DEVELOPMENT AND APPLICATIONS OF NOVEL OPTIMAL CONTROL ALGORITHMS

By

Victor Manuel Becerra

A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

CONTROL ENGINEERING RESEARCH CENTRE
DEPARTMENT OF ELECTRICAL, ELECTRONIC
AND INFORMATION ENGINEERING

CITY UNIVERSITY, LONDON

FEBRUARY, 1994

To

Mamá, Eybi, Teresa, Elvira and Bárbara

TABLE OF CONTENTS

LIST OF TABLES

Table No. Page

LIST OF FIGURES

Figure No. Page

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to my research supervisor, Prof. Peter David Roberts, for his guidance, advice, understanding and kindness given throughout this work. His professionalism and high standards have provided me with a source of inspiration.

I am indebted to Mr. G.W. Griffiths and Dr. J. Lin from SAST Ltd. for the interest they showed in my project, for their valuable suggestions and for having allowed me to implement new developments using the facilities at SAST and to test them using industrial scale process simulations.

I am grateful to the following organizations for their financial support which made this project a reality:

* CEPET (Centro de Formación y Adiestramiento de Petróleos de Venezuela y sus Filiales).

* SAST Ltd., U.K. (Special Analysis and Simulation Technology).

* Gilchrist Educational Trust, U.K.

Special thanks to Mrs. J. Rivellini from City University, for her kindness and caring attitude.

Last, but not least, I want to thank my wife, Eybi, for her constant support, her patience, help, love and understanding.

DECLARATION

The author grants power of discretion to the University Librarian to allow this thesis to be copied in whole or part without further reference to him. This permission covers only single copies made for study purposes only, subject to normal conditions of acknowledgement.

ABSTRACT

The main concern of this thesis is to advance and improve the existing knowledge of a dynamic optimal control technique known as DISOPE, so as to make it attractive on one hand for its implementation in the process industry and, on the other hand, as a novel nonlinear optimal control algorithm. The main feature of the technique is that it has been designed so as to achieve the correct optimum of the process in spite of inaccuracies in the mathematical model employed in the computations.

Several extensions of the basic continuous time DISOPE technique are proposed in this work. For the development of the algorithms, emphasis is placed on making the techniques implementable in digital computer based industrial process control problems. These extensions include discrete-time, and set-point tracking versions, extensions for handling control and state dependent inequality constraints, and a hierarchical version.

Applications of DISOPE are proposed in the following areas: nonlinear predictive control, predictive optimising control based on adaptive state-space linear dynamic models, and batch process optimisation.

All the algorithms and techniques proposed in this thesis have been implemented in software and tested with relevant simulations. These studies include dynamic simulations of low order chemical reaction systems and studies on the dynamic optimisation of an industrial-scale multicomponent distillation column using a rigorous process simulator.

Comparisons with existing techniques are provided and suggestions are made for future research in the area treated in this thesis.

LIST OF SYMBOLS

Symbol	Description
n	Order of the system
m	Number of control inputs
n_o	Number of output variables
u	Control vector
x	State vector
x_0	Initial state vector
f^*	Real state mapping
f	Model-based state mapping
L^*	Real performance measure function
L	Model-based performance measure function
J^*	Real performance index
J_m	MOP performance index
J_e	EOP performance index
J_M	MMOP performance index
H^*	Real Hamiltonian
H	Model-based or augmented Hamiltonian
p	Costate vector
φ	Terminal weighting function
A	Model-based system matrix
B	Model-based control distribution matrix
Q $(\bar{Q})$	Intermediate state weighting matrix (augmented)
R $(\bar{R})$	Control weighting matrix (augmented)
Φ	Terminal state weighting matrix
α	State related time-varying parameter vector
γ	Performance related time-varying parameter

z	State separation vector
v	Control separation vector
$\hat{p}$	Costate separation vector
μ	Lagrange multiplier vector
ξ	Lagrange multiplier
λ ($\bar{\lambda}$)	Lagrange multiplier vector (augmented)
β ($\bar{\beta}$)	Lagrange multiplier vector (augmented)
t	time
k	time index
$[t_0, t_f]$	time interval
$[N_0, N_f]$	discrete time interval
K	Continuous time Riccati matrix
$k(t)$	Continuous time Riccati vector
S	Discrete time Riccati matrix
h	Discrete time Riccati vector
G	State feedback gain matrix
g	Driving input vector
W	Time varying matrix
F	Time varying matrix
ν	Lagrange multiplier
r_1	Control convexification factor
r_2	State convexification factor
k_v	Control relaxation gain
k_z	State relaxation gain
k_p	Costate relaxation gain
ε_v	Control convergence tolerance
$C(u,t)$	Input dependent constraint mapping
Θ	Kuhn-Tucker multiplier

Ω	Set of admissible control trajectories
$u_{\min}$, $u_{\max}$	Control bounds
SAT, *sat*	Saturation functions
ζ	State distribution mapping i-th subsystem
θ_i	Interactions distribution mapping i-th subsystem
w_i	Interaction input vector for i-th subsystem
M_{ij}	Coupling matrix from subsystem j to subsystem i
C_i	Interaction distribution matrix i-th subsystem
Ω_i	Lagrange multiplier for interactions, i-th subsystem
ε_w	Convergence tolerance for interactions
y	Output vector
$\zeta(x)$	Output mapping
C	Output matrix
r_0	Output reference trajectory
J_o	SCROP performance index
L_o^*	SCROP performance measure function
Ψ	State dependent constraint mapping
ρ	Penalty factor
P_ε	Smoothed penalty relaxation function
ε	A small scalar value
k_r	Convexification gain factor
$\hat{A}$, $\hat{B}$	Estimated jacobian matrices
$\hat{x}$	Estimated state vector
F^*	Continuous state mapping
t'	Continuous time variable
N_c	Control horizon
N_p, N	Prediction horizon

J_r	Receding horizon performance index
d_y	Prediction error
G_f	Noise distribution matrix
y_m	Measured output
w_p	Process noise vector
v_m	Measurement noise vector
Q_f	Process noise covariance matrix
R_f	Measurement noise covariance matrix
$\bar{x}_0$	Expected value of initial state
P_0	Initial state covariance matrix
A_f, C_f	Jacobian matrices
K_f	Kalman gain
$\hat{\theta}$	Vector of estimated parameters
$\hat{\theta}_0$	Expected value of initial parameter vector
P_θ	Parameter vector covariance matrix
η	Pseudo-noise vector
Q_θ	Pseudo-noise covariance matrix
N_u	Moving horizon identifier update period
N_d	Moving horizon identifier data length
q^{-1}	Backward shift operator
$A(q^{-1})$	Polynomial matrix
$B(q^{-1})$	Polynomial matrix
$C(q^{-1})$	Polynomial matrix
A_1, B_1, C_1	Matrix polynomial coefficients
N^*	Steady state objective function
g^*	Steady-state mapping
$\mathfrak{L}$	Lagrangian function

ϕ	Linear terminal weighting vector
A^* , B^*	Exact jacobian matrices
$\Re^n$	n-dimensional real space
I_n	Identity matrix of order n
x	$[x_1 x_2 \ldots x_n]^T$ where x_j are scalars
$\nabla_x H$	$\left[\frac{\partial H}{\partial x}\right]^T = \left[\frac{\partial H}{\partial x_1} \ldots \frac{\partial H}{\partial x_n}\right]^T$ where H and x_j are scalars
$\frac{\partial f}{\partial x}$	$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ . & \cdots & . \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$ where f_i and x_j are scalars

LIST OF ABBREVIATIONS

Abbreviation	Description
ISOPE	Integrated system optimisation and parameter estimation
DISOPE	Dynamic ISOPE
ROP	Real optimal control problem
MOP	Model-based optimal control problem
EOP	Expanded optimal control problem
MMOP	Modified model-based optimal control problem
SCROP	State constrained real optimal control problem
EKF	Extended Kalman Filter
DMC	Dynamic Matrix Control
MAC	Model Algorithmic Control
GPC	Generalized Predictive Control
PID	Proportional-Integral-Derivative
LQ	Linear-Quadratic
LQG	Linear-Quadratic-Gaussian
SSOP	Steady-state optimisation problem
RHDOP	Receding horizon dynamic optimisation problem
DSSO	Steady-state optimiser based on dynamic information
NLP-DISOPE	Nonlinear predictive DISOPE
LP-DISOPE	Linear predictive DISOPE
MTS	Modified two step method
CDC	Central difference formula
CPU	Central processing unit
min	Minimize
lim	Limit of

CHAPTER 1

INTRODUCTION

1.1 CONTROL SYSTEMS IN INDUSTRIAL PROCESSES

The purpose of control systems is to influence physical processes in such a way that certain control objectives may be achieved. Such control objectives reflect the goals of the industry which operates the process. Those goals are usually related to product quality, safety, economics, environmental regulations and operational constraints.

A physical process is a combination of operations carried out to change something in the physical world. Processes are characterized by their input and output elements in terms of matter, energy and information. An industrial process outputs products from raw materials and energy input. The input information to a process is a set of variables that influence or *control* the behaviour of the process. The output information of a process is a set of measured variables that characterize the operation of the process. Factors which cannot be manipulated but influence the process are called *disturbances* and they reflect the effects of the surrounding environment. The undesirable effects of external disturbances on the process must be suppressed. Control systems are used to achieve this.

Many processes are essentially unstable and must be equipped with control systems to ensure stable and, hence, safe and reliable operation.

Product quality and economic goals are of paramount importance. An industrial process should produce the desired amounts of the final products with certain minimum levels of quality. Also, the operation of the plant must correspond with the market conditions. Further, the operation should be as economical as possible in its use of energy, raw materials and labour.

Quality and economics can be defined by means of *optimisation*. The most common form of optimisation is *steady-state optimisation*, in which the optimum operating points are calculated and sent to the control systems as reference values (set-points). Control systems are used to *regulate* the operation of the process about

such optimum set-points. Often, the economic objectives which are pursued by optimisation yield optimum operational conditions which lie at the intersection of process constraints. Such constraints arise due to physical limitations inherent to process equipment and their operation, safety considerations and environmental regulations. Thus control systems should be able to take into account constraints in some way.

In the last few decades, there has been a rapid development in the field of digital computers and microelectronics. The application of digital computers in process control has evolved from the pioneering works in the 1950's to becoming a standard technique for implementation of new control systems in the 1980's and 90's. This involves from single loop controllers to large-scale distributed control systems. Digital computers are also being used increasingly as means for the design and analysis of control systems.

During the 1930's proportional-integral-derivative (PID) controllers first appeared. They were originally implemented using pneumatic technology and have passed through several development stages. Today, most new PID controllers are implemented digitally. They are the standard tool for solving process control problems. They are relatively easy to tune and no explicit model of the process model is required. In the presence of process nonlinearities a change in operating conditions may produce loss of performance of the PID controller. The presence of time-delays and multiple interacting control loops complicates the tuning process and may also limit controller performance. Today PID controllers which possess automatic tuning capabilities are becoming increasingly popular (Åström and Hägglund, 1988).

In the *model-based* approach for control, a process model is developed which can be used either as the basis for classical controller design methods or it can be incorporated directly in the control law. The latter is the starting point for many advanced control techniques, such as predictive control. The development of a model for a complicated process may be an expensive and time consuming task. However, the model based approach is becoming advantageous. Two reasons for this are: firstly, the high integration of modern processing plants makes plant operation more difficult; secondly, there are incentives for operating closer to limiting constraints to maximize profitability while taking into account safety criteria and environmental restrictions (Seborg *et al*, 1989).

1.2 SYSTEM IDENTIFICATION AND ADAPTIVE CONTROL

System identification is the field of mathematical modelling of systems from experimental data. It has acquired widespread applications in many areas. In control engineering system identification methods are used to obtain appropriate models for the design of control systems and simulation (Soderström and Stoica, 1989).

Ever since the beginning of systematic controller design there has been the problem of finding the proper controller structure and parameters for a given process. Another difficulty has been the fact that the controller must perform well for a range of operating points. Automatic adjustment of the controller parameters was first proposed in the 1940's. The term adaptive control was first used at that time (Isermann *et al*, 1992).

Given a process model structure, on-line system identification may be integrated with controller design. Thus, the parameters of the identified model and the performance requirements of the system are used in a controller design stage in order to obtain the controller parameters. The controller parameters may be updated on-line. This gives rise to a class of adaptive controller called self-tuning controller. In this class of adaptive controller the uncertainty in the estimated parameters of the process model is not accounted for in the controller design stage (Åström and Wittenmark, 1989).

1.3 OPTIMISATION OF INDUSTRIAL PROCESSES

The process industry has experienced important changes during the last few decades due to the increased costs of energy and raw materials, and increasingly strict environmental regulations. It is believed that emphasis should be on improving efficiency and increasing profitability of existing plants rather than on plant expansion (Edgar and Himmelblau, 1988). To achieve such a goal, one of the most important means is optimisation.

The desire to operate industrial processes optimally is not new. However, the capability to automate process optimisation is relatively recent. The theoretical basis for automating process optimisation has been available for about thirty years.

Moreover, the computing power required for implementing such systems has been made available at reasonable prices during the last decade (Balchen and Mumme, 1988).

Steady-state process optimisation, often named *optimising control,* can be situated within the functional hierarchy of the overall plant management and control system at a level called *supervisory control.*

The economic objective which is pursued by steady-state optimisation is usually the maximisation of net profit, given the product demand, prices of final products, raw materials, energy, equipment wearing, pollution taxes, operational costs, etc.

The basis for optimising control is the economic objective which is quantified by means of a performance index (or objective function), a (steady-state) mathematical model of the plant and knowledge of the relevant process constraints. The result from steady-state optimisation is a set of optimal controller set-points, at which the process should be regulated until a change in economic parameters determines a new optimum operating point.

Optimising control is particularly common in large integrated process systems (oil refineries, petrochemicals, etc.).

Dynamic optimisation is more complicated. It takes into account the dynamic behaviour of the process and intends to manipulate the input of the process during transient conditions in such a way that some dynamic criterion is optimized. Hence, a dynamic model of the plant is required. Dynamic optimisation is often termed *optimal control.*

Dynamic optimisation may be applied in the process industry, for instance, in the following cases which are directly related with economic objectives (Rijnsdorp, 1991):

* Optimisation of total process run, in cases when process behaviour deteriorates with time.
* When switching continuous processes from one mode of operation to another, trying to minimize the amount of off-specification product.
* Optimisation of the operation of batch processes, seeking to maximize, for example, product yield.

* When disturbances with economic impact have a frequency such that the process rarely reaches steady state.

However, dynamic optimisation with economic objectives is not widely applied in the process industry (Arkun and Stephanopoulos, 1980). Some reasons for this are (i) dynamic models are more difficult to develop than steady-state models; (ii) The solution of a dynamic optimisation problem is far more complicated and time consuming than steady state optimisation. The latter is becoming less important due to the increasing computer power becoming available at decreasing prices.

One particular class of dynamic optimisation, with regulatory rather than economic objectives, is a technique called LQG (linear-quadratic-gaussian). This is an established method for the design of multivariable control systems which has the advantages that the input-output pairing problem is avoided, for the structure of the controller is given by the design; stability is guaranteed if the model is perfect, and it may easily handle systems with time-delays and non-minimum phase behaviour (Anderson and Moore, 1989). Nevertheless, this technique has not become popular in the process industry because a good process model is required, and hence lacks robustness. Additionally, it may not handle constraints explicitly. For a survey of LQG applications in the process industries see the paper by Johnson (1993).

1.4 PREDICTIVE CONTROL

A particular technique which has been widely applied within the last two decades in the process industry is *Predictive Control* based on linear input-output models of the plant. Predictive control belongs to the class of *model-based* controller design concepts, because a model of the plant is used to compute the control action. Some well known predictive controllers are Dynamic Matrix Control, DMC (Cutler and Ramaker, 1979), and Model Algorithmic Control, MAC (Richalet *et al*, 1978). The reasons for their acceptance are many, but the main ones are: they are easy to tune; they may handle systematically process constraints, multivariable processes and time delays; knowledge of future set-point changes can be included; their computational requirements are modest (Soeterboek, 1992; Garcia *et al*, 1989).

Linear predictive controllers, such as DMC, have been extended to handle input and output constraints (Garcia and Morshedi, 1986) and some of them, such as Generalized Predictive Control (GPC) have adaptive features (Clarke *et al*, 1987).

Predictive controllers implement dynamic optimisation in a *receding-horizon* framework. If the objective is to follow a set-point, the process model is linear, the performance index is quadratic, and in the absence of process constraints, then predictive controllers allow a relatively simple analytical solution. Otherwise, iterative methods have to be used (Mayne and Michalska, 1990).

There have been growing interest in the last few years on extending predictive control concepts to take into account process nonlinearities. Some of those schemes have been presented in the literature (Sistu and Bequette, 1991; Gattu and Zafiriou, 1992; Balchen *et al*, 1992). A common characteristic is their increased computational load. Most of them are based on nonlinear state space models of the process and some of them include state and parameter estimation to give robustness, adaptability, and stability to the controller. If the dynamic optimisation scheme allows for general performance index specifications, economic criteria can be included (Balchen *et al*, 1992).

1.5 DYNAMIC OPTIMAL CONTROL

Optimal control theory is the mathematical tool used for addressing and solving dynamic optimisation problems.

