate variables lambda')
ylabel('g/l');
xlabel('time in samples');
legend('Lambda x','Lambda s','Lambda p','Lambda
v','Location','NorthEastOutside')
%=========================================================================
```

E.2: MATLAB script file – subprob2_fedbatch.m

```
M-File: subprob2_fedbatch.m
%=========================M-FILE DESCRIPTION ===========================
%The function is used to calculate the optimal trajectories of the state
and control by different number of workers in parallel

function [x,s,p,v,T,F]=subprob2_fedbatch

global par T F kd m pl n si  K M1 M2 eps1 mu l x s p v el conmin conmax
statemin statemax astate acon dt y0 x1 s1 p1 v1 T1 F1

%======Calculation of the initial and final point of the subintervals===
dd=K/numlabs
a=(labindex-1)*dd+1
b=labindex*dd
%=========================================================================

for (k=a:b) %Decomposition in time domain
i=1
while i <= M1 %First level iterations

%=========PreAllocation of variables for faster execution==============
dmmaxdT = zeros(1,K);
dKsdT =  zeros(1,K);
dYxsdT = zeros(1,K);
dYpxdT = zeros(1,K);
mmax   = zeros(1,K);
Ks     = zeros(1,K);
Yxs    = zeros(1,K);
Ypx    = zeros(1,K);
ff     = zeros(1,K);
fx     = zeros(1,K);
fs     = zeros(1,K);
fp     = zeros(1,K);
f1     = zeros(1,K);
f2     = zeros(1,K);
eex    = zeros(1,K);
```

```
ees    = zeros(1,K);
eep    = zeros(1,K);
ex     = zeros(1,K);
es     = zeros(1,K);
ep     = zeros(1,K);
eT = zeros(1,K);
eF = zeros(1,K);
%======================================================================

%=====================For the temperature==============================
dmmaxdT(k) = par(2)+2*par(3)*T(k); %Equations 5.10
dKsdT(k)=par(5)+2*par(6)*T(k);
dYxsdT(k)=par(8)+2*par(9)*T(k);
dYpxdT(k)=par(11)+2*par(12)*T(k);
%======================================================================

%=============Calculation of the kinetic coefficients==================
mmax(k)=par(1)+par(2)*T(k)+par(3)*(T(k)^2);   % Equation 2.72
Ks(k)=par(4)+par(5)*T(k)+par(6)*(T(k)^2);     % Equation 2.73
Yxs(k)=par(7)+par(8)*T(k)+par(9)*(T(k)^2);    % Equation 2.74
Ypx(k)=par(10)+par(11)*T(k)+par(12)*(T(k)^2); % Equation 2.75
%======================================================================

%Calculation of the expressions lambda_v_(k), k=0,K-1, v = x,s,p,v and
lamda1(k), lamda2(k)
ff(k)=dt*mmax(k)*((1-p(k)/pl)^n);
fx(k)=ff(k)*s(k)/(Ks(k) + s(k));    % Equation 5.25
fs(k)=ff(k)*Ks(k)*x(k)/((Ks(k) + s(k))^2); %Equation 5.27
fp(k)=dt*mmax(k)*((1-p(k)/pl)^(n-1))*x(k)*s(k)/(pl*(Ks(k)+s(k)));%
Equation 5.26
fv(k)=dt*F(k)/(v(k)^2); %Equation 5.29
ff1(k)=ff(k)*x(k)*s(k)/(Ks(k) + s(k)); %Equation 5.28
f2(k)=ff1(k)/(Ks(k) + s(k)); % Equation 5.30
f1(k) = ff1(k)/mmax(k)
%======================================================================

%Calculation of the expressions ee_v_(k)=lambda(k)-m_v*E_lambda_v_(k)
eex(k)=l(1,k)-mu(1)*el(1,k);
ees(k)=l(2,k)-mu(2)*el(2,k);
eep(k)=l(3,k)-mu(3)*el(3,k);
eev(k)=l(4,k)-mu(4)*el(4,k);
%======================================================================

%=======Calculation of the gradients for state variables===============
ex(k)=(1+fx(k)-dt*kd+dt*F(k)/v(k))*eex(k)+((fx(k)/Yxs(k))-
dt*m)*ees(k)+fx(k)*Ypx(k) * eep(k);%Equation 5.31
es(k)=fs(k)*eex(k)+(1-fs(k)/Yxs(k)-dt*F(k)/v(k))*ees(k)+Ypx(k)*
fs(k)*eep(k);%Equation 5.32
ep(k)=-fp(k)*eex(k)+(fp(k)/Yxs(k))*ees(k)+(1-fp(k)-
dt*F(k)/v(k))*eep(k);%Equation 5.33
ev(k)=fv(k)*x(k)*eex(k)-(fv(k)*(si-s(k))*ees(k))+(fv(k)*p(k)*eep(k))+
eev(k);
%======================================================================

%=========Calculation of the gradients for control variables==========
eF(k)=dt*x(k)/v(k)*eex(k)+(dt*(si-s(k))/v(k))*ees(k)-
(dt*p(k)/v(k))*eep(k)+dt*eev(k);%Equation 5.34
```

```
eT(k) = (f1(k)*dmmaxdT(k)+f2(k)*dKsdT(k))*eex(k) + ((-
f1(k)/(Yxs(k))^2)*(dmmaxdT(k)*Yxs(k)-mmax(k)*dYxsdT(k))+
(f2(k)/Yxs(k))*dKsdT(k)) *ees(k) +
(f1(k)*(dYpxdT(k)*mmax(k)+Ypx(k)*dmmaxdT(k))-
Ypx(k)*f2(k)*dKsdT(k))*eep(k);
%Equation 5.35
n1=[ex(k) es(k) ep(k) ev(k) eF(k) eT(k)]';
%=====================================================================

nn = norm(n1) %Calculation norm of gradients
if nn <=eps1%Check the condition for end of the iterations on the first
level
break
%Calculation of the new values of the state and control variables
elseif k == 1
s1(1,k,labindex)=y0(2);
p1(1,k,labindex)=y0(3);
v1(1,k,labindex)=y0(4);
T1(1,k,labindex) = T(k) + acon(1)*eT(k);
F1(1,k,labindex) = F(k) + acon(4)*eF(k);
else
x1(1,k,labindex) = x(k) + astate(1)*ex(k);
s1(1,k,labindex) = s(k) + astate(2)*es(k);
p1(1,k,labindex) = p(k) + astate(3)*ep(k);
v1(1,k,labindex) = v(k) + astate(4)*ev(k);
T1(1,k,labindex) = T(k) + acon(1)*eT(k)
F1(1,k,labindex) = F(k) + acon(2)*eF(k);
end
i=i+1
end
%=====================Check against constraints=====================
if x1(1,k,labindex) < statemin(1)
    x1(1,k,labindex) = statemin(1);
elseif x1(1,k,labindex) > statemax(1)
    x1(1,k,labindex) = statemax(1);
end
if s1(1,k,labindex) <statemin(2)
    s1(1,k,labindex) = statemin(2);
elseif s1(1,k,labindex) > statemax(2)
    s1(1,k,labindex) = statemax(2);
end
if p1(1,k,labindex) < statemin(3)
    p1(1,k,labindex) = statemin(3);
elseif p1(1,k,labindex) > statemax(3)
    p1(1,k,labindex) = statemax(3);
end
if v1(1,k,labindex) < statemin(4)
    v1(1,k,labindex) = statemin(4);
elseif v1(1,k,labindex) > statemax(4)
    v1(1,k,labindex) = statemax(4);
end

if T1(1,k,labindex) < conmin(1)
    T1(1,k,labindex) = conmin(1);
elseif T1(1,k,labindex) > conmax(1)
   T1(1,k,labindex) = conmax(1);
end
```