In the seventeenth century Bernoulli studied the brachistocrone problem and then initiated the classical calculus of variations. After three centuries of developments, optimal control theory has been formalized as a general extension of the calculus of variations. It has developed into a very mature field and it has attracted the attention of numerous researchers from very diverse disciplines. Many successful applications of optimal control theory have been reported in the literature.

With the development of computers and the current emphasis of optimal design and operation of large-scale systems, optimal control theory has become increasingly important. The mathematical complexity of the optimal control approach and the sophistication of real world problems do not allow straightforward analytical

solutions to optimal control problems. Thus, algorithms implemented on digital computers have to be used.

The classical tool for solving optimal control problems is the calculus of variations. However, it cannot deal with control magnitude constraints. Two very important developments were Pontryagin's *minimum principle* (Pontryagin *et al*, 1962) and Bellman's *dynamic programming* (Bellman and Dreyfus, 1962). The minimum principle solves problems without control constraints in a similar way to the calculus of variations. However, the method is more general because it can work with control constraints. Dynamic programming can handle control and state constraints. Its main disadvantage is the so called "curse of dimensionality" which implies that the approach requires too much computer memory even with relatively low order problems.

Numerous algorithms have been proposed in the last two decades for solving optimal control problems. It is not the purpose to review here all of them. Nonetheless, it is worth mentioning a rather general classification according to the approach used for the solution:

* *Function space algorithms*: The necessary optimality conditions obtained from the application of the maximum principle are enforced iteratively in some way. Some examples here are quasilinearization methods, function space gradient algorithms, variation of extremals methods (Bryson and Ho, 1975; Sage & White, 1977; Kirk, 1970).
* *Parametrization methods*: Here control (and sometimes state) variables are parametrized in an approximated way (usually in terms of basis functions) and then the objective function is minimized (or maximized) by using finite dimensional mathematical programming (Sisirena and Tan, 1974; Sargent and Sullivan, 1978; Teo *et al*, 1991; Biegler, 1984).

Key algorithmic issues are convergence rates, memory requirements, computational load, handling of constraints, suboptimality, handling of large-scale systems and numerical precision.

Provided the dynamic model is linear, the performance index is quadratic (in terms of state and control variables) and constraints are absent, then the optimal control problem has a relatively straightforward non iterative solution usually based

on the solution of matrix differential (or difference) equations. This is a very important particular case of optimal control problems and is usually termed as LQ (linear-quadratic) optimal control (Anderson and Moore, 1989; Lewis, 1986a).

1.6 ISOPE ALGORITHMS

As has been mentioned in Section 1.2 the optimal set-points calculated by steady-state optimisation are based on a mathematical model of the controlled plant. Because, in general, the model will not be a faithful representation of the real physical process, the set-points so obtained will only be optimal for the model and not for the real plant. Moreover, the process operates in an environment which keeps changing. Hence it is important to provide the mathematical model of the process with some adaption.

In order to take into account differences between the mathematical model and the real process a technique called the *two-step method* has been proposed. Here the steady-state model contains parameters which are estimated by comparing model-based and measured outputs. Then the system optimisation and parameter estimation problems are treated separately and solved repeatedly until convergence is achieved. However, there is interaction between the optimisation and parameter estimation problems and the solution so obtained will be, in general, suboptimal. The reason for this lies in the fact that inadequate output derivative information (with respect to the manipulated variables) from the plant is used in the model-based optimisation.

In order to allow for the interaction between parameter estimation and system optimisation, the interacting variables are separated and, as a consequence, a modifier is introduced in the model-based optimisation. This modifier takes into account differences between the real process and model-based output derivatives with respect to the manipulated inputs. This enables the iterative technique to achieve the correct optimal operating point of the real process in spite of *model-reality* differences. This technique was originally introduced by Roberts (1979) and is called *integrated system optimisation and parameter estimation* (ISOPE).

ISOPE has been established for about fourteen years and a considerable number of ISOPE algorithms, centralized and hierarchical, have been developed (Roberts, Wan and Lin, 1992). Conditions for the convergence of the algorithms

have been rigorously investigated (Brdyś and Roberts, 1987). Several simulations and laboratory pilot-plant implementations have been carried out and the utility of the technique has been demonstrated. Furthermore, on-line implementation on large-scale process plants seems very likely in the near future (Lin and Griffiths, 1992; Griffiths *et al*, 1993).

Since the origin of the ISOPE technique its extension to dynamic optimisation has been suggested (Roberts, 1979). More recent works have also recommended such an extension (Amini-Largani, 1990). In recent research by Roberts (1992) a dynamic extension of ISOPE has been introduced. It has been termed DISOPE (Dynamic-ISOPE) and, as an extension, the philosophy behind the techniques remains very similar. However, DISOPE is in itself a new technique with a different range of applications. It can be considered as a distinctive field of research. Furthermore, as it has been mentioned in Section 1.2, dynamic optimisation is more complicated.

The development of novel optimal control algorithms, mainly new extensions of the DISOPE approach, and the study of their potential applications in process control have been the central areas of research of the doctoral work described in this thesis.

1.7 SCOPE AND AIMS OF THE THESIS

As it was mentioned in Section 1.5 ISOPE is a well established technique for optimising control of industrial processes. Dynamic ISOPE, on the other hand, is relatively recent.

As it was originally developed and published (Roberts, 1992), DISOPE addressed continuous-time, unconstrained, centralized and time invariant optimal control problems.

The central aim of this thesis is to advance and improve the existing knowledge on the technique so as to make it attractive on one hand for its implementation in the process industry and, on the other hand, as a novel nonlinear optimal control algorithm. Additionally, as a result of research work carried out during the project, a new technique for steady-state process optimisation based on dynamic information (DSSO) has been developed.

The means through which the central objective is to be achieved are:

* To develop discrete-time versions of DISOPE so as to make it suitable for digital computer on-line implementations.
* To extend the DISOPE technique to handle control and state dependent constraints.
* To develop hierarchical extensions of DISOPE so as to make it applicable in large-scale systems.
* To extend the DISOPE algorithm for taking into account reference trajectories which should be tracked.
* To investigate the ways the techniques developed can be applied in the process industry, particularly in the fields of nonlinear predictive control, batch process optimisation and predictive optimising control.
* To implement in software the algorithms developed and to test their performance through simulation studies.
* To investigate the ways the inherent flexibility of the DISOPE approach can be exploited.
* To compare the structure and performance of DISOPE with other existing techniques.

The scope and original contributions of this thesis are briefly summarized below.

Contribution to algorithm development

Several DISOPE control algorithms have been developed. This contribution includes theoretical derivation, development of implementable versions and actual software implementation of: discrete time, set-point tracking, hierarchical, control constrained, state constrained and receding horizon DISOPE algorithms. Emphasis has been given to make the technique suitable for industrial implementation. Furthermore, a steady-state optimiser based on dynamic information (DSSO) has been developed, implemented in software and tested with simulation examples.

Contribution to DISOPE approach flexibility exploitation

Some novel techniques which directly exploit the inherent flexibility of the DISOPE approach in terms of model-reality differences in both model dynamics and

performance indexes have been developed and investigated. This includes: (i) the use of saturation functions for handling control magnitude constraints by using (straightforward) unconstrained linear-quadratic model-based calculations (ii) The exact discretization scheme by which continuous time systems are dynamically optimized in an exact way while using model-based calculations in the (convenient) discrete domain. (iii) The use of penalty functions to take into account state or output magnitude constraints. (iv) The use of quadratic *incremental* control weighting in the dynamic performance index to provide zero off-set tracking *for constant set-points.*

Contribution to the prospective industrial application of the techniques developed

Several topics directly related with the application of DISOPE in the process industry have been addressed in this doctoral work. The most relevant are: (i) the study of the application of the technique in nonlinear predictive control, in which the suitability of DISOPE to be used as a dynamic optimiser in a receding horizon scheme is investigated. (ii) The application of DISOPE in batch process optimisation, where the real dynamic optimum of the batch process is achieved by integrating the algorithm with the batchwise operation of the plant. (iii) The application of DISOPE in receding horizon as a predictive optimising controller, based on adaptive linear models of the controlled plant. (iv) The application of DSSO for adaptive process optimisation. All the topics are supported by relevant simulations and in cases (iii) and (iv) realistic simulations of an industrial distillation column using a rigorous, high fidelity process simulator are presented.

Contribution to software implementation and algorithm testing

All the algorithms proposed have been implemented in software. The main tools used have been the C and C++ programming languages. The use of object oriented programming has allowed a natural way of handling matrices and their operations which, in this framework, has facilitated the programming and debugging stages while keeping a high speed of execution. Code reusability principles has been used. Some of the algorithms developed in this work have been implemented and tested on an industrial process simulator environment (OTISS^{TM}), bearing in mind prospective on-line implementations. Several simulations with different levels of complexity have been carried out using the software developed. These simulations

have allowed us to gain a greater understanding of the DISOPE technique, to test experimentally the algorithms developed and ideas proposed, and to compare the results obtained with those reported in published works by using other methods.

1.8 OUTLINE OF THE THESIS

An outline of the thesis is given below. A more detailed introduction is given at the beginning of each individual chapter.

Chapter 2: reviews the development and algorithmic details of DISOPE as originally published (Roberts, 1992), in its continuous time, unconstrained and centralized versions. Simulation studies are presented which illustrate the basic algorithm capabilities for solving highly nonlinear systems with model-reality differences. The effect of the tuning parameters (relaxation gains, convexification factors, etc.) is investigated.

Chapter 3: seeks to extend the basic DISOPE algorithm for handling control dependent constraints. An extension is developed using the minimum principle and the resulting algorithm remains basically unchanged, the main difference being the explicit handling of constraints in the model-based optimisation step. An alternative and more convenient way of handling control magnitude constraints by using a variable transformation based on a saturation function is introduced. Both approaches are implemented in software and tested with simulations.

Chapter 4: describes the development of a hierarchical two-level DISOPE algorithm. The approach used is based on the interaction-prediction approach (Jamshidi, 1983) and on the basic DISOPE technique. A new modifier is introduced to take into account the interactions between subsystems. The model-based problem for each subsystem is independent from the other subsystems, which shows that the algorithm is suitable for parallel or distributed processing. The algorithm is implemented in software and simulations are carried out.

Chapter 5: reports the development of a discrete-time DISOPE algorithm. An implementable version, using an LQ model-based problem, is also developed. The exact discretization scheme is introduced. This allows the application of discrete-time DISOPE to continuous time systems by integrating the nonlinear model dynamics between sampling times, so avoiding the use of approximate discretization techniques, such as Euler's method. A software implementation is developed and the algorithm's performance is studied by means of simulations.

Chapter 6: describes the use of discrete-time DISOPE for developing a new technique for solving the optimal set-point tracking problem for nonlinear systems. This is achieved by solving a sequence of LQ tracking problems with converge to the correct optimal solution. The algorithm is implemented in software and numerical simulations are carried out. The use of quadratic incremental control weighting, which provides zero off-set tracking for constant inputs, is introduced exploiting the algorithm's flexibility, while the model-based calculations use quadratic absolute control weighting. Its suitability to be used in a nonlinear predictive control scheme is emphasized.

Chapter 7: Extends the DISOPE technique for handling optimal control problems with state dependent inequality constraints. The approach used is the penalty relaxation technique. A state constrained simulation example is presented.

Chapter 8: reports the application of DISOPE in batch process optimisation. This is achieved by integrating the algorithm's iterations with the batchwise operation of the plant, in such a way that the correct dynamic optimum of the plant is achieved in a sequential manner in spite of model-plant mismatch. The problem of measuring the time-varying jacobian matrices is addressed by using the shadow model concept (Griffiths, 1993). Comprehensive simulation studies are provided.

Chapter 9: describes the application of set-point tracking DISOPE in nonlinear predictive control. The receding horizon dynamic optimisation is carried out at every sampling interval by using DISOPE. State and uncertain parameters are estimated from the possibly noisy output measurements by using an Extended Kalman Filter

(Lewis, 1986b). The controller is implemented in software and comprehensive simulation studies are provided.

Chapter 10: describes the development of two optimising controllers which are able to drive a process from a suboptimal operational condition to its steady-state optimum. The controllers use derivative and state information from the plant via a shadow model and an adaptive state-space linear model identifier. The new algorithms are developed from the basis that a nonlinear model of the process is not available for prediction purposes. One approach, known as DSSO, does not requires predictions. The second algorithm, known as LP-DISOPE, uses predictions based on an adaptive linear model of the process. The steady-state optimality of the procedures is analyzed. Both techniques are tested with simulation examples, including realistic industrial-scale simulations of the optimisation of a multicomponent distillation column.

Chapter 11: compares the DISOPE technique with a well established nonlinear optimal control technique such as quasilinearization. Furthermore, comparisons with previous work by Hassan and Singh (1976) and Mahmoud *et al* (1980) are discussed.

Chapter 12: draws conclusions from the results obtained in this thesis and presents a series of suggestions for further work in this area.

1.9 SUMMARY

The development and applications of novel optimal control algorithms is the central subject of the research work described in this thesis. The main objective is to advance and improve the existing knowledge of a dynamic optimisation technique called DISOPE, so as to make it attractive on one hand for its implementation in the process industry and, on the other hand, as a novel nonlinear optimal control algorithm.

In this introductory chapter, in order to address some motivational issues, to review the background to the research area, and to establish the framework of the thesis, some important topics have been discussed. A short discussion on the role of

control systems in industrial processes has been related to an overview on the use of optimisation in the process industry. Further, some historical background on system identification and adaptive control, model-based predictive control and on dynamic optimal control have also been presented. Moreover, a brief review on the ISOPE technique, which is the steady-state predecessor of DISOPE, has been given. This has been followed by a discussion on the aims, scope and original contribution of this doctoral work. Finally, the contents and structure of the thesis have been described.

CHAPTER 2

CONTINUOUS TIME DISOPE ALGORITHM

In this chapter, we shall deal with the original DISOPE algorithm as was introduced by Roberts (1992, 1993a). Initially, a brief introduction on the theory of optimal control is given. The formulation addresses the continuous time, unconstrained and centralized optimal control problem, with fixed terminal time and terminal state value constraints. The basic mathematical tool is the calculus of variations. Simulation examples are given illustrating the basic properties of the algorithm.

2.1 THE OPTIMAL CONTROL PROBLEM

Before introducing the DISOPE theory, it is convenient to present some background on the basic optimal control problem. To solve an optimal control problem we must first define a goal or performance index for the process we intend to optimize. The performance index is selected to make the plant exhibit a desired kind of behaviour. This requires an appropriate definition of the problem from the physical point of view and a translation into convenient mathematical terms. To be able to apply optimal control to a process in an effective way we must estimate the current state of the process from the (very often incomplete and noisy) output measurements. This is called *state estimation*. Further, we must obtain a mathematical model with the appropriate structure and parameters so that it describes properly the dynamics of the process. This is called *system identification*. The *optimal control problem* consists in finding the best control inputs (manipulated variables) so as to minimize (or maximize) the performance index, given knowledge of the system state, and the mathematical model of the process (Sage and White, 1977; Lewis, 1986a).

2.1.1 Problem formulation

Suppose that the plant is described by the nonlinear time-varying differential equation

$$\dot{x} = f^*(\, x(t),\ u(t),\ t) \tag{2,1}$$

where $f^* : \Re^n \times \Re^m \times \Re \rightarrow \Re^n$ represents a set of state equations which describe the process with state $x(t) \in \Re^n$ and control input $u(t) \in \Re^m$. Further assume that the following performance index has been chosen:

$$J^* = \varphi(x(t_f)) + \int_{t_0}^{t_f} L^*(x(t), u(t), t)\ dt \tag{2,2}$$

where $[t_0, t_f]$ is the fixed time interval of interest, $\varphi : \Re^n \rightarrow \Re$ is a scalar valued terminal weighting function and $L^* : \Re^n \times \Re^m \times \Re \rightarrow \Re$ is a continuous performance function.

If the state of the system at the initial time t_0 is assumed known (being measured or estimated), with value $x(t_0) = x_0$, and if no constraints on the values of control and state variables are taken into consideration, apart from the dynamic constraint (2,1), the optimal control problem can be formulated as follows:

$$\min_{u(t)} \quad J^* = \varphi(x(t_f)) + \int_{t_0}^{t_f} L^*(x(t), u(t), t)\ dt$$

subject to

$$\dot{x} = f^*(x(t),\ u(t),\ t)$$
$$x(t_0) = x_0$$

2.1.2 Necessary optimality conditions

The problem so formulated is by no means general but will suffice our introductory purposes. The problem as stated can be treated by using the classical *calculus of variations*. For convenience, a scalar function H^* (the *Hamiltonian*) is defined as follows:

$$H^*(x(t),u(t),p(t),t) = L^*(x(t),u(t),t) + p(t)^{\mathrm{T}} f^*(x(t),u(t),t) \qquad \textbf{(2,3)}$$

where $p(t)$ is a multiplier function usually termed as the *costate*. By using calculus of variations and relatively straightforward algebraic manipulations (see, for example, (Lewis, 1986a: 150-153) for the derivation), the following well known necessary optimality conditions are obtained.

Stationarity:

$$\nabla_u H^* = 0 \qquad \textbf{(2,4)}$$

Costate equation:

$$\nabla_x H^* + \dot{p}(t) = 0 \qquad \textbf{(2,5)}$$

State equation:

$$\nabla_p H^* - \dot{x}(t) = 0 \qquad \textbf{(2,6)}$$

Boundary conditions:

$$\begin{aligned} x(t_0) &= x_0 \\ p(t_f) &= \nabla_x \varphi(x(t))\big|_{t=t_f} \end{aligned} \qquad \textbf{(2,7)}$$

Thus, in order to obtain a control function $u(t)$ to minimize the value of the performance index J^* one must solve the differential equations (2,5) and (2,6), with $u(t)$ given by the algebraic equation (2,4). The boundary conditions for the coupled differential equations to be solved are split, because $x(t_0)$ is given and $p(t_f)$ can be computed from (2,7). This is known as a two-point boundary value problem (TPBVP). In general, it is difficult to solve these problems.