```
if F1(1,k,labindex) < conmin(2)
    F1(1,k,labindex) = conmin(2);
elseif F1(1,k,labindex) > conmax(2)
    F1(1,k,labindex) = conmax(2);
end
end  %end of the iterations in the subproblem time interval (a,b)
%=======================================================================

labBarrier
if labindex==1
x2=reshape(x1,1,numel(size(x1)));
s2=reshape(s1,1,numel(size(s1)));
p2=reshape(p1,1,numel(size(p1)));
T2=reshape(T1,1,numel(size(T1)))
F2=reshape(F1,1,numel(size(F1)));

x=x2([a:b,labindex])
s=s2([a:b,labindex])
p=p2([a:b,labindex])
T=T2([a:b,labindex])
F=F2([a:b,labindex])

%======Calculation of the final value of x(k)=x(K+1)=xK1;sK1,pK1========
sK1=s(K)-f1(K)/Yxs(K)-dt*m*x(K)+dt*F(K)*(si-s(K))/v(K);
pK1=p(K)+f1(K)*Ypx(K)-dt*F(K)*p(K)/v(K);
vK1=v(K)+dt*F(K);
if xK1 <= statemin(1)
xK1 = statemin(1);
elseif xK1 >=statemax(1)
xK1 = statemax(1);
end
x(49)=xK1
if sK1 <= statemin(2)
    sK1 = statemin(2);
elseif sK1 >= statemax(2)
    sK1 = statemax(2);
end
s(49)=sK1
if pK1 <= statemin(3)
    pK1 = statemin(3);
elseif pK1 >= statemax(3)
    pK1=statemax(3)
end
 p(49)=pK1
if vK1 <= statemin(4)
    vK1 = statemin(4);
elseif vK1 >= statemax(4)
    vK1 = statemax(4);
end
v(49)=vK1
end
%=======================================================================
```

E.3: MATLAB script file – modelfedbatch_func2.m

```
M-File: modelfedbatch_func2.m
%========================M-FILE DESCRIPTION ==========================
%The function is used to calculate the trajectories of the derivatives
of the state space vector

function ydot=modelfedbatch_func2(t,y,par, T, F, kd, m, pl, n, si)

%========================Model_variables==========================
ydot(1)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))-(kd*y(1))-
F*y(1)/y(4)); %Equation 2.84
ydot(2)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))/-
(par(7)+par(8)*T+par(9)*(T*T))-m*y(1)-F*(si-y(2))/y(4));%Equation 2.85
ydot(3)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T))+y(2))*(par(10)+p
ar(11)*T+par(12)*(T*T))-F*y(3)/y(4);%Equation 2.86
ydot(4)=F; %Equation 2.87
ydot=[ydot(1) ydot(2) ydot(3) ydot(4)]';
%=================================================================
```

APPENDIX F: MATLAB PARALLEL PROGRAM FOR CALCULATION OF CONTROL AND STATE FOR CONTINUOS FERMENTATION PROCESS

F.1: MATLAB script file – continuous_parallel.m

```
% M-File: continuous_parallel.m
% ==========================M-FILE DESCRIPTION ======================
%  Calculation  of  state  and  control  trajectories  for  continuous
fermentation
% process

clear all% clear workspace
global par T D kd m pl n si  K M1 M2 eps1 mu l x s p el conmin conmax
statemin statemax astate acon dt y0 x1 s1 p1 T1 D1
%========================Set Initial values========================
M1 = 100; M2 = 50; %Max number of iterations of the first and second
levels
K = 256; %Number of steps in the optimization horizon
ax = 0.001; as = 0.001; ap = 0.001; %Steps of the gradient procedures
for calculation of the State variables
aT = 0.05; aD = 0.05; %Steps of the gradient procedures for calculation
of the Control variables
alx = 0.001; als = 0.001; alp = 0.001; %Step of the gradient procedures
for calculation of the Conjugate variables
adx = 0.001; ads = 0.001; adp = 0.001; %Steps of the gradient procedures
for calculation of the Interconnections in time
eps1=0.001; %tolerances for stop of the calculation of the first levels
eps2=0.001; %tolerances for stop of the calculation of the second levels
%==================================================================

%================Initial values of the penalty coefficients==========
mx=0.0010;
ms=0.0010;
mp=0.0010;
%==================================================================
dt = 0.01967; %Initial value of the sampling period

%==============Initial trajectories for the conjugate variables========
for k = 1: K+1
lx (k) =0.001;
ls (k) =0.005;
lp (k) =0.002;
end
%==================================================================

%===========Initial trajectories of the control variables============
T = 16;
D = 0;
%==================================================================

%=========================Model co-efficients=======================
a1 = 0.0004; a2 = 0.0001; a3 = 0.0002;
b1 = 0.2; b2 = 0.03; b3 = 0.01;
c1 = 2; c2 = 0.3; c3 = 0.021;
d1 = 0.0040; d2 = 0.006; d3 = 0.001;
par = [a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3];
```

```
kd = 0.00008; % death constant
pl = 50; % inhibition factor
m = 0.11; % Maintenance co-efficient
n = 0.52; % Power co-efficient
si= 266.6; % Input substrate
Ks = 0.5; %Initial value of Monod constant for calculation of dtmin and
dtmax
mmax = 0.5; %Initial value of maximum growth rate for calculation of
dtmin and dtmax
Yxs = 0.175; Initial value of stochiometry coefficient for biomass
growth for calculation of dtmin and dtmax
Ypx = 0.421; Initial value of stochiometry coefficient for biomass
growth for calculation of dtmin and dtmax
%=====================================================================

%====================Initial values of state variables================
x0=1.83;
s0=150.0;
p0= 0;
y0=[x0 s0 p0]';
%=====================================================================

%==================steady state values of state variables=============
xbar = 100*ones (1, K+1);
sbar = 30*ones (1, K+1);
pbar = 100*ones (1, K+1);
%=====================================================================

%======================Constraints of control variables===============
Tmin = 10;
Dmin = 0;
Tmax = 25;
Dmax = 5;
%=====================================================================

%=======================Constraints of State variables================
xmin = 0.83;
xmax = 100;
smin = 5;
smax = 300;
pmin = 0;
pmax = 100;
%=====================================================================

%====================Calculation of delta min and max=================
dtmax = 0.01*(Ks + smax)/ ((mmax-kd-Dmax)*smax - Ks*(kd + Dmax));
dtmin = 0.01*(Ks + smin)/ ((mmax-kd-Dmin)*smin - Ks*(kd + Dmin));
%=====================================================================


%=====Calculation of the initial trajectories of state variables======
t=1: K+1;
[t, y] =ode15s (@ (t, y) modelcontinuos_func2 (t, y, par, T, D, kd, m,
pl, n, si), t, y0);
x=(y (:, 1))';
s=(y(:,2))';
p=(y(:,3))';
```

```
figure(1)
t = 1:K+1;
plot(t,x,'-k',t,s,':k',t,p,'-.k')
title('Initial trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','Location','NorthEastOutside')

%=========================================================================

%Calculation of the initial trajectories of the interconnections in time
domain=====================================================================
for k = 1:K
dx(k) = 10.0;
ds(k) = 138;
dp(k) = 25.0;
end
%=========================================================================
%============Initial trajectories of the control variables=============
T = 16*ones (1, K);
D = 1.25*ones (1, K);
%=========================================================================

%=======Calculation of the initial trajectories of el_v_(k,v=s,x,p======
 for k=1:K
 elx(k) = dx(k)-x(k+1);
 els(k) = ds(k)-s(k+1);
 elp(k) = dp(k)-p(k+1);
 end
%=========================================================================

 %=============Calculation of elx(K) at the last point=================
 elx(K) = x(K+1)- xbar(K+1);
 els(k) = s(K+1)- sbar(K+1);
 elp(k) = p(K+1)- pbar(K+1);
 %========================================================================

% ====================START OF THE COORDINATING PROCEDURE==============
j=1;
while j <= M2;%Second level iterations
if matlabpool('size')> 0 %Checking if there is a pool of workers open
   matlabpool('close')% Sending a close signal to workers if there is a
pool open
end
matlabpool open jobmanagerconfig5%Opening 32 matlab workers from the
cluster
t0 = clock; % start of the clock on second level calculation
j=1
while j <= M2;%Second level iterations
diary parallelfedbatch %opening blank page for printing results

jm = findResource('scheduler','configuration','jobmanagerconfig5')
disp(get(jm))%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% check the status of
the jobmanager

timingStart = tic;
start = tic;
```

```
pjob3=createParallelJob(jm,'Configuration','jobmanagerconfig5'); %create
parallel job from the jobmanager
get(pjob3) % Monitor the status of pjob1