2.1.3 Linear-Quadratic Optimal Control

An important family of unconstrained optimal control problems is that of the linear-quadratic type. The name of the linear-quadratic (often termed as LQ) problems arises because the system dynamics are represented by linear differential equations while the performance index is quadratic in terms of state and control variables. The linear quadratic optimal control theory will be very important for the developments presented in this thesis and therefore a brief introduction to the topic is relevant. There are two types of LQ problems: regulator and tracking problems. The linear-quadratic regulator problems are usually formulated as follows:

$$\min_{u(t)} \quad J_l = \frac{1}{2}x(t_f)^{\top}\Phi x(t_f) + \int_{t_0}^{t_f}\frac{1}{2}[\, x(t)^{\top}Qx(t)+u(t)^{\top}Ru(t)\,]\, dt$$

subject to

$$\dot{x} = Ax(t) + Bu(t)$$
$$x(t_0) = x_0$$

where $\Phi \geq 0$, $Q \geq 0$ and $R > 0$ are symmetric weighting matrices of the appropriate dimensions, A is the system dynamic matrix and B is the control distribution matrix. The corresponding Hamiltonian function is:

$$H_l = \frac{1}{2}(\, x(t)^{\top}Qx(t)+u(t)^{\top}Ru(t)\,) + p(t)^{\top}(\, Ax(t)+Bu(t)\,) \qquad \textbf{(2,8)}$$

In this case, the two point boundary value problem is relatively easy to solve. A popular method of solution, given its computational efficiency, is known as the

backward sweep method (Bryson and Ho, 1975; Lewis, 1986a). Here a linear relationship between the costate and state variables of the form $p(t) = K(t)x(t)$, where $K(t)$ is a time-varying *nxn* matrix, is assumed. Application of the necessary optimality conditions and boundary conditions outlined in Section 2.1.2 gives rise to the following noniterative solution procedure:

Procedure 2.1.3: Simple LQ regulator solution

Step a: Solve backwards from t_f to t_0 the following Ricatti differential equation, with terminal condition $K(t_f) = \Phi$:

$$\dot{K}(t) = K(t)BR^{-1}B^{\top}K(t) - A^{\top}K(t) - K(t)^{\top}A + Q$$

Step b: Compute the state $x(t)$, $t \in [t_0, t_f]$ by integrating from the initial condition $x(t_0) = x_0$ the following equation:

$$\dot{x} = (A - BG(t))x(t)$$

Step c: Compute the optimal control $u(t)$, $t \in [t_0, t_f]$ from the state feedback control law:

$$u(t) = -G(t)x(t)$$

where the Kalman gain is given by $G(t) = R^{-1}B^{\top}K(t)$.

Some advantages of using LQ optimal control are (Anderson and Moore, 1989):

* Nearly all LQ optimal control problems have solutions achievable with relatively little computing effort, as opposed to some nonlinear optimal control problems.
* LQ optimal control results can be applied to nonlinear systems operating in a small signal basis.
* Under certain conditions, LQ optimal controllers possess a number of attractive properties regarding the stability of the system and the robustness of the controller.
* The relatively simple computational procedures used in LQ optimal control can sometimes be carried over to nonlinear optimal control problems.

2.2 DYNAMIC ISOPE

A key issue of the ISOPE techniques is that the computations based on an approximated model of the process converge to the real optimum of the plant, in spite of deficiencies in the mathematical model. In the DISOPE approach this particular aspect continues to be very important as will be shown in this thesis. The distinction between *reality* and *model* plays a crucial role in the DISOPE framework. The definition of reality will have basically two interpretations in this thesis. On the one hand reality may be taken as the actual plant dynamics which are normally unknown and uncertain. On the other hand, reality may be interpreted as a known but difficult to tackle dynamic description of the plant. The differentiation between model and reality extends also to the performance index. There will be a distinction between the performance index associated to the real dynamics and the performance index which corresponds to the dynamic model. Thus we can refer to *real* and *model-based* performance indexes. In any case, the model represents an (sometimes convenient and intentional) approximation of reality, whatever reality means. In the particular applications presented in this thesis this duality should not cause any confusions. The particular interpretation of the term reality will be implicit or clearly stated.

2.2.1 Problem formulation and solution

Recall now the problem formulated in Section 2.1.1, assume that the superscript * denotes reality and let us call it Real Optimal Control Problem (ROP2). Consider now the following, possibly simplified Model-Based Optimal Control Problem (MOP2):

MOP2

$$\min_{u(t)} \quad J_m = \varphi(x(t_f)) + \int_{t_0}^{t_f} L(x(t),\ u(t),\ \gamma(t))\ dt$$

subject to

$$\dot{x} = f(x(t),\ u(t),\ \alpha(t))$$
$$x(t_0) = x_0$$

where state and control vectors have the same dimensions as in ROP2, J_m is a model-based performance index, $L : \Re^n \times \Re^m \times \Re \rightarrow \Re$ is a continuous weighting function and perhaps a simplification of a known L^*, $f : \Re^n \times \Re^m \times \Re^r \rightarrow \Re^n$ is an approximated dynamic model of f^*, $\gamma(t) \in \Re$ and $\alpha(t) \in \Re^r$ are continuous parameters. The role of $\gamma(t)$ and $\alpha(t)$ will be to take into account model reality-differences *in value*. Notice that as in the original formulation of the DISOPE technique, the terminal weighting function φ in MOP2 is identical to that in ROP2.

Using a two-step method, the solution of MOP2 (optimisation step) provides the control $u(t)$ as a function of the current parameter estimates $\alpha(u(t))$ and $\gamma(u(t))$. In turn such estimates may be obtained by matching model and real state equations and continuous weighting functions (parameter estimation step) at the current computed control $u(t)=u(\alpha(t),\gamma(t))$. It is easy to notice that optimisation and parameter estimation interact and, in general, because the model is only an approximation to reality, several iterations may be required before convergence is achieved. However, such iterations do not lead, in general, to the correct optimal

solution of ROP2 (Durbeck, 1965; Foord, 1974), and it is necessary to integrate optimisation with parameter estimation taking into account their mutual interaction.

The key to integrating system optimisation and parameter estimation is to define an Expanded Optimal Control Problem (EOP2) which, in spite of being model-based, is made equivalent to ROP2 by adding appropriate equality constraints on state equations and continuous weighting function values. Furthermore, state and control variables are separated between parameter estimation and optimisation steps by introducing the new state and control variables $z(t)$ and $v(t)$, respectively.

EOP2

$$\min_{u(t)} \quad J_e = \varphi(x(t_f)) + \int_{t_0}^{t_f} L(x(t),\ u(t),\ \gamma(t))\ dt$$

subject to

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

$$x(t_0) = x_0$$

$$f(z(t),v(t),\alpha(t)) = f^*(z(t),v(t),t)$$

$$L(z(t),v(t),\gamma(t)) = L^*(z(t),v(t),t)$$

$$u(t) = v(t)$$

$$x(t) = z(t)$$

Adjoining all the equality constraints to the performance index by using Lagrange multipliers, we obtain the following augmented performance index J_e':

$$J_e' = \varphi(x(t_f)) + \int_{t_0}^{t_f} [\ L(x,u,\gamma) + p^{\top}(f(x,u,\alpha)-\dot{x}) + \lambda^{\top}(v-u) \qquad \textbf{(2,9)}$$

$$+ \beta^{\top}(z-x)+\mu^{\top}(f^*(z,v,t)-f(z,v,\alpha)) + \xi(L^*(z,v,t)-L(z,v,\gamma))\ ]\ dt$$

where the time index has been dropped for convenience and $p(t)\in\Re^n$, $\lambda(t)\in\Re^m$, $\beta(t)\in\Re^n$, $\mu(t)\in\Re^m$ and $\xi(t)\in\Re$ are multiplier functions.

If we define the *augmented Hamiltonian function* as:

$$H = L(x,u,\gamma) + p^{\top} f(x,u,\alpha) - \lambda^{\top} u - \beta^{\top} x \qquad \textbf{(2,10)}$$

then we can rewrite (2,9) as:

$$J_e' = [\varphi(x)]_{t=t_f} + \int_{t_0}^{t_f} [\, H - p^{\top}\dot{x} + \lambda^{\top} v + \beta^{\top} z + \mu^{\top}(f^*(z,v,t) - f(z,v,\alpha)) + \xi(L^*(z,v,t) - L(z,v,\gamma)) \,]\, dt \qquad \textbf{(2,11)}$$

By applying calculus of variations to (2,11) (see for example (Kirk, 1970) for an introduction to the subject), and taking into account that t_0, t_f and x_0 are fixed, we obtain the following expression for the first variation of J_e':

$$\delta J_e' = \nabla_x \varphi^{\top} \delta x\big|_{t=t_f} + \int_{t_0}^{t_f} \{ \nabla_u H^{\top} \delta u + \nabla_x H^{\top} \delta x - p^{\top} \delta \dot{x} + [\nabla_p H - \dot{x}]^{\top} \delta p + [\lambda + (f_v^* - f_v)^{\top} \mu + \xi(\nabla_v L^* - \nabla_v L)]^{\top} \delta v + [\beta + (f_z^* - f_z)^{\top} \mu + \xi(\nabla_z L^* - \nabla_z L)]^{\top} \delta z + [\nabla_\alpha H - f_\alpha^{\top} \mu]^{\top} \delta \alpha + [\nabla_\gamma H - \xi \nabla_\gamma L]^{\top} \delta \gamma \}\, dt \qquad \textbf{(2,12)}$$

Integrating by parts to eliminate the variation in $\dot{x}$ we obtain:

$$\delta J_e' = [\nabla_x \varphi - p]^{\top} \delta x\big|_{t=t_f} + p^{\top} \delta x\big|_{t=t_0} + \int_{t_0}^{t_f} \{ \nabla_u H^{\top} \delta u + [\nabla_x H + \dot{p}]^{\top} \delta x + [\nabla_p H - \dot{x}]^{\top} \delta p + [\lambda + (f_v^* - f_v)^{\top} \mu + \xi(\nabla_v L^* - \nabla_v L)]^{\top} \delta v + [\beta + (f_z^* - f_z)^{\top} \mu + \xi(\nabla_z L^* - \nabla_z L)]^{\top} \delta z + [\nabla_\alpha H - f_\alpha^{\top} \mu]^{\top} \delta \alpha + [\nabla_\gamma H - \xi \nabla_\gamma L]^{\top} \delta \gamma \}\, dt \qquad \textbf{(2,13)}$$

According to the Lagrange multiplier theory the constrained minimum of J_e is achieved at the unconstrained minimum of J_e', which is achieved when $\delta J_e' = 0$. Setting to zero the coefficients of the independent increments in (2,12), and concluding by inspection that $\xi = 1$ and $\mu = p$ the following necessary optimality conditions are obtained:

Stationarity:

$$\nabla_u H = 0 \tag{2,14}$$

Costate equation:

$$\nabla_x H + \dot{p}(t) = 0 \tag{2,15}$$

State equation:

$$\nabla_p H^* - \dot{x}(t) = 0 \tag{2,16}$$

Boundary conditions:

$$\begin{aligned} x(t_0) &= x_0 \\ p(t_f) &= \nabla_x \varphi(x(t))\big|_{t=t_f} \end{aligned} \tag{2,17}$$

Multiplier equations:

$$\lambda(t) = [\frac{\partial f}{\partial v} - \frac{\partial f^*}{\partial v}]^T \hat{p}(t) + [\nabla_v L - \nabla_v L^*] \tag{2,18}$$

$$\beta(t) = [\frac{\partial f}{\partial z} - \frac{\partial f^*}{\partial z}]^T \hat{p}(t) + [\nabla_z L - \nabla_z L^*]$$

plus the following equality constraints stated in the formulation of EOP2:

$$\begin{aligned} f(z(t), v(t), \alpha(t)) &= f^*(z(t), v(t), t) \\ L(z(t), v(t), \gamma(t)) &= L^*(z(t), v(t), t) \end{aligned} \tag{2,19}$$

$$\begin{aligned} v(t) &= u(t) \\ z(t) &= x(t) \\ \hat{p}(t) &= p(t) \end{aligned} \tag{2,20}$$

where $\hat{p}(t)$ has been introduced as a costate separation variable. We assume that the structure of f and L is such that given $v(t)$ and $z(t)$ $t \in [t_0, t_f]$ the values of $\alpha(t)$ and $\gamma(t)$ $t \in [t_0, t_f]$ can be uniquely determined from (2,19). Notice that optimality conditions (2,14), (2,15) and (2,16) are model-based, and that $\lambda(t)$ and $\beta(t)$ $t \in [t_0, t_f]$

carry information on model-reality differences, *in curvature*, as opposed to $\alpha(t)$ and $\gamma(t)$ $t \in [t_0, t_f]$ which carry information on model-reality differences *in value*.

Recall our goal which is to solve ROP2. We have defined EOP2 which is equivalent to ROP2 and we have derived its necessary optimality conditions by using variational calculus. Thus if we satisfy the optimality conditions of EOP2, given the equivalence, we are also satisfying the optimality conditions of ROP2. We intend to solve ROP2 by using model-based calculations. Given values of $\alpha(t)$, $\gamma(t)$, $\lambda(t)$ and $\beta(t)$, it is easy to see that the solution of the following problem satisfies the definition of the augmented Hamiltonian (2,10) and also the model-based optimality conditions (2,14), (2,15) and (2,16), and border conditions (2,17). It is called Modified Model Based Problem (MMOP2) and is defined as follows:

MMOP2

$$\min_{u(t)} \quad J_M = \varphi(x(t_f)) + \int_{t_0}^{t_f} [\, L(x(t),\, u(t),\, \gamma(t)) - \lambda(t)^T u(t) - \beta(t)^T x(t) \,]\, dt$$

subject to

$$\dot{x} = f(x(t),\, u(t),\, \alpha(t))$$
$$x(t_0) = x_0$$

Now, if based on given values of $v(t)$, $z(t)$ and $\hat{p}(t)$ we compute the functions $\alpha(t)$, $\gamma(t)$, $\lambda(t)$, and $\beta(t)$ from (2,18) and (2,19), and if the solution $u(t)$, $x(t)$ and $p(t)$ of MMOP2 obtained from those functions additionally satisfy (2,20), then that solution is also the solution of ROP2.

This reasoning gives rise to the following DISOPE algorithm which, assuming convergence, achieves the solution of ROP2 via repeated solutions of MMOP2 (Roberts, 1992).

Algorithm 2.2.1: Continuous time DISOPE algorithm

Data — f, L, φ, x_0, t_0, t_f, and means for calculating f^* and L^*.

Step 0 — Compute or choose a nominal solution $u^0(t)$, $x^0(t)$ and $p^0(t)$. Set $i=0$, $v^0(t)=u^0(t)$, $z^0(t)=x^0(t)$, $\hat{p}^0(t)=p^0(t)$, $t \in [t_0,t_f]$.

Step 1 — Compute the parameters $\alpha^i(t)$, $\gamma^i(t)$ to satisfy (2,19). This is called the *parameter estimation step.*

Step 2 — Compute the multipliers $\lambda^i(t)$ and $\beta^i(t)$ from (2,18).

Step 3 — With specified $\alpha(t)$, $\gamma(t)$, $\lambda(t)$ and $\beta(t)$ solve the modified model-based optimal control problem MMOP2 to obtain $u^{i+1}(t)$, $x^{i+1}(t)$ and $p^{i+1}(t)$. This is called the *system optimisation step.*

Step 4 — This step tests convergence and updates the estimate for the optimal solution of ROP2. In order to provide a mechanism for regulating convergence, a simple relaxation method is employed to satisfy (2,20). This is:

$$\begin{aligned} v^{i+1}(t) &= v^i(t) + k_v(u^{i+1}(t) - v^i(t)) \\ z^{i+1}(t) &= z^i(t) + k_z(x^{i+1}(t) - z^i(t)) \\ \hat{p}^{i+1}(t) &= \hat{p}^i(t) + k_p(p^{i+1}(t) - \hat{p}^i(t)) \end{aligned} \qquad \textbf{(2,21)}$$

where k_v, k_z and $k_p \in (0,1]$ are scalar gains. If $v^{i+1}(t) = v^i(t)$ within a given tolerance stop, else set $i=i+1$ and continue from step 1.

2.2.2 Performance index augmentation

Variable augmentation has been used in steady-state ISOPE algorithms in order to make the technique insensitive to the choice of relaxation gains, with the additional effect that it also improves the convergence behaviour of the algorithms.

This is because variable augmentation convexifies the optimisation problem (Rockafellar, 1970, 1974; Brdys *et al*, 1987).

Augmentation was introduced by Roberts (1993a) in the DISOPE technique, resulting in improved algorithm stability and convergence behaviour in difficult cases. Similar techniques have been applied in optimal control algorithms by Hassan and Singh (1976) and Sakawa and Shindo (1980).