%=============How long it takes to create a job========================
times.jobCreateTime = toc(start)
description.jobCreateTime = 'Job creation time'
%=====================================================================

Tasks = findTask(pjob2)
[pending running finished] = findTask(pjob1)
set(pjob3,'Configuration','jobmanagerconfig5')
set(pjob3,'MinimumNumberOfWorkers',32);
set(pjob3,'MaximumNumberOfWorkers',32);
set(pjob3,'FileDependencies',{'subprob2_fedbatch.m'})
get(jm)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Notice  how  jobmanager
is changing
disp(get(pjob3))
disp('busy workers')
disp(get(jm, 'NumberOfbusyWorkers'))

%=======================STARTING PARALLEL COMPUTING====================

%Create task- only one with 5 output arguments  and not input arguments,
%measure how long that takes and measure how long it takes to submit the
%job to the cluster
task=createTask(pjob3, @subprob2_batch,5,{})
get(pjob2,'task')

%============How long it takes to create a task=======================
times.taskCreateTime = toc(start)
description.taskCreateTime = 'Task creation time'
%=====================================================================
submit(pjob3)

%============How long it takes to submit a job=======================
times.submitTime = toc(start)
description.submitTime = 'Job submission time'
%=====================================================================

%Once the job has been submitted, we hope all its tasks execute in
%parallel. Measure how long it takes for all the tasks to start and to
run
%to completion.
waitForState(pjob3,'finished')
times.jobWaitTime = toc(start)
description.pjobWaitTime = 'Job wait time'
get(jm)

%Tasks have now completed, so we are again executing a code in the
matlab
%client. Measure how long it takes to retrieve all the job results
results=getAllOutputArguments(pjob3)
times.resultsTime = toc(start)
description.resultsTime = 'Results retrieval time'
%=====================================================================
%===============Verify that the job ran without errors=================
errmsgs = get(pjob2.task, {'ErrorMessage'})
nonempty = ~cellfun(@isempty, errmsgs)
celldisp(errmsgs(nonempty))
%=====================================================================

%Measure the total time elapsed from creating the job up to the results
times.totalTime = toc(start)
operationtime=etime(clock,t0)%operation time for completion of parallel
computing
destroy(pjob3)
matlapool close% close matlapool of workers

%Calculation on the coordinating level
% Calculation of the new values of the gradients of the coordinating
% variables using the values of the calculated state and control
variables
for k=1:K
elx(K+1) = x(K+1)-xbar(K+1);
els(K+1) = s(K+1)-sbar(K+1);
elp(K+1) = p(K+1)-pbar(K+1);
end
nn1=norm(elx)+ norm(els)+ norm(elp);

if nn1<=eps2
break
else
% Calculation of the improved values of lv, v = x,s,p
for k =1:K+1
lx(k) = lx(k) + alx*elx(k);
ls(k) = ls(k) + als*els(k);
lp(k) = lp(k) + alp*elp(k);
end
%Calculation of the improved values of the sampling period
mmax(K)=par(1)+par(2)*T(K)+par(3)*(T(K)^2); % Equation 2.72
Ks(K)=par(4)+par(5)*T(K)+par(6)*(T(K)^2);   % Equation 2.73
Yxs(K)=par(7)+par(8)*T(K)+par(9)*(T(K)^2);  % Equation 2.74
Ypx(K)=par(10)+par(11)*T(K)+par(12)*(T(K)^2); % Equation 2.75
eex(K)=lx(K)+ mx*(x(K)-xbar(K));
ees(K)=ls(K)+ ms*(s(K)-sbar(K));
eep(K)=lp(K)+ mp*(p(K)-pbar(K));
ff1(K) = mmax(K) *((1-pk/pl)^n);
f3(K) = ff1(K)*x(K)*s(K)/(Ks(K) + s(K));% Equation 6.24
dt =  (-K -(f3(K)-(kd*x(K)-D(K)*x(K))*eex(K)+(f3(K)/Yxs(K)+ m*x(K)-
D(K)*(si-s(K))*ees(K)-(Ypx(K)*f3(K)-D(K)*p(K))*eep(K))/(mx*(f3(K)-
kd*x(K)-D(K)*x(K))^2 + ms*(f3(K)/Yxs(K)+ m*x(K)-D(K)*(si-s(K))^2+
mp*(Ypx(K)*f3(K))-D(K)*p(K))^2);    % Equation 6.63

if dt <= dtmin
dt = dtmin;
elseif dt >=dtmax
dt = dtmax;
%end
for k = 1:K
dx(k) = dx(k) - adx*(-mx*elx(k)-lx(k));
ds(k) = ds(k) - ads*(-ms*els(k)-ls(k));
dp(k) = dp(k) - adp*(-mp*elp(k)-lp(k));
```

```
en
end
j = j+1;
end % End of the second level iteration
toc;
figure(2)
t=1:K+1;
plot(t,x,'-k')
title('Optimised Results for the biomass state')
ylabel('g/l');
xlabel('time in samples');
legend('biomass','Location','NorthEastOutside')
figure(3)
plot(t,p,':k')
title('Optimised Results for the product state')
ylabel('g/l');
xlabel('time in samples');
legend('product','Location','NorthEastOutside')
figure(4)
plot(t,s,'-k')
title('Optimised Results for the substrate')
ylabel('g/l');
xlabel('time in samples');
legend('substrate','Location','NorthEastOutside')
t1=1:K;
figure(5)
plot(t1,T,'-k')
title('Optimised Results for the Temperature T')
ylabel('g/l');
xlabel('time in samples');
legend('Temperature T','Location','NorthEastOutside')
t1=1:K;
figure(6)
plot(t1,D,'--k')
title('Optimised Results for the Dilution rate D')
ylabel('g/l');
xlabel('time in samples');
legend('Dilution rate D','Location','NorthEastOutside')
figure(7)
t1=1:K+1;
plot(t1,lx,'-k',t1,ls,':k',t1,lp,'-.k')
title('Optimised Results for the conjugate variables lambda')
ylabel('g/l');
xlabel('time in samples');
legend('Lambda x','Lambda s','Lambda p','Location','NorthEastOutside')
figure(8)
t = 1:K+1;
plot(t,x,'-k',t,s,':k',t,p,'-.k')
title('Optimal trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','Location','NorthEastOutside')
%=================================================================
```

F.2: MATLAB script file – subprob2_continuous.m

```
M-File: subprob2_continuos.m
```

```
%========================M-FILE DESCRIPTION =========================
%The function is used to calculate the optimal trajectories of the state
and control by different number of workers in parallel

function[x,s,p,v,T,D]=subprob2_continuous

global par T D kd m pl n si  K M1 M2 eps1 mu 1 x s p el conmin conmax
statemin statemax astate acon dt y0 x1 s1 p1 T1 D1

%======Calculation of the initial and final point of the subintervals===
dd=K/numlabs
a=(labindex-1)*dd+1
b=labindex*dd
%=====================================================================

for (k=a:b) %Decomposition in time domain
i=1
while i <= M1 %First level iterations

%=========PreAllocation of variables for faster execution=============
dmmaxdT = zeros(1,K);
dKsdT =  zeros(1,K);
dYxsdT = zeros(1,K);
dYpxdT = zeros(1,K);
mmax   = zeros(1,K);
Ks     = zeros(1,K);
Yxs    = zeros(1,K);
Ypx    = zeros(1,K);
ff     = zeros(1,K);
fx     = zeros(1,K);
fs     = zeros(1,K);
fp     = zeros(1,K);
f1     = zeros(1,K);
f2     = zeros(1,K);
eex    = zeros(1,K);
ees    = zeros(1,K);
eep    = zeros(1,K);
ex     = zeros(1,K);
es     = zeros(1,K);
ep     = zeros(1,K);
eT     = zeros(1,K);
eD     = zeros(1,K);
%=====================================================================