Variable augmentation is applied in DISOPE by adding convexification terms to the performance index of EOP2. This is, J_e becomes:

$$J_e = \varphi(x(t_f)) + \int_{t_0}^{t_f} [\, L(x(t),\, u(t),\, \gamma(t)) + \tfrac{1}{2} r_1 \|u(t)-v(t)\|^2 + \tfrac{1}{2} r_2 \|x(t)-z(t)\|^2 \,]\, dt \qquad \textbf{(2,22)}$$

where r_1 and r_2 are given scalar convexification factors.

The definition of the augmented Hamiltonian is changed to take into account the new terms:

$$H = L(x,u,\gamma) + p^{\top} f(x,u,\alpha) - \lambda^{\top} u - \beta^{\top} x + \tfrac{1}{2} r_1 \|u-v\|^2 + \tfrac{1}{2} r_2 \|x-z\|^2 \qquad \textbf{(2,23)}$$

By using variational calculus it is possible to find that the model-based optimality conditions obtained in Section 2.2.1 continue to be valid. Furthermore, the main change in Algorithm 2.2.1 is that the solution of MMOP2 requires information on $v(t)$ and $z(t)$, for the performance index in MMOP2 becomes:

$$J_M = \varphi(x(t_f)) + \int_{t_0}^{t_f} [\, L(x(t),\, u(t),\, \gamma(t)) - \lambda(t)^{\top} u(t) - \beta(t)^{\top} x(t) + \tfrac{1}{2} r_1 \|u(t)-v(t)\|^2 + \tfrac{1}{2} r_2 \|x(t)-z(t)\|^2 \,]\, dt \qquad \textbf{(2,24)}$$

Notice that, at the end of the iterations, $u(t)=v(t)$ and $x(t)=z(t)$ so that at this stage the augmentation terms and their derivatives are zero, so having no effect in the real optimality of the solution.

2.2.3 Terminal state constraints

In some problems we are interested in restricting functions of the terminal state to have prescribed values, that is:

$$\psi(x(t_f), t_f) = 0 \tag{2,25}$$

for a given function $\psi : \Re^n \times \Re \rightarrow \Re^q$. Equation (2,25) is an additional equality constraint to ROP2, MOP2, EOP2 and MMOP2. It can be treated by using the Lagrange multiplier theory (see, for example, Bryson and Ho, 1975). As a consequence, a new boundary condition for the costate is obtained as follows:

$$p(t_f) = \nabla_x \varphi + \frac{\partial \psi^\tau}{\partial x} \nu \Big|_{x=x(t_f)} \tag{2,26}$$

where $\nu \in \Re^q$ is a Lagrange multiplier vector to be found so that the additional necessary condition (2,25) is satisfied. Notice that in terminal state constrained problems some reachability conditions must be satisfied for a solution to exist (see, for example, Lewis (1986a) for a discussion on the LQ case).

2.3 DISOPE WITH LINEAR-QUADRATIC MODEL-BASED PROBLEM

2.3.1 Formulation

If the model-based problem (MOP2) is chosen as a linear-quadratic approximation of ROP2, then noniterative methods similar to Procedure 2.1.3 can be used for the model-based computations.

Thus for computational advantage, a linear model-based dynamic function f and quadratic weighting functions L and φ may be chosen. Considering that $\alpha(t)$ and $\gamma(t)$ enter as shift parameters we have:

$$\begin{aligned} L(x,u,\gamma) &= \tfrac{1}{2} x(t)^\tau Q x(t) + \tfrac{1}{2} u(t)^\tau R u(t) + \gamma(t) \\ \varphi(x(t_f)) &= \tfrac{1}{2} x(t_f)^\tau \Phi x(t_f) \\ f(x,u,\alpha) &= A x(t) + B u(t) + \alpha(t) \end{aligned} \tag{2,27}$$

where $\Phi \geq 0$, $Q \geq 0$ and $R > 0$ are symmetric weighting matrices of the appropriate dimensions, A and B are matrices which represent a linear model of f^*.

Notice that more general terminal weighting specifications may be introduced by exploiting the fact that a performance index of the form

$$J_e = \varphi(x(t_f)) + \int_{t_0}^{t_f} L^*(x(t),u(t),t)$$

can be represented by making the following substitution (Sakawa and Shindo, 1980):

$$L^*(x,u,t) \leftarrow L^*(x,u,t) + \nabla_x \varphi^\top f^*(x,u,t)$$

By using (2,27), and including the variable augmentation discussed in Section 2.2.2 the linear-quadratic MMOP2 can be written as:

$$\min_{u(t)} \; J_M = \tfrac{1}{2}x(t_f)^\top \Phi x(t_f) + \int_{t_0}^{t_f} [\; \tfrac{1}{2}x(t)^\top Q x(t) + \tfrac{1}{2}u(t)^\top R u(t) + \gamma(t)$$
$$-\lambda^\top(t)u(t) - \beta^\top(t)x(t) + \tfrac{1}{2}r_1\|u(t)-v(t)\|^2 + \tfrac{1}{2}r_2\|x(t)-z(t)\|^2]\; dt$$

subject to

$$\dot{x} = Ax(t) + Bu(t) + \alpha(t)$$
$$x(t_0) = x_0$$

The corresponding Hamiltonian function is:

$$H = \tfrac{1}{2}(x(t)^\top Q x(t) + u(t)^\top R u(t)) + \gamma(t) + p(t)^\top (Ax(t) + Bu(t) + \alpha(t)) \qquad \textbf{(2,28)}$$
$$-\lambda(t)^\top u(t) - \beta(t)^\top x(t) + \tfrac{1}{2}r_1\|u(t)-v(t)\|^2 + \tfrac{1}{2}r_2\|x(t)-z(t)\|^2$$

Applying the model-based optimality conditions (2,14), (2,15), (2,16) and (2,17) with H given by (2,28), the following TPBVP is obtained:

$$\dot{x} = Ax(t) - B\bar{R}^{-1}(B^\top p(t) - \bar{\lambda}(t)) + \alpha(t) \qquad \textbf{(2,29)}$$
$$\dot{p} = -\bar{Q}x(t) - A^\top p(t) + \bar{\beta}(t)$$

with border conditions:

$$x(t_0) = x_0$$
$$p(t_f) = \Phi x(t_f) \qquad \textbf{(2,30)}$$

where $\bar{R} = R + r_1 I_m$ and $\bar{Q} = Q + r_2 I_n$ are augmented weighting matrices and $\bar{\lambda}(t) = \lambda(t) + r_1 v(t)$, and $\bar{\beta}(t) = \beta(t) + r_2 z(t)$ are augmented multipliers.

This linear TPBVP can be solved by using the backward sweep method (Bryson and Ho, 1975; Lewis, 1986a). The key is to assume the relationship between costate and state as $p(t) = K(t)x(t) + k(t)$, where $K(t)$ is a nxn matrix and $k(t) \in \Re^n$. This gives rise to the following noniterative solution procedure (See Appendix A for the derivation):

Procedure 2.3.1: Solution of linear-quadratic MMOP

Step a: Solve backwards from t_f to t_0 the following differential equations, with terminal conditions $K(t_f) = \Phi$ and $k(t_f) = 0$:

$$\dot{K} = K(t)B\bar{R}^{-1}B^T K(t) - A^T K(t) - K(t)A - \bar{Q}$$
$$\dot{k} = K(t)B\bar{R}^{-1}B^T k(t) - A^T k(t) - K(t)B\bar{R}^{-1}\bar{\lambda}(t) - K(t)\alpha(t) + \bar{\beta}(t)$$

Step b: Compute the time-varying Kalman gain $G(t)$, $t \in [t_0, t_f]$ and driving input $g(t) \in \Re^m$, $t \in [t_0, t_f]$ from:

$$G(t) = \bar{R}^{-1}B^T K(t)$$
$$g(t) = -\bar{R}^{-1}(B^T k(t) - \bar{\lambda}(t))$$

Step c: Compute the state $x(t)$, $t \in [t_0, t_f]$ by integrating from the initial condition $x(t_0) = x_0$ the following differential equation:

$$\dot{x} = (A - BG(t))x(t) + Bg(t) + \alpha(t)$$

Step d: Compute the costate $p(t)$, $t \in [t_0, t_f]$ from:

$$p(t) = K(t)x(t) + k(t)$$

Step e: Compute the control input $u(t)$, $t\in[t_0,t_f]$ from the control law:

$$u(t) = -G(t)x(t)+g(t)$$

The linear-quadratic formulation enables the augmented multipliers $\bar{\lambda}(t)$ and $\bar{\beta}(t)$, $t\in[t_0,t_f]$ to be expressed as (see equation (2,18)):

$$\begin{aligned}\bar{\lambda}(t) &= [\frac{\partial f}{\partial v}-\frac{\partial f^*}{\partial v}]^T\hat{p}(t) + [\bar{R}v-\nabla_v L^*]\\ \bar{\beta}(t) &= [\frac{\partial f}{\partial z}-\frac{\partial f^*}{\partial z}]^T\hat{p}(t) + [\bar{Q}z-\nabla_z L^*]\end{aligned} \qquad \textbf{(2,31)}$$

while the calculation of parameter $\alpha(t)$, $t\in[t_0,t_f]$ becomes (see equation (2,19)), noting that it is not necessary to calculate $\gamma(t)$:

$$\alpha(t) = f^*(z(t),v(t),t) - Az(t) - Bv(t) \qquad \textbf{(2,32)}$$

2.3.2 Terminal state constraints

Terminal state constraints of the type $x_i(t_f)=0$, $i\in[1,q]$ will be taken into consideration. This kind of constraint can be written as:

$$\psi(x(t_f)) = Cx(t_f) = 0 \qquad \textbf{(2,33)}$$

where $C = [I_q|0]$ is a qxn matrix. Notice that a terminal constraint of the type $x_i(t_f)=x_{if}$, $i\in[1,q]$ can be achieved by a straightforward shift change of state variable. The resulting TPBVP is identical to (2,29), but with boundary conditions (see equation (2,26)):

$$x(t_0) = x_0$$
$$p(t_f) = \Phi x(t_f) + C^{\top}\nu \qquad \textbf{(2,34)}$$

where $\nu \in \Re^q$ is a Lagrange multiplier to be determined such that (2,33) is satisfied. The solution of MMOP2 taking into account the terminal state constraint is again based on the backward sweep method (Bryson and Ho, 1975). The key is to assume the relationship between costate and state as $p(t) = K(t)x(t) + k(t) + F(t)\nu$, and express the (fixed) terminal state function as $\psi = F(t)^{\top}x(t) + W(t)\nu + \eta(t) = Cx(t_f)$, where $K(t)$ is a nxn matrix, $F(t)$ is a nxq matrix, $W(t)$ is a qxq matrix, $k(t) \in \Re^n$, and $\eta(t) \in \Re^q$. The resulting noniterative solution procedure is as follows (See Appendix B for the derivation):

Procedure 2.3.2: Solution of linear-quadratic MMOP2 with terminal constraints

---

Step a: Solve backwards from t_f to t_0 the following set of differential equations, with terminal conditions $K(t_f) = \Phi$, $k(t_f) = 0$, $F(t_f) = C^{\top}$, $W(t_f) = 0$, $\eta(t_f) = 0$:

$$\dot{K} = K(t)B\bar{R}^{-1}B^{\top}K(t) - A^{\top}K(t) - K(t)A - \bar{Q}$$
$$\dot{k} = K(t)B\bar{R}^{-1}B^{\top}k(t) - A^{\top}k(t) - K(t)B\bar{R}^{-1}\bar{\lambda}(t) - K(t)\alpha(t) + \bar{\beta}(t)$$
$$\dot{F} = K(t)B\bar{R}^{-1}B^{\top}F(t) - A^{\top}F(t)$$
$$\dot{W} = F(t)^{\top}B\bar{R}^{-1}B^{\top}F(t)$$
$$\dot{\eta} = F^{\top}B\bar{R}^{-1}(B^{\top}k(t) - \bar{\lambda}(t)) - F(t)^{\top}\alpha(t)$$

Step b: Compute the time-varying Kalman gain $G(t)$, $t \in [t_0, t_f]$, multiplier ν and driving input $g(t) \in \Re^m$, $t \in [t_0, t_f]$ from:

$$G(t) = \bar{R}^{-1}B^{\top}K(t)$$
$$\nu = -W(t_0)^{-1}(F(t_0)^{\top}x_0 - \eta(t_0))$$
$$g(t) = -\bar{R}^{-1}(B^{\top}(k(t) + F(t)\nu) - \bar{\lambda}(t))$$

Step c: Compute the state $x(t)$, $t\in[t_0,t_f]$ by integrating from the initial condition $x(t_0) = x_0$ the following differential equation:

$$\dot{x} = (A - BG(t))x(t) + Bg(t) + \alpha(t)$$

Step d: Compute the costate $p(t)$, $t\in[t_0,t_f]$ from:

$$p(t) = K(t)x(t) + k(t) + F(t)\nu$$

Step e: Compute the control input $u(t)$, $t\in[t_0,t_f]$ from the control law:

$$u(t) = -G(t)x(t) + g(t)$$

2.3.3 DISOPE algorithm with LQ model-based problem

DISOPE requires a nominal solution to start the iterations. A recommended starting point is to use the solution of MMOP2 under $\alpha(t)=0$, $\bar{\lambda}(t)=0$, $\bar{\beta}(t)=0$, $t\in[t_0,t_f]$, $r_1 = r_2 = 0$ (relaxed MMOP2).

It is recommended that the relaxation gains k_v, k_z and k_p should initially be set to 1 and the convexification factors should be chosen as $r_1=r_2=0$, and adjusted only if convergence problems arise.

The linear dynamic model (A, B) should approximate the real dynamics represented by f^* in the dynamic range of interest. One way to ensure this is to choose the pair (A, B) as a linearization of $f^*(x,u,t)$ about an appropriate point. The value of the weighting matrices (Q, R) should be such that the resulting L approximates L^* in the range of values of states and controls of interest.

From the above analysis, the following DISOPE algorithm with linear-quadratic model based problem has been proposed (Roberts, 1993a).

Algorithm 2.3.3: Continuous time DISOPE algorithm with LQ model-based problem

---

Data — A, B, Q, R, Φ, r_1, r_2, k_v, k_z, k_p, x_0, t_0, t_f, q and means for calculating f^* and L^*.

Step 0 — Compute or choose a nominal solution $u^0(t)$, $x^0(t)$ and $p^0(t)$. Set i=0, $v^0(t)$=$u^0(t)$, $z^0(t)$=$x^0(t)$, $\hat{p}^0(t)$=$p^0(t)$, $t \in [t_0, t_f]$.

Step 1 — Compute the parameter $\alpha^i(t)$ to satisfy (3,1). This is called the *parameter estimation step.*

Step 2 — Compute the augmented multipliers $\bar{\lambda}^i(t)$ and $\bar{\beta}^i(t)$ from (2,31).

Step 3 — With specified $\alpha(t)$, $\bar{\lambda}(t)$ and $\bar{\beta}(t)$ solve the modified model based optimal control problem MMOP2 (by using Procedure 2.3.1 if q=0 or Procedure 2.3.2 if q>0) to obtain $u^{i+1}(t)$, $x^{i+1}(t)$ and $p^{i+1}(t)$. This is called the *system optimisation step.*

Step 4 — This step tests convergence and updates the estimate for the optimal solution of ROP2. The simple relaxation method (2,21) is employed to satisfy (2,20). If $v^{i+1}(t) = v^i(t)$ within a given tolerance stop, else set i=i+1 and continue from step 1.

In practice, the achievement of the equality $v^{i+1}(t) = v^i(t)$, $t \in [t_0, t_f]$ may be evaluated by using the following 2-norm (*control variation norm* between iterates):

$$\|v^{i+1}-v^{i}\|_2 = \sqrt{\frac{1}{\Delta t}\int_{t_0}^{t_f}\|v^{i+1}(t)-v^{i}(t)\|_i^2\,dt} \qquad \textbf{(2,35)}$$

where Δt is the numerical integration step, and comparing its value with a given small tolerance ε_v.

Notice that Algorithm 2.3.3 solves a sequence of LQ optimal control problems which converge to the solution of ROP2 (which may be nonlinear and non-quadratic). From this point of view, the algorithm may be compared to the Sequential Quadratic Programming technique (Fletcher, 1981), where a sequence of quadratic programming problems (consisting of a quadratic objective function with linear constraints) are solved to find the solution of a nonlinearly constrained mathematical programming problem. It is also noticeable that Algorithm 2.3.3 is an infeasible path approach, in the sense that intermediate solutions $u(t)$, $x(t)$, $t\in[t_0,t_f]$ are not exact solutions of the differential constraint $\dot{x}=f^*(x,u,t)$.

2.4 SIMULATION EXAMPLES

Algorithm 2.3.3 was implemented in the C++ programming language (Becerra, 1993a) using object oriented programming techniques which allow us to handle naturally matrices and their operations (Gorlen *et al*, 1990). A fixed step size fourth order Runge-Kutta integrator (Press *et al*, 1992) was used for solving all the ODE's involved in the algorithm. The trapezoidal rule was used for numerical quadrature. All the derivatives (jacobian matrices, gradients) were computed by using the Central Difference Formula (Press et al, 1992), so avoiding tedious analytical derivative calculations, which are also prone to human errors.

Note that $K(t)$, $G(t)$, $F(t)$, and $W(t)$ need only be computed once and do not change between iterations, provided r_1 and r_2 are kept constant during the process.

The CPU times presented below are for an IBM compatible 486DX-based machine with 33 MHz clock speed.

For further details about the C++ implementation of the continuous time DISOPE algorithm see (Becerra, 1993a).