%====================For the temperature=============================
dmmaxdT(k) = par(2)+2*par(3)*T(k); %Equations 5.10
dKsdT(k)=par(5)+2*par(6)*T(k);
dYxsdT(k)=par(8)+2*par(9)*T(k);
dYpxdT(k)=par(11)+2*par(12)*T(k);
%=====================================================================

%============Calculation of the kinetic coefficients=================
mmax(k)=par(1)+par(2)*T(k)+par(3)*(T(k)^2);    % Equation 2.72
Ks(k)=par(4)+par(5)*T(k)+par(6)*(T(k)^2);      % Equation 2.73
Yxs(k)=par(7)+par(8)*T(k)+par(9)*(T(k)^2);% Equation 2.74
Ypx(k)=par(10)+par(11)*T(k)+par(12)*(T(k)^2);%Equation2.75
%=====================================================================
%=======Calculation of the expressions from equation (6.38)-(6.42=======
pk=p(k);
ff(k)=dt*mmax(k)  *((1-pk/pl)^n);
ff2(k)= dt*((1-pk/pl)^n);
fx(k)=ff(k)*s(k)/(Ks(k)  + s(k));  % Equation 6.38
fs(k)=ff(k)*Ks(k)  * x(k)/((Ks(k)  + s(k))^2); %Equation 6.40
fp(k)=-dt*mmax(k)*((1-p(k)/pl)^(n-1))*n*x(k)*s(k)/(pl*(Ks(k)  + s(k))); %
Equation 6.39
f1(k)=ff2(k)*x(k)*s(k)/(Ks(k)  + s(k)); %Equation 6.41
f2(k)=-f1(k)*mmax(k)/(Ks(k)  + s(k)); % Equation 6.42
%=======================================================================

%Calculation of the expressions ee_v_(k)=lambda(k)-m_v*E_lambda_v_(k)
eex(k)=l(1,k)-mu(1)*el(1,k);
ees(k)=l(2,k)-mu(2)*el(2,k);
eep(k)=l(3,k)-mu(3)*el(3,k);
%=======================================================================

%=======Calculation of the gradients for state variables===============
ex(k)=(1+fx(k)-dt*kd-dt*D(k))*eex(k)+((fx(k)/Yxs(k))-dt*m)*ees(k)+
fx(k)*Ypx(k)*eep(k);%Equation 6.43
es(k)=fs(k)*eex(k)+(1-fs(k)/Yxs(k)-dt*D(k))*ees(k)+Ypx(k)*
fs(k)*eep(k);%Equation 6.44
ep(k)=fp(k)*eex(k)-(fp(k)/Yxs(k))*ees(k)+(1+fp(k)-
dt*D(k))*eep(k);%Equation 6.45
%=======================================================================

%==========Calculation of the gradients for control variables==========
eD(k)=-dt*x(k)*eex(k)+(dt*(si-s(k))*ees(k))-  (dt*p(k)*eep(k)); %Equation
6.47
eT(k)  =  (f1(k)*dmmaxdT(k)+f2(k)*dKsdT(k))*eex(k)  +  ((-
f1(k)/(Yxs(k))^2)*(dmmaxdT(k)*Yxs(k)-mmax(k)*dYxsdT(k))+
(f2(k)/Yxs(k))*dKsdT(k))  *ees(k)  +
(f1(k)*(dYpxdT(k)*mmax(k)+Ypx(k)*dmmaxdT(k))-
Ypx(k)*f2(k)*dKsdT(k))*eep(k);
%Equation 6.46
n1=[ex(k)  es(k)  ep(k)  eD(k)  eT(k)]';
%=======================================================================

nn = norm(n1) %Calculation norm of gradients
if nn <=eps1%Check the condition for end of the iterations on the first
level
break
%Calculation of the new values of the state and control variables
elseif k == 1
s1(1,k,labindex)=y0(2);
p1(1,k,labindex)=y0(3);
x1(1,k,labindex)=y0(1);
T1(1,k,labindex)  =  T(k)  +  acon(1)*eT(k);
D1(1,k,labindex)  =  D(k)  +  acon(2)*eD(k);
else
x1(1,k,labindex)  =  x(k)  +  astate(1)*ex(k);
s1(1,k,labindex)  =  s(k)  +  astate(2)*es(k);
p1(1,k,labindex)  =  p(k)  +  astate(3)*ep(k);
T1(1,k,labindex)  =  T(k)  +  acon(1)*eT(k)
D1(1,k,labindex)  =  D(k)  +  acon(2)*eD(k);
```

```
end
i=i+1
end
%======================Check against constraints=====================

if x1(1,k,labindex) < statemin(1)
    x1(1,k,labindex) = statemin(1);
elseif x1(1,k,labindex) > statemax(1)
    x1(1,k,labindex) = statemax(1);
end
if s1(1,k,labindex) <statemin(2)
    s1(1,k,labindex) = statemin(2);
elseif s1(1,k,labindex) > statemax(2)
    s1(1,k,labindex) = statemax(2);
end
if p1(1,k,labindex) < statemin(3)
    p1(1,k,labindex) = statemin(3);
elseif p1(1,k,labindex) > statemax(3)
    p1(1,k,labindex) = statemax(3);
end
if T1(1,k,labindex) < conmin(1)
    T1(1,k,labindex) = conmin(1);
elseif T1(1,k,labindex) > conmax(1)
   T1(1,k,labindex) = conmax(1);
end
if D1(1,k,labindex) < conmin(2)
    D1(1,k,labindex) = conmin(2);
elseif D1(1,k,labindex) > conmax(2)
    D1(1,k,labindex) = conmax(2);
end
end  %end of the iterations in the subproblem time interval (a,b)
%==================================================================
labBarrier
if labindex==1
x2=reshape(x1,1,numel(size(x1)));
s2=reshape(s1,1,numel(size(s1)));
p2=reshape(p1,1,numel(size(p1)));
T2=reshape(T1,1,numel(size(T1)))
D2=reshape(D1,1,numel(size(D1)));

x=x2([a:b,labindex])
s=s2([a:b,labindex])
p=p2([a:b,labindex])
T=T2([a:b,labindex])
D=D2([a:b,labindex])

%======Calculation of the final value of x(k)=x(K+1)=xK1;sK1,pK1=======
K1 = K+1;
ex(K1)= lx(K1)+ mx*(x(K1)-xbar(K1));
es(K1)= ls(K1)+ ms*(s(K1)-sbar(K1));
ep(K1)= lp(K1)+ mp*(p(K1)-pbar(K1));

%Calculation of the final value of state
xK1 = x(K1)  - alx*ex(K1);
sK1 = s(K1)  - als*es(K1);
pK1 = p(K1)  - alp*ep(K1);
% Check the constraints at the final point
```

```
if xK1 <= xmin
    xK1 = xmin;
elseif xK1 >=xmax
    xK1 = xmax;
end
if sK1 <= smin
    sK1 = smin;
elseif sK1 >= smax
    sK1 = smax;
end
if pK1 <= pmin
    pK1 = pmin;
elseif pK1 >= pmax
    pK1=pmax;
end
%========================================================================
```

F.3: MATLAB script file – modelcontinuous_func2.m

```
M-File: modelcontinuous_func2.m

%=======================M-FILE DESCRIPTION =======================

%The function is used to calculate the trajectories of the derivatives
of the state space vector

function ydot= modelcontinuos_func2(t,y, par, T, D, kd, m, pl, n, si)

%===============================Model_variables========================

ydot(1)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))-(kd*y(1))-
D*y(1)); %Equation 2.93

ydot(2)=(par(1)+par(2)*T+par(3)*(T*T))*((1-
(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T)+y(2))/-
(par(7)+par(8)*T+par(9)*(T*T))-m*y(1)+ D*(si-y(2)));%Equation 2.94

ydot(3)=(par(1)+par(2)*T+par(3)*(T*T))*(par(10)+par(11)*T+par(12)*(T*T))
*((1-(y(3)/pl))^n)*y(1)*y(2)/((par(4)+par(5)*T+par(6)*(T*T))+y(2))-
(D*y(3));%Equation 2.95

ydot=[ydot(1) ydot(2) ydot(3)]';

%========================================================================
```