Example 2.4.1: continuous stirred tank reactor (CSTR).

This example has been taken from two classical books on optimal control (Lapidus and Luus, 1967; Kirk, 1970) and consists of the dynamic optimisation of a first order irreversible chemical reaction carried out under non-isothermal conditions in a continuous stirred tank reactor (CSTR). Control of the reactor is achieved by manipulation of the flow of cooling fluid through a cooling coil inserted in the reactor. Here $x_1(t)$ represents the deviation from the dimensionless temperature, $x_2(t)$ represents the deviation from dimensionless concentration. The control variable $u(t)$ depends on the opening of the valve. It is required to find the optimal input $u(t)$ so as to minimize a quadratic performance index subject to the nonlinear dynamic constraints. The model-based dynamics have been chosen as a linearization about the origin. Here we will test the sensitivity of the algorithm with respect to the tuning parameters r_1 and k_v. The numerical integration step used was $\Delta t = 0.01$ and the tolerance specified for convergence was $\varepsilon_v = 0.01$. The relaxation gains k_z, and k_p were both set to 1 and the convexification factor r_2 was set to zero. The reason for this selection is that current research by Roberts (1993b) indicates that DISOPE is more sensitive to the choice of r_1 and k_v, than to the values of the other tuning parameters.

ROP:

$$\min_{u(t)} \int_0^{0.78} (x_1^2 + x_2^2 + 0.1u^2)\,dt$$

subject to

$$\dot{x}_1 = -(x_1+0.25)+(x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right)-(1+u)(x_1+0.25)$$

$$\dot{x}_2 = 0.5-x_2-(x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right)$$

$$x(0)=[0.05 \quad 0]^T$$

MOP:

$$\min_{u(t)} \int_0^{0.78} (x_1^2+x_2^2+0.1u^2)\,dt$$

subject to

$$\dot{x} = \begin{bmatrix} 4.25 & 1 \\ -6.25 & -2 \end{bmatrix} x(t) + \begin{bmatrix} -0.25 \\ 0 \end{bmatrix} u(t) + \alpha(t)$$

$$x(0)=[0.05 \quad 0]^T$$

Table 2.4.1 shows the algorithm's performance in terms of number of iterations, CPU time and final (J_f^*) performance indexes, for different values of r_1 and k_v. The value of the real performance index for the nominal control $u^0(t)$ (which was obtained from the relaxed MMOP) was $J_0^* = 0.031$. Figure 2.4.1.1 shows the final state responses. Figure 2.4.1.2 shows the nominal and final control signals, where the difference between the initial and final solutions can be appreciated. Figures 2.4.1.3 and 2.4.1.4 show the convergence of the real performance index and of the control variation norm, for the default values $r_1=0$ and $k_v=1$, and the best tested tuning parameter set $r_1=0.1$ and $k_v=0.8$.

r_1	k_v	Number of iterations	CPU (s)	J_f^*
0	1	10	63	0.026617
0.1	1	8	53	0.026618
0.5	1	9	59	0.026647
0	0.8	9	56	0.026617
0	0.3	13	80	0.026616
0.1	0.8	5	34	0.026620

Table 2.4.1: Algorithm's performance for example 2.4.1

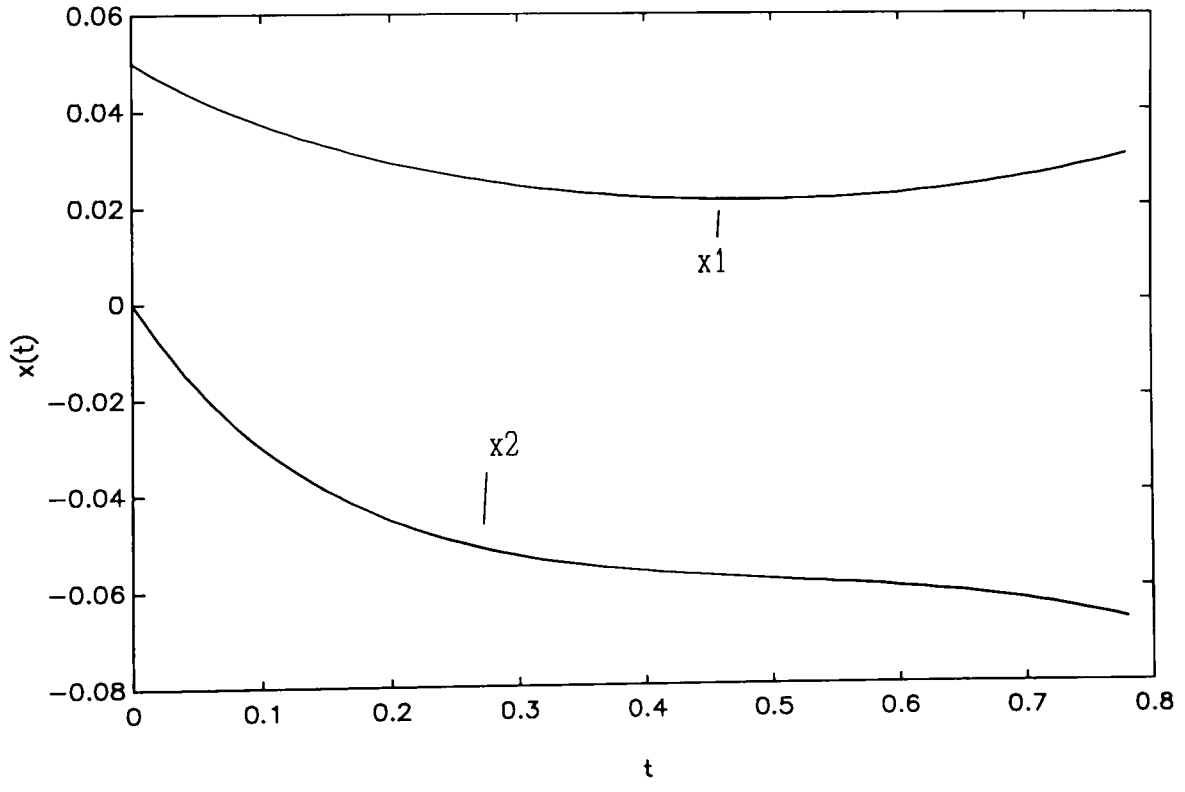

Figure 2.4.1.1: Example 2.4.1, final state vector

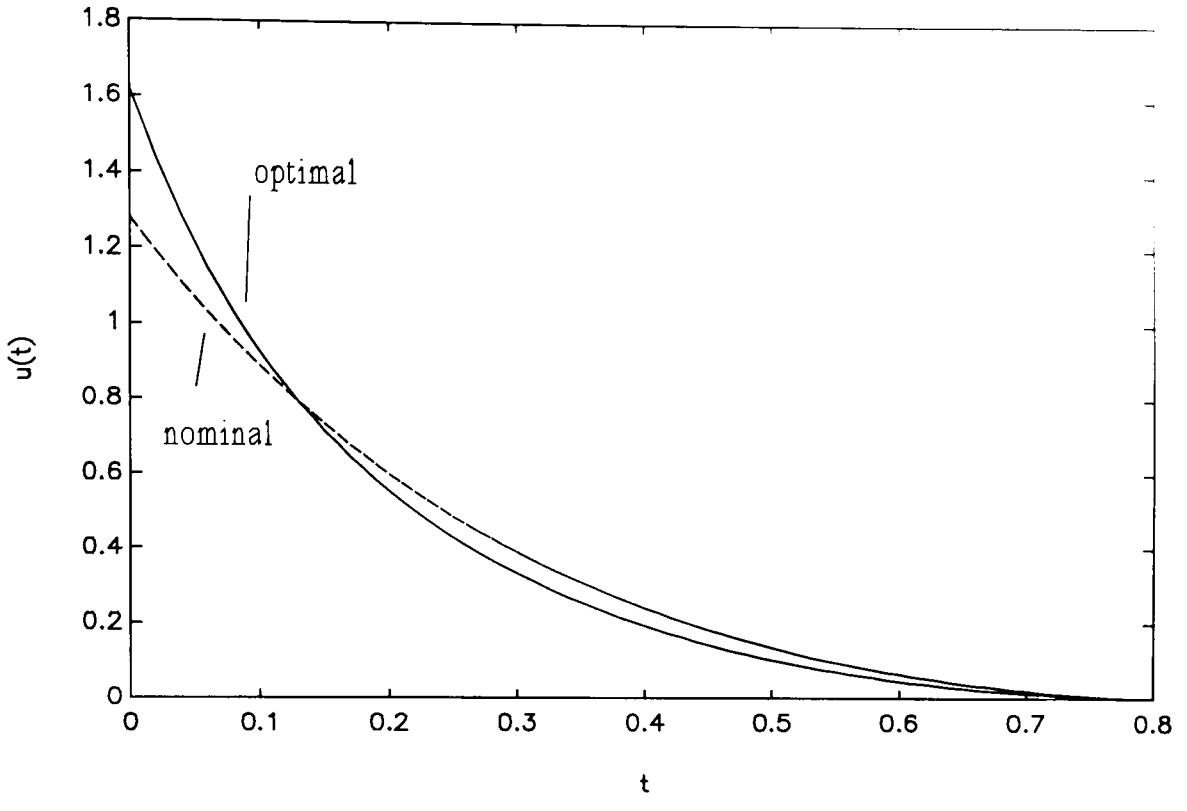

Figure 2.4.1.2: Example 2.4.1, final and nominal control signals

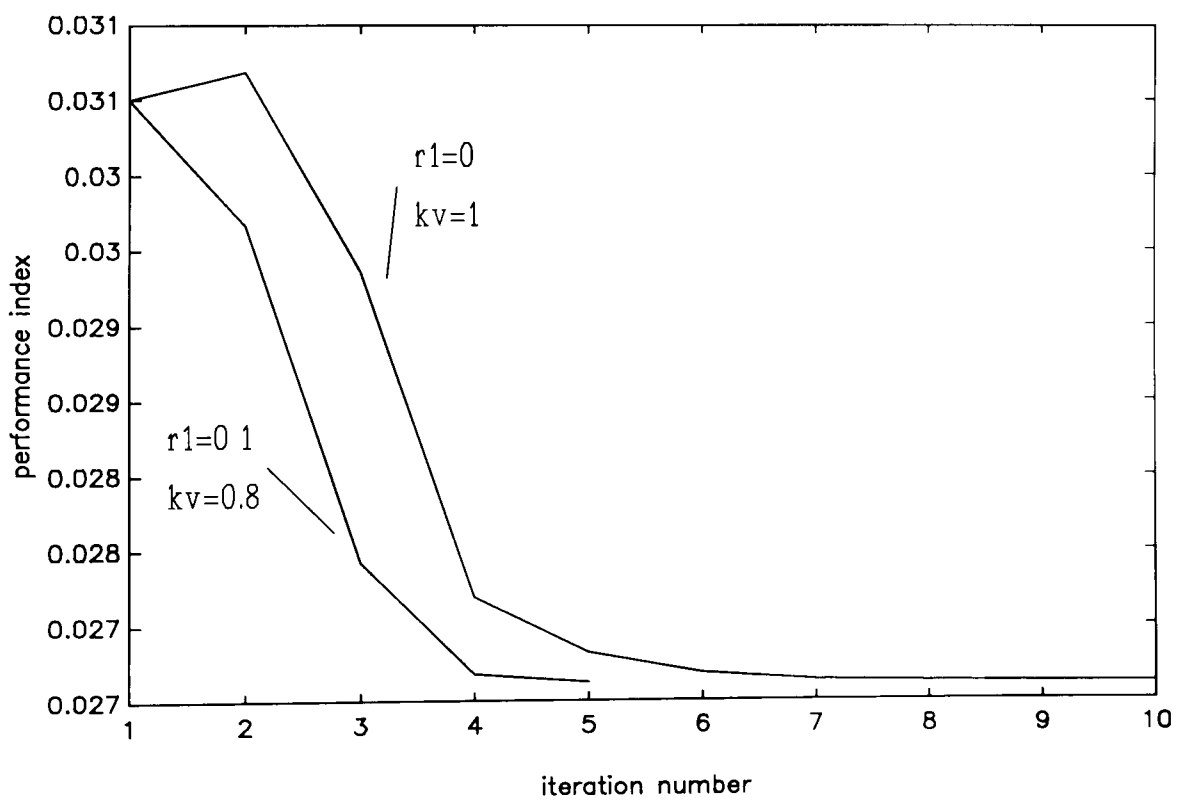

Figure 2.4.1.3: Example 2.4.1, convergence of the real performance index

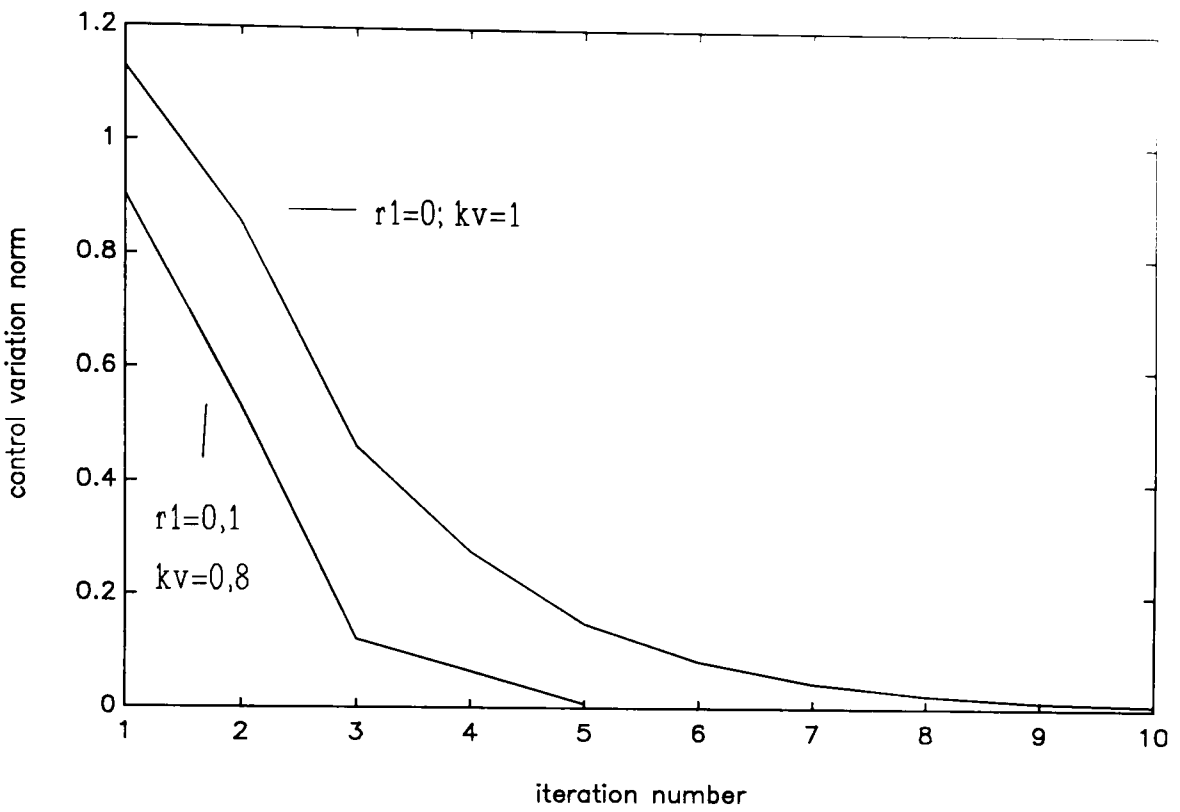

Figure 2.4.1.4: Example 2.4.1, convergence of the control variation norm

It can be seen that the speed of convergence depends on the values of the tuning parameters r_1 and k_v. Furthermore, there must be an optimum pair (r_1, k_v) which provides a minimum number of iterations for convergence. The minimum number of iterations achieved with the combinations tested was 5, while the iterations with the default values $r_1 = 0$ and $k_v = 1$ was 10. This agrees with the results obtained by Roberts (1993b).

We can observe in Figure 2.4.1.3 that the performance index does not decrease monotonically with $r_1 = 0$ and $k_v = 1$, while it does with $r_1 = 0.1$ and $k_v = 0.8$. The control variations are obviously smaller between iterates and convergence is faster with the latter set of parameters, as can be seen in Figure 2.4.1.4.

It is also important to test the sensitivity of the algorithm with respect to the nominal solution. An additional simulation was carried out using the same model-based problem, but the nominal solution was taken as: $u(t) = 0,\ x(t) = x_0,\ p(t) = 0,\ t \in [0, 1]$

using $r_1 = 0$ and $k_v = 1$. Then convergence was achieved after 12 iterations, with an initial performance index $J_0^* = 0.223810$. This result indicates that although the nominal solution obtained from the relaxed MMOP is closer to the optimum, the algorithm was able to find the optimum starting from a gross approximation without a significant increase in the number of iterations required. Furthermore, DISOPE does not require the nominal solution to be very close to the optimum.

Example 2.4.2: Third order nonlinear system

This example consist of the optimisation of a non quadratic performance index subject to nonlinear dynamic equations and a terminal state constraint. The model-based problem consists of a quadratic approximation to the original performance index and a linear approximation to the real dynamics. Here we will illustrate the sensitivity of the algorithm with respect to the linear dynamic approximation used in the model-based problem. The numerical integration step used was $\Delta t = 0.04$ and the tolerance specified for convergence was $\varepsilon_v = 0.01$. Relaxation gains and convexification factors were set to the default values one and zero, respectively. It is noted that the nonlinearities in ROP are substantial and the model-reality differences are considerable.

ROP:

$$\min_{u(t)} \int_0^2 (x_1^4 + x_2^2 + x_3^2 + u_1^2 + u_2^6)\, dt$$

subject to

$$\dot{x}_1 = -x_1 + x_1 x_2 + u_1$$

$$\dot{x}_2 = x_1 - 2x_2 + x_1^3$$

$$\dot{x}_3 = -3x_1 + x_2 + \sin(u_2)$$

$$x(0) = [1.2 \quad 0.0 \quad 1.0]^T; \quad x_1(2) = 0$$

MOP:

$$\min_{u(t)} \int_0^2 (x_1^2 + x_2^2 + x_3^2 + u_1^2 + u_2^2)\, dt$$

subject to

$$\dot{x} = Ax(t) + Bu(t) + \alpha(t)$$

$$x(0)=[1.2 \quad 0.0 \quad 1.0]^{\tau}; \quad x_1(2) = 0$$

Table 2.4.2 shows the model based matrices used and the results obtained in terms of number of iterations for convergence, CPU time, nominal ($x_1^0(2)$) and final ($x_1(2)$) values of the terminal constrained state, and the final value of the real performance index J^*. Notice that in case (a) the dynamic matrices were computed as a linearization of the real dynamics about the origin $x=[0\ 0\ 0]^{\tau}$, $u=[0\ 0]^{\tau}$, while in case (b) these matrices have been (arbitrarily) multiplied by a factor of 2.

Case	A	B	No. Iter.	CPU (s)	$x_1^0(2)$	$x_1(2)$	J^*
a	$\begin{bmatrix} -1 & 0 & 0 \\ 1 & -2 & 0 \\ 0 & 1 & -3 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$	23	177	0.0958	-0.0060	0.4865
b	$\begin{bmatrix} -2 & 0 & 0 \\ 2 & -4 & 0 \\ 0 & 2 & -6 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 \\ 0 & 0 \\ 0 & 2 \end{bmatrix}$	29	221	0.2769	-0.0067	0.4873

Table 2.4.2: Algorithm's performance for example 2.4.2

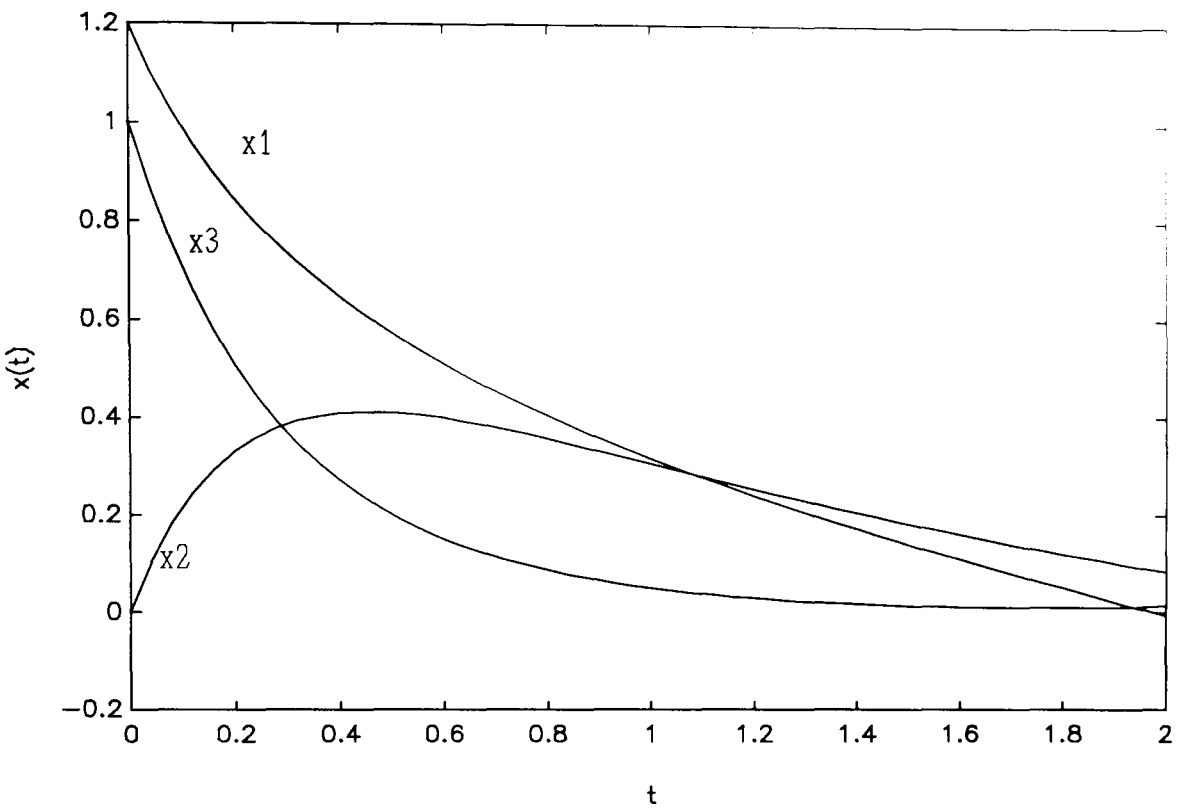

Figure 2.4.2.1: Example 2.4.2, final state response

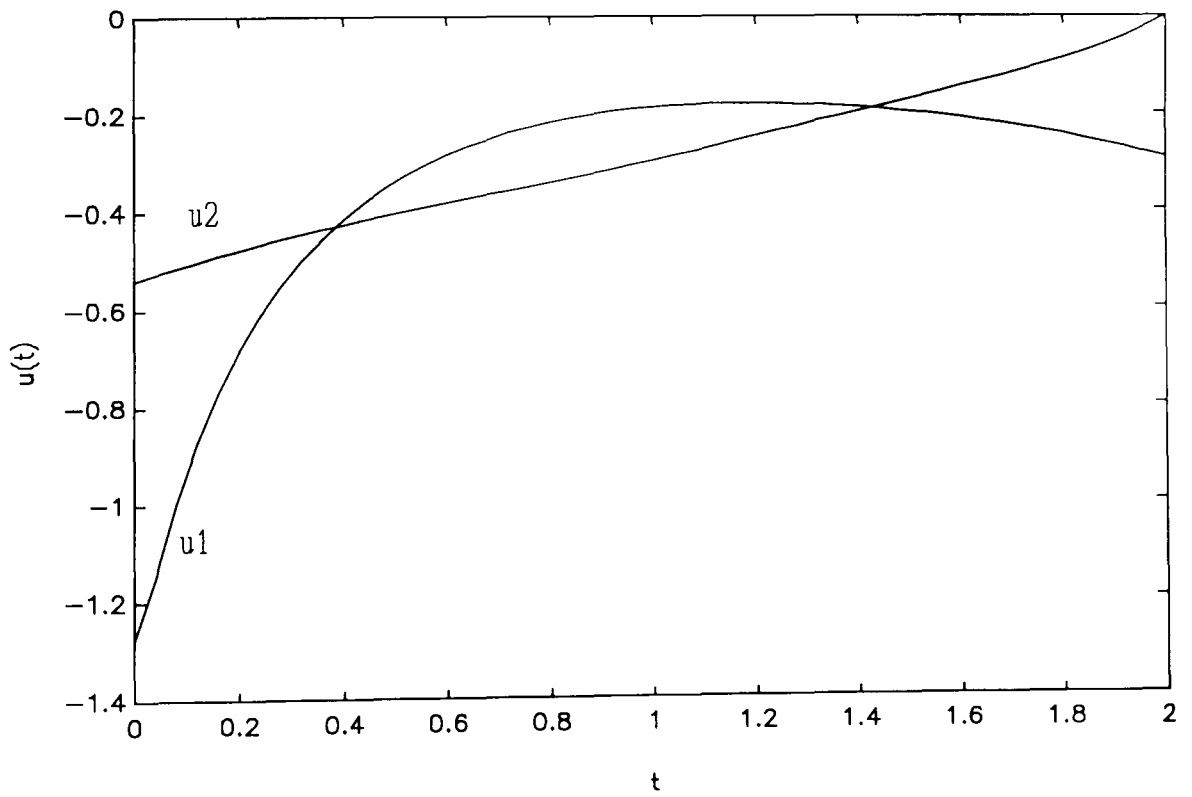

Figure 2.4.2.2: Example 2.4.2, final control signals

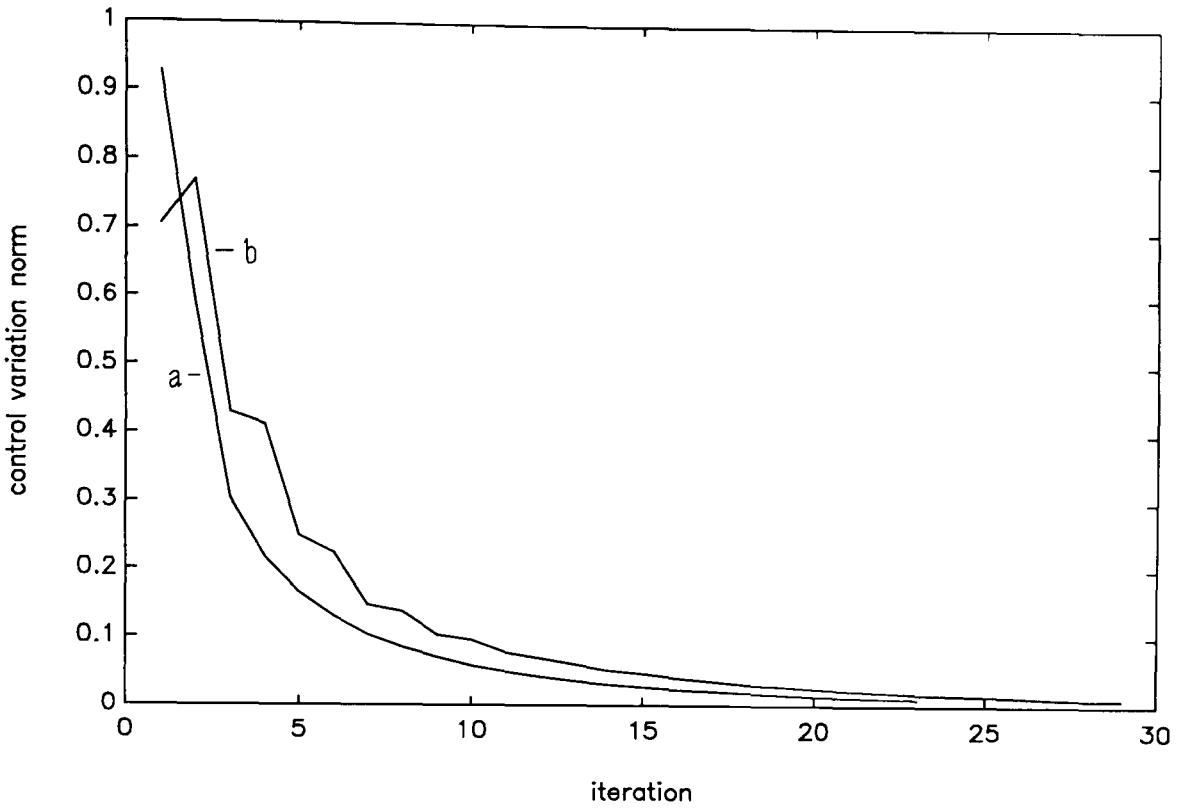

Figure 2.4.2.3: Example 2.4.2, convergence of the control variation norm, cases (a) and (b)

From the results presented it can be seen that in both cases the algorithm achieved the same solution to ROP within the tolerance specified. However, it is observed that case (a) yielded faster convergence. The reason for this is that the linear dynamics in case (a), being a linearization about the origin, are a better approximation to the real dynamics. This is illustrated in Figure 2.4.2.3 where it can be seen that the control variation norm converges monotonically, while in case (b) some oscillations are present. This indicates that different model-based approximations to reality have distinctive convergence behaviour.

Notice that although intermediate iterates $x^i(t)$, $u^i(t)$, $t \in [t_0, t_f]$ satisfy the terminal constraint $x_1(t_f) = 0$, the solution of $\dot{x} = f^*(x, u, t)$ given $u^i(t)$, $t \in [t_0, t_f]$ will only satisfy the terminal condition at the end of the iterations.

2.5 SUMMARY

In this chapter, after a brief introduction to the theory of optimal control, the theoretical development of the DISOPE approach has been presented and the continuous time DISOPE algorithm has been described as originally introduced by Roberts (1992, 1993a). The main mathematical tool used for the derivation is the calculus of variations. Topics such as variable augmentation and handling of terminal state constraints have been treated. Furthermore, a version of continuous time DISOPE with a linear-quadratic model-based problem has been implemented in software. Such an implementation has been used to test the algorithm through simulation examples. The effects of some tuning parameters, such as relaxation gains and convexification factors have been investigated. Moreover, the sensitivity of the algorithm with respect to various factors, such as the initial solution guess and the model-based dynamic approximation of reality, has been illustrated. The capability of the algorithm for solving nonlinear optimal control problems with model-reality differences has been emphasized.

CHAPTER 3

DISOPE WITH CONSTRAINED CONTROLS

In this chapter, an extension of the DISOPE algorithm is developed for handling optimal control problems with input-dependent inequality constraints. The new algorithm is implemented in software, and it is tested with one example with magnitude constraints on the control signal. Alternative ways of handling magnitude constraints are also treated. The research work presented in this chapter is also described in (Becerra and Roberts, 1993).

3.1 OPTIMAL CONTROL WITH INPUT DEPENDENT CONSTRAINTS

The importance of nonlinear control problems with control input dependent constraints is widely recognized (Quintana and Davison, 1974). These constraints arise naturally from the physical limitations of controllers, control valves, actuators and/or processes and are normally called hard constraints, because no violations are allowed at any time. Such constraints must be handled explicitly in any optimisation procedure (Soeterboek, 1992).

3.1.1 Problem formulation

The following formulation extends the real optimal control problem (ROP2) formulated in Section 2.1.1 to accommodate a set of input-dependent constraints. Consider the following fixed terminal time real optimal control problem (ROP3) with input-dependent inequality constraints:

ROP3

$$\min_{u(t)} \quad J^* = \varphi(x(t_f)) + \int_{t_0}^{t_f} L^*(x(t),u(t),t)\ dt$$

subject to

$$\dot{x} = f^*(x(t),\ u(t),\ t)$$
$$x(t_0) = x_0$$
$$C(u(t),t) \leq 0$$

where t_0, t_f, f^*, $x(t)$, $u(t)$, L^*, φ are defined in Section 2.1.1 and $C : \Re^n \times \Re^n \rightarrow \Re^n$ is a set of input-dependent inequality constraints.

3.1.2 Necessary optimality conditions (The minimum principle)

The necessary optimality conditions of ROP3 are given in terms of the Hamiltonian (2,3) and it was shown by Pontryagin et al (1962) that the costate equation (2,5), state equation (2,6) and boundary conditions (2,7) still hold as necessary conditions of optimality, but at all points on $C=0$ (i.e. the constraint is *active*) the optimal u has the property that (Bryson and Ho, 1975)

$$\delta H^* = \nabla_u H^{*T} \delta u \geq 0$$
$$\delta C = \frac{\partial C^T}{\partial u} \delta u \leq 0 \qquad \textbf{(3,1)}$$

Furthermore if $C<0$

$$\nabla_u H^* = 0 \qquad \textbf{(3,2)}$$

The above conditions can be stated as "H^* must be minimized over the set of all possible u". This is known as Pontryagin's Minimum Principle (Pontryagin *et al*, 1962). An equivalent approach to the above formulation is to define (Bryson and Ho, 1975):

$$H' = L(x,u,t) + p^T f(x,u,t) + \Theta^T C(u,t) \qquad \textbf{(3,3)}$$

where $\Theta(t) \in \Re^n$ is a Kuhn-Tucker multiplier which has the requirement that:

$$\Theta(t) \begin{cases} >0, & C(u,t)=0 \\ =0, & C(u,t)<0 \end{cases} \qquad \textbf{(3,4)}$$

then the stationarity condition on H' is:

$$\nabla_u H' = \nabla_u L + \frac{\partial f^T}{\partial u} p(t) + \frac{\partial C^T}{\partial u} \Theta(t) = 0 \qquad \textbf{(3,5)}$$

Notice that as $\nabla_x H' = \nabla_x H$ and $\nabla_p H' = \nabla_p H$ then equations (2,5) and (2,6) also apply as necessary optimality conditions in this alternative formulation.

3.2 DYNAMIC ISOPE APPROACH

In Chapter 2, we derived the DISOPE algorithm by using variational calculus. In that case we assumed that there were no constraints on the values the control signal may achieve. In this Section we shall take into account such constraints in the formulation by using the minimum principle stated above.

3.2.1 Problem formulation and solution

Instead of solving the real problem ROP3 the following possibly simplified model based problem (MOP3) is considered:

MOP3

$$\min_{u(t)} \quad J_m = \varphi(x(t_f)) + \int_{t_0}^{t_f} L(x(t),\ u(t),\ \gamma(t))\ dt$$

subject to

$$\dot{x} = f(x(t),\ u(t),\ \alpha(t))$$
$$x(t_0) = x_0$$
$$C(u(t),t) \leq 0$$

where J_m is a model-based performance index, L is a continuous weighting function and perhaps a simplification of a known L^*, f is an approximated dynamic model of f^*, $\gamma(t)\in\Re$ and $\alpha(t)\in\Re^r$ are continuous parameters.

Now, an expanded optimal control problem (EOP3), which is equivalent to the real optimal control problem ROP3, is defined as follows:

EOP3

$$\min_{u(t)} \quad J_e = \varphi(x(t_f)) + \int_{t_0}^{t_f} L(x(t),\ u(t),\ \gamma(t))\ dt$$

subject to

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

$$x(t_0) = x_0$$

$$C(u(t),t)\leq 0$$

$$f(z(t),v(t),\alpha(t)) = f^*(z(t),v(t),t)$$

$$L(z(t),v(t),\gamma(t)) = L^*(z(t),v(t),t)$$

$$u(t) = v(t)$$

$$x(t) = z(t)$$

where the role of $z(t)$, $v(t)$ and that of the additional equality constraints has already been discussed in Section 2.1.1.

Adjoining constraints, we obtain the following augmented performance index J_e':

$$\begin{aligned} J_e' = \varphi(x(t_f)) + \int_{t_0}^{t_f} [\ & L(x,u,\gamma) + p^{\top}(f(x,u,\alpha)-\dot{x}) + \lambda^{\top}(v-u) \\ & + \beta^{\top}(z-x) + \mu^{\top}(f^*(z,v,t)-f(z,v,\alpha)) \\ & + \xi(L^*(z,v,t)-L(z,v,\gamma)) + \Theta^{\top}C(u,t)\ ]\ dt \end{aligned} \qquad \textbf{(3,6)}$$

where the time index has been dropped for convenience, $p(t)\in\Re^n$, $\lambda(t)\in\Re^m$, $\beta(t)\in\Re^n$, $\mu(t)\in\Re^m$ and $\xi(t)\in\Re$ are Lagrange multipliers and $\Theta(t)\in\Re^{n_c}$ is a Kuhn-Tucker multiplier such that (3,4) is satisfied.

Notice that by adjoining the inequality constraint $C(u,t)\leq 0$ with the Kuhn-Tucker multiplier $\Theta(t)$ we are firstly eliminating the inequality constraint such that the control variations become free, and secondly we are implicitly applying the minimum principle.

Define a function (the augmented Hamiltonian):

$$H = L(x,u,\gamma) + p^{\top} f(x,u,\alpha) - \lambda^{\top} u - \beta^{\top} x + \Theta^{\top} C(u,t) \qquad \textbf{(3,7)}$$

Thus, by applying the calculus of variations to (3,6) and taking into account the new definition of the augmented Hamiltonian H it is easy to see that the necessary optimality conditions of EOP3 are identical to those obtained for EOP2, namely (2,14) to (2,20). As a consequence, it follows that given values of $\alpha(t)$, $\gamma(t)$, $\lambda(t)$ and $\beta(t)$, the solution of the following problem satisfies the definition of the augmented Hamiltonian (3,7) and also the model-based optimality conditions (2,14), (2,15) and (2,16) plus the border conditions (2,17).

MMOP3

$$\min_{u(t)} \quad J_M = \varphi(x(t_f)) + \int_{t_0}^{t_f} [\, L(x(t),\, u(t),\, \gamma(t)) - \lambda(t)^{\top} u(t) - \beta(t)^{\top} z(t) \,]\, dt$$

subject to

$$\dot{x} = f(x(t),\, u(t),\, \alpha(t))$$
$$x(t_0) = x_0$$
$$C(u(t),t)\leq 0$$

The above analysis gives rise to the following DISOPE algorithm with inequality constraints on the control variables.

Define:

$$\Omega = \{\, u_j(t) : C(u,t)\leq 0,\ j\in[1,m],\ t\in[t_0,t_f]\} \qquad \textbf{(3,8)}$$

as the set of admissible control trajectories.

Algorithm 3.2.1: DISOPE algorithm with input dependent inequality constraints

---

Data — f, L, φ, x_0, t_0, t_f, C, and means for calculating f^* and L^*.

Step 0 — Compute or choose a nominal solution $u^0(t) \in \Omega$, $x^0(t)$ and $p^0(t)$. Set $i=0$, $v^0(t)=u^0(t)$, $z^0(t)=x^0(t)$, $\hat{p}^0(t)=p^0(t)$, $t \in [t_0, t_f]$.

Step 1 — Compute the parameters $\alpha^i(t)$, $\gamma^i(t)$ to satisfy (2,19). This is called the *parameter estimation step.*

Step 2 — Compute the multipliers $\lambda^i(t)$ and $\beta^i(t)$ from (2,20).

Step 3 — With specified $\alpha(t)$, $\gamma(t)$, $\lambda(t)$ and $\beta(t)$ solve the modified model-based optimal control problem MMOP3 to obtain $u^{i+1}(t)$, $x^{i+1}(t)$ and $p^{i+1}(t)$. This is called the *system optimisation step.* This step should be performed by an optimal control algorithm capable of handling input dependent inequality constraints.

Step 4 — This step tests convergence and updates the estimate for the optimal solution of ROP3. In order to provide a mechanism for regulating convergence, the simple relaxation method (2,21) may be employed to satisfy (2,20). If $v^{i+1}(t) = v^i(t)$ within a given tolerance stop, else set $i=i+1$ and continue from step 1.

3.3 CASE OF SIMPLE BOUNDS ON THE CONTROLS

It is of particular practical importance the case when the control input magnitude is bounded at upper and lower levels (i.e. $u_{\min_j} \leq u_j \leq u_{\max_j}$, $j \in [1,m]$). In this case, we have the following set of control input inequality constraints:

$$\begin{cases} u_j - u_{\max_j} \le 0 \\ -u_j + u_{\min_j} \le 0 \end{cases} \quad j \in [1,m] \tag{3,9}$$

One way of handling magnitude constraints on the control inputs is the use of a variable transformation technique. Several authors have proposed the use of variable transformations to convert the constrained problem into an unconstrained one in a new control variable (Sisirena and Tan, 1974). A particular transformation suitable to be used within the DISOPE framework is the following vector saturation function:

$$u(t) = SAT(\bar{u}(t)) = \begin{bmatrix} sat(\bar{u}_1) \\ \cdots \\ sat(\bar{u}_m) \end{bmatrix} \tag{3,10}$$

where $\bar{u}(t)$ is the transformed (unconstrained) control variable, $u(t)$ is the constrained control variable, and the scalar saturation function $sat(.)$ is given by:

$$sat(u_j) = \begin{cases} u_{\max_j} & u_j > u_{\max_j} \\ u_j & u_{\min_j} \le u_j \le u_{\max_j} \\ u_{\min_j} & u_j < u_{\min_j} \end{cases} \tag{3,11}$$

By using such a transformation, unconstrained DISOPE (i.e. Algorithm 2.3.3) and hence iterative LQ solution methods may be applied to the transformed problem.

3.4 SIMULATION EXAMPLES

Recall Algorithm 3.2.1 and notice that it requires that step 3 must be solved by using an optimal control algorithm capable of handling control dependent inequality constraints. As mentioned in Section 3.3, bound or magnitude constraints on the controls are a particular case of the general inequality constraints $C(u,t) \le 0$ which has practical relevance. In this work, Algorithm 3.2.1 has been implemented

for such a particular case. A multiple-input extension of the single-input conjugate-gradient algorithm presented by Quintana and Davison (1974) is used as an auxiliary algorithm for solving MMOP3 at every iteration of DISOPE. The algorithm (described in Appendix C) is easy to implement and its convergence has been proven for any arbitrary and feasible initial estimate of the optimal control. The key concepts of this algorithm are the numerical integration of the state and costate equations, and a gradient in function space to update the controls which are clipped-off at the bounds so as to minimize the Hamiltonian. The algorithm has shown good performance and has been compared favourably with other constrained optimal control algorithms (Jones and Finch, 1984).

The following simulations were run on an IBM compatible 486DX-based microcomputer with 33 MHz clock speed. The example was solved by two methods:

(a) Algorithm 3.2.1 (DISOPE with constrained controls), using a conjugate-gradient algorithm to find the solution of MMOP3 at every iteration.

(b) Algorithm 2.3.3 (DISOPE with LQ model-based problem), using a variable transformation to handle the control bounds.

More details on the conjugate-gradient algorithm used, together with the definition of its tuning parameters may be found in Appendix C. The tolerances specified for the conjugate-gradient algorithm were $\varepsilon_1 = 0.05$ and $\varepsilon_2 = 0.05$, resetting the algorithm to steepest descent every 3 iterations.

Example 3.4.1: continuous stirred tank reactor (CSTR) with bounded control

This example consists of the same dynamic equations and performance index as in Example 2.4.1, but here the control signal is bounded between upper and lower levels $-1 \le u(t) \le 1$. The model-based dynamics have been chosen as a linearization about the origin. The numerical integration step used was $\Delta t = 0.01$ and the tolerance specified for convergence of DISOPE was $\varepsilon_v = 0.01$. The relaxation gains k_z, and k_p were both set to 1 and the convexification factor r_2 was set to zero. The values of k_v and r_1 were set to 1 and 0, in method (a), and 0.9 and 0.5 in method (b).

ROP:

$$\min_{u(t)} \int_0^{0.78} (x_1^2 + x_2^2 + 0.1u^2)\,dt$$

subject to

$$\dot{x}_1 = -(x_1 + 0.25) + (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right) - (1 + u)(x_1 + 0.25)$$

$$\dot{x}_2 = 0.5 - x_2 - (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right)$$

$$x(0) = [0.05 \quad 0]^T$$

$$-1 \leq u(t) \leq 1$$

MOP:

$$\min_{u(t)} \int_0^{0.78} (x_1^2 + x_2^2 + 0.1u^2 + \gamma)\,dt$$

subject to

$$\dot{x} = \begin{bmatrix} 4.25 & 1 \\ -6.25 & -2 \end{bmatrix} x(t) + \begin{bmatrix} -0.25 \\ 0 \end{bmatrix} u(t) + \alpha(t)$$

$$x(0) = [0.05 \quad 0]^T$$

$$-1 \leq u(t) \leq 1$$

The performances of the methods being tested (Algorithms 3.2.1 and 2.3.3) are presented in Table 3.4.1. The final control signals are compared in Figure 3.4.1.1. The convergence behaviour is illustrated in Figures 3.4.1.2 and 3.4.1.3.

Algorithm	Number of DISOPE iterations	Number of performance index evaluations	CPU (s)	J_0^*	J^*
(a) 3.2.1	6	98	281	0.053664	0.028952
(b) 2.3.3	11	11	71	0.053644	0.028953

Table 3.4.1: Performances for example 3.4.1

It can be seen in Table 3.4.1 that even though Algorithm 3.2.1 required a lower number of iterations for convergence, the CPU time it used was significantly higher than that used by Algorithm 2.3.3. The CPU time per iteration in Algorithm 3.2.1 was about 5 times higher than that in Algorithm 2.3.3, which is explained by the iterative nature of the conjugate-gradient algorithm used for solving MMOP3, as opposed by the non-iterative LQ solution procedure used in Algorithm 2.3.3 (Procedure 2.3.1). Notice that the performance index evaluations in Algorithm 2.3.3 are carried out for later analysis, and these are not needed for the algorithm's calculations. Notice also that the changes in the performance index are very small after a few iterations with Algorithm 2.3.3. However, the convergence of the control variations is slower than that of the performance index. This occurs because changes in the (unconstrained) model-based control are sensed by DISOPE, but the saturation function filters such changes when the performance index is computed.

It can be checked that the solutions obtained satisfy the necessary optimality conditions (within the tolerances specified) and, therefore, the constrained DISOPE algorithm achieved the correct optimal solution in spite of the model-reality differences. This verifies, by means of simulations, the validity of Algorithm 3.2.1 as well as the usefulness of the variable transformation technique.

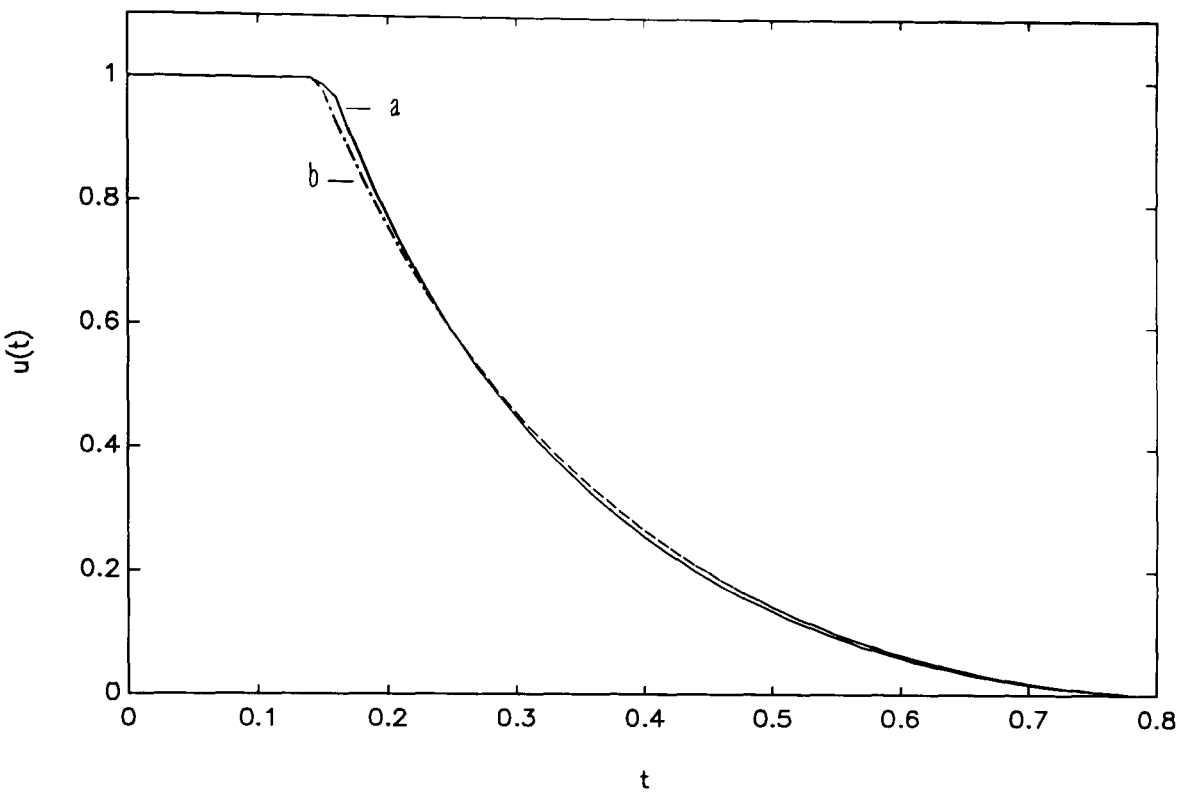

Figure 3.4.1.1: Example 3.4.1, final control signal

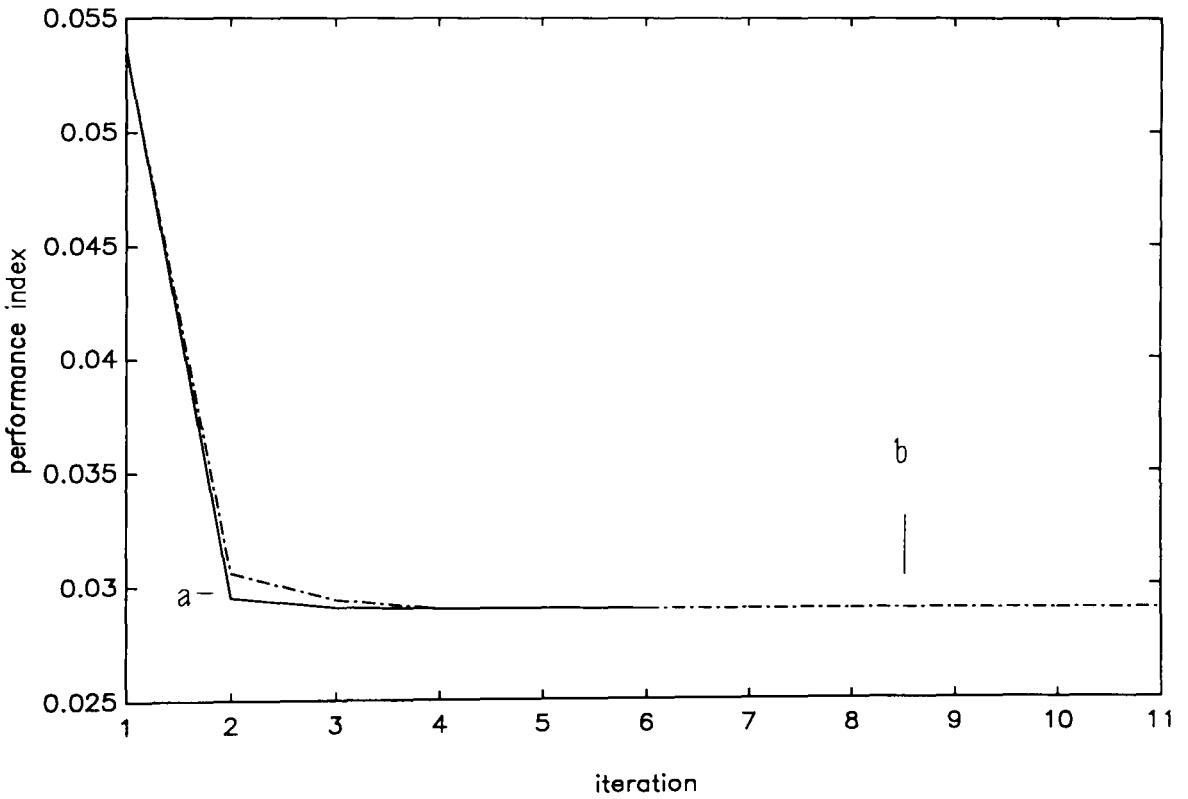

Figure 3.4.1.2: Example 3.4.1, Convergence of the performance index

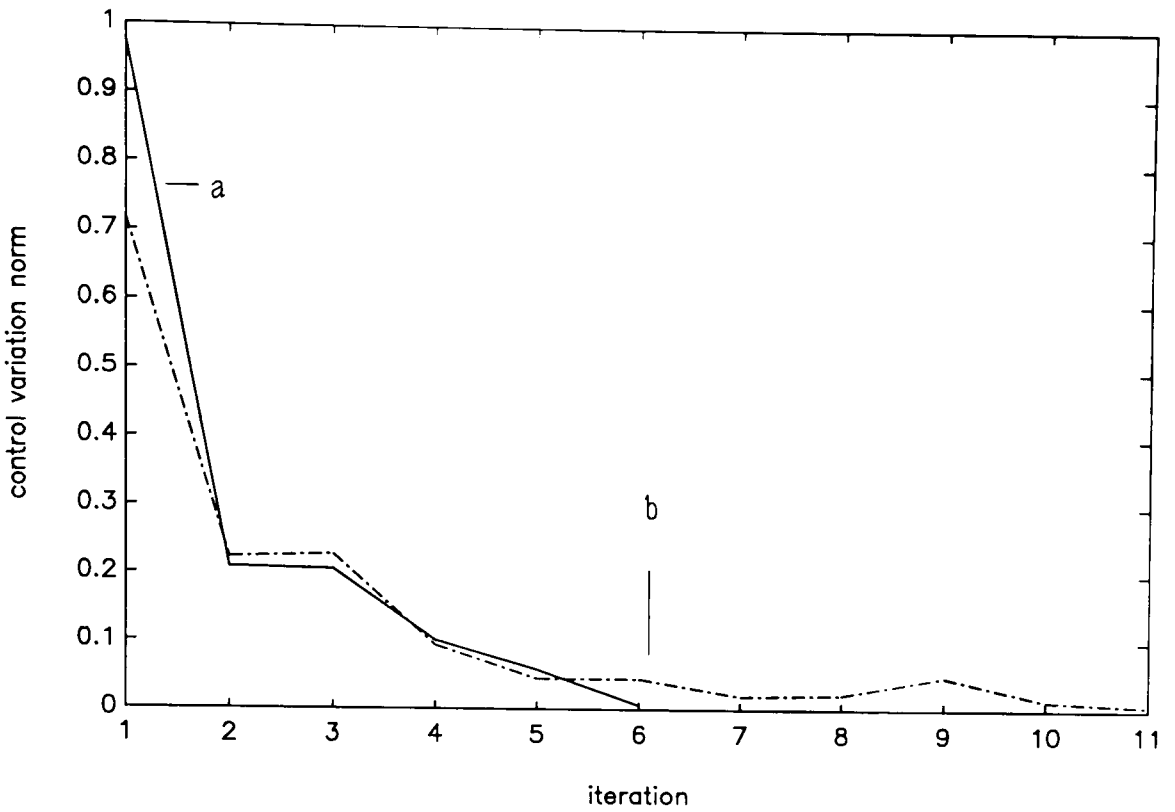

Figure 3.4.1.3: Example 3.4.1, control variation norm versus iterations

3.5 SUMMARY

An extension of the DISOPE algorithm for handling optimal control problems with general control dependent inequality constraints has been developed. The original DISOPE algorithm remains basically unchanged when including the control dependent constraints, the important differences being the specification of a feasible nominal solution and the explicit handling of the constraints when solving MMOP3.

The algorithm developed has been implemented in software, for the particular case of control magnitude bounds, using a conjugate-gradient algorithm for solving the constrained modified model-based problem. Additionally, a variable transformation technique which converts problems with simple bound constraints into unconstrained problems has been tested. The transformed unconstrained problems were solved by using the DISOPE algorithm with unconstrained LQ model based problem (Algorithm 2.3.3).

The implemented constrained DISOPE algorithm and the variable transformation technique have been tested with one example with bounded control input. The results indicate that the variable transformation technique together with the unconstrained version of DISOPE is a more efficient alternative for handling

bound constraints from the computational time point of view, than using the constrained version of DISOPE together with the conjugate-gradient algorithm, even though the constrained DISOPE algorithm required less iterations for convergence.

CHAPTER 4

HIERARCHICAL DISOPE ALGORITHM

In this chapter, a hierarchical DISOPE algorithm for solving large-scale nonlinear optimal control problems with model-reality differences is developed. The approach used is based on the interaction-prediction approach and on the centralized DISOPE technique. A new multiplier is introduced to take into account the constraints related with the interactions between subsystems. A version of the algorithm with a linear-quadratic model-based problem is developed and implemented in software. The technique is suitable for parallel or distributed processing. The algorithm implemented is tested with one simulation example. The research work presented in this chapter is also described in (Becerra, 1993c).

4.1 LARGE-SCALE SYSTEMS AND HIERARCHICAL CONTROL

Systems complexity in many real-life plants and processes has led to a new class of systems theory called *large-scale systems*. A system is considered as being of large-scale when it can be decomposed into a finite number of subsystems or when it is distributed in such a way that the concept of centrality does not hold. Most large-scale industrial processes consist of interconnected subsystems or sub-processes according to workshops, units, functions and geographical positions. Examples may be found in several industries such as chemical, petrochemical, electrical power, etc. One class of control of large-scale industrial processes is the hierarchical one, where decision units, which are positioned at upper levels in the hierarchy, control or coordinate the process units or subsystems located at the lower levels. The control or coordination functions are normally performed by a set of computers (decision units) connected in a hierarchical or multilevel structure. Hierarchical optimisation of dynamic systems may yield substantial computational savings (when compared to centralized dynamic optimisation), in both storage and computer time and these benefits increase when parallel processing is used (Jamshidi, 1983; Singh, 1980).

4.2 PROBLEM FORMULATION AND SOLUTION APPROACH

Consider the large-scale interconnected system described by the following time-varying ordinary differential equation:

$$\begin{aligned} \dot{x} &= f^*(x(t), u(t), t) \\ x(t_0) &= x_0 \end{aligned} \tag{4,1}$$

where $u(t) \in \Re^m$, $x(t) \in \Re^n$ represent the overall control and state vectors respectively. $x_0 \in \Re^n$ represents a given initial state, and $f^* : \Re^n \times \Re^m \times \Re \rightarrow \Re^n$ represents the overall dynamic system.

This large-scale system is decomposed into N sub-systems (Jamshidi, 1983), giving the following dynamic equation for the *ith* subsystem:

$$\begin{aligned} \dot{x}_i &= f_i^*(x_i(t), u_i(t), t) + \zeta_i(x(t), t) \\ x_i(0) &= x_{i0} \end{aligned} \tag{4,2}$$

where $u_i(t) \in \Re^{m_i}$, $x_i(t) \in \Re^{n_i}$ represent the *ith* control and state vectors respectively, $x_{i0} \in \Re^{n_i}$ represents the subsystem initial state, $f_i^* : \Re^{n_i} \times \Re^{m_i} \times \Re \rightarrow \Re^{n_i}$ represents the subsystem's dynamics, $\zeta_i : \Re^n \times \Re \rightarrow \Re^{n_i}$ represent the interactions or interconnections between subsystems and is given by

$$\zeta_i(x(t), t) = \sum_{j=1}^{N} \zeta_{ij}(x_j(t), t) \tag{4,3}$$

The objective is to find the control vectors $u_1(t) \ldots u_N(t)$ so that the following overall performance index is minimized:

$$J^* = \varphi(x(t_f)) + \int_{t_0}^{t_f} L^*(x(t),u(t),t)\ dt \qquad \textbf{(4,4)}$$

where $\varphi : \Re^n \rightarrow \Re$ is a given terminal measure and $L^* : \Re^n \times \Re^m \times \Re \rightarrow \Re$ is the overall performance measure function.

It is supposed that the overall performance index is additively separable:

$$J^* = \sum_{i=1}^{N} J_i^* = \sum_{i=1}^{N} \{\ \varphi_i(x_i(t_f)) + \int_{t_0}^{t_f} L_i^*(x_i(t),u_i(t),t)\ dt\ \} \qquad \textbf{(4,5)}$$

where $\varphi_i : \Re^{n_i} \rightarrow \Re$ is the *ith* terminal measure and $L_i^* : \Re^n \times \Re^m \times \Re \rightarrow \Re$ is the *ith* performance measure function.

Then, the large-scale system optimal control problem can be written as:

$$\min_{\substack{u_i(t) \\ i \in [1,N]}} J^* = \sum_{i=1}^{N} J_i^* = \sum_{i=1}^{N} \{\ \varphi_i(x_i(t_f)) + \int_{t_0}^{t_f} L_i^*(x_i(t),u_i(t),t)\ dt\ \}$$

subject to

$$\dot{x}_i(t) = f_i^*(x_i(t),u_i(t),t) + \theta_i(w_i(t),t)$$

$$x_i(t_0) = x_{io}$$

$$\theta_i(w_i(t),t) = \sum_{j=1}^{N} \zeta_{ij}(x_j(t),t)$$

where $\theta_i \in \Re^{n_i}$ and $w_i \in \Re^{s_i}$ represent the interactions between subsystems.

Now, it is assumed that the interactions are linear. This is:

$$\theta_i(w_i(t),t) = C_i w_i(t)$$

$$w_i(t) = \sum_{j=1}^{N} M_{ij} x_j(t) \qquad \textbf{(4,6)}$$

where $C_i \in \Re^{n_i \times s_i}, M_{ij} \in \Re^{s_i \times n_j}$.

The following formulation is based on the interaction-prediction method (Singh, 1980; Jamshidi, 1983; Mahmoud et al, 1985) and on the centralized continuous time version of the DISOPE algorithm (Roberts, 1992, 1993a; see Chapter 2). After the above analysis and assumptions, consider the following large-scale system real optimal control problem (ROP4):

ROP4

$$\min_{\substack{u_i(t) \\ i\in[1,N]}} J^* = \sum_{i=1}^{N} J_i^* = \sum_{i=1}^{N} \{ \varphi_i(x_i(t_f)) + \int_{t_0}^{t_f} L_i^*(x_i(t),u_i(t),t)\, dt \}$$

subject to

$$\dot{x}_i = f_i^*(x_i(t),u_i(t),t) + C_i w_i$$

$$x_i(t_0)=x_{io}$$

$$w_i(t) = \sum_{j=1}^{N} M_{ij} x_j(t)$$

Instead of solving ROP4, the following possibly simplified large-scale system model-based optimal control problem (MOP4) is considered:

MOP4

$$\min_{\substack{u_i(t) \\ i\in[1,N]}} J_m = \sum_{i=1}^{N} J_{mi} = \sum_{i=1}^{N} \{ \varphi_i(x_i(t_f)) + \int_{t_0}^{t_f} L_i(x_i(t),u_i(t),\gamma_i(t))\, dt \}$$

subject to

$$\dot{x}_i = f_i(x_i(t),u_i(t),\alpha_i(t))$$

$$x_i(t_0)=x_{io}$$

where $L_i : \Re^{n_i}\times\Re^{m_i}\times\Re\rightarrow\Re$ is a model-based performance measure function, $f_i : \Re^{n_i}\times\Re^{m_i}\times\Re^{r_i}\rightarrow\Re^{n_i}$ represents a model of f_i^*, $\alpha_i(t)\in\Re^{r_i}$ and $\gamma_i(t)\in\Re$ are continuous parameters. Notice that the dynamic equation in MOP4 does not have an interaction term. It will be clarified later that the set of parameters $\alpha_i(t)$, $i\in[1,N]$

will not only take into account the model-reality differences in value between f_i and f_i^* but also the interactions or interconnections between subsystems.

Now, a large-scale system expanded optimal control problem (EOP4), which is equivalent to ROP4, is considered:

EOP4

$$\min_{\substack{u_i(t) \\ i\in[1,N]}} \quad J_e = \sum_{i=1}^{N} J_{ei} = \sum_{i=1}^{N} \{ \varphi_i(x_i(t_f)) + \int_{t_0}^{t_f} L_i(x_i(t), u_i(t), \gamma_i(t)) \, dt \}$$

subject to

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t), \alpha_i(t))$$

$$x_i(t_0) = x_{io}$$

$$f_i(z_i(t), v_i(t), \alpha_i(t)) = f_i^*(z_i(t), v_i(t), t) + C_i w_i(t)$$

$$L_i(z_i(t), v_i(t), \gamma_i(t)) = L_i^*(z_i(t), v_i(t), t)$$

$$v_i(t) = u_i(t)$$

$$z_i(t) = x_i(t)$$

$$w_i(t) = \sum_{j=1}^{N} M_{ij} x_j(t)$$

where $v(t)\in\Re^{m_i}$ and $z(t)\in\Re^{n_i}$ are introduced as separation variables. Adjoining constraints in EOP4,

$$\begin{aligned} J_e' = \sum_{i=1}^{N} \{ \varphi_i(x_i(t_f)) + \int_{t_0}^{t_f} [& L_i(x_i,u_i,\gamma_i) + p_i^{\mathrm{T}}(f_i(x_i,u_i,\alpha_i) - \dot{x}_i) \\ & + \lambda_i^{\mathrm{T}}(v_i - u_i) + \beta_i^{\mathrm{T}}(z_i - x_i) + \mu_i^{\mathrm{T}}(f_i^*(z_i,v_i,t) + C_i w_i - f_i(z_i,v_i,\alpha_i)) \\ & + \xi(L_i^*(z_i,v_i,t) - L_i(z_i,v_i,\gamma_i)) + \Omega_i^{\mathrm{T}}(w_i - \sum_{j=1}^{N} M_{ij} x_j)] dt \} \end{aligned} \qquad \textbf{(4,7)}$$

where $p_i(t)\in\Re^{n_i}$, $\lambda_i(t)\in\Re^{m_i}$, $\beta_i(t)\in\Re^{n_i}$, $\mu_i(t)\in\Re^{n_i}$, $\xi_i(t)\in\Re$ and $\Omega_i(t)\in\Re^{N}$ are Lagrange multiplier functions. Define:

$$H_i = L_i(x_i, u_i, \gamma_i) + p_i^{\top} f(x_i, u_i, \alpha_i) - \lambda_i^{\top} u_i - \beta_i^{\top} x_i - \sum_{j=1}^{N} \Omega_j^{\top} M_{ji} x_i \qquad \textbf{(4,8)}$$

Following a similar procedure to that outlined in Section 2.2.1, we then use (4,8) and apply calculus of variations to (4,7). After concluding that $\mu_i(t) = p_i(t)$ and $\xi_i(t) = 1$, the necessary optimality conditions presented below are obtained:

Stationarity

$$\nabla H_{u_i} = 0 \qquad \textbf{(4,9)}$$

Costate equation

$$\nabla H_{x_i} + \dot{p}_i = 0 \qquad \textbf{(4,10)}$$

State equation

$$\dot{x}_i = f_i(x_i(t), u_i(t), \alpha_i(t)) \qquad \textbf{(4,11)}$$

Boundary conditions

$$\begin{aligned} x_i(t_0) &= x_{io} \\ p_i(t_f) &= \nabla_{x_i} \varphi_i(x_i(t_f)) \end{aligned} \qquad \textbf{(4,12)}$$

Multiplier equations

$$\begin{aligned} \lambda_i(t) &= [\frac{\partial f_i}{\partial v_i} - \frac{\partial f_i^*}{\partial v_i}]^{\top} \hat{p}_i(t) + [\nabla_{v_i} L_i - \nabla_{v_i} L_i^*] \\ \beta_i(t) &= [\frac{\partial f_i}{\partial z_i} - \frac{\partial f_i^*}{\partial z_i}]^{\top} \hat{p}_i(t) + [\nabla_{z_i} L_i - \nabla_{z_i} L_i^*] \end{aligned} \qquad \textbf{(4,13)}$$

$$\Omega_i(t) = -C_i^{\top} \hat{p}_i(t) \qquad \textbf{(4,14)}$$

plus the following equality constraints stated in the formulation of EOP4

$$
\begin{aligned}
f_i(z_i(t),v_i(t),\alpha_i(t)) &= f_i^*(z_i(t),v_i(t),t) - C_i w_i(t) \\
L_i(z_i(t),v_i(t),\gamma_i(t)) &= L_i^*(z_i(t),v_i(t),t)
\end{aligned}
\tag{4,15}
$$

$$
\begin{aligned}
v_i(t) &= u_i(t) \\
z_i(t) &= x_i(t) \\
\hat{p}_i(t) &= p_i(t)
\end{aligned}
\tag{4,16}
$$

$$
w_i(t) = \sum_{j=1}^{N} M_{ij} x_j(t) \tag{4,17}
$$

where $i \in [1,N]$ and $\hat{p}_i(t)$ has been introduced as a costate separation variable.

Definition (4,8) and optimality conditions (4,9), (4,10), (4,11) and (4,12) are satisfied by solving the following modified model-based optimal control problem (MMOP4):

MMOP4

$$
\begin{aligned}
\min_{\substack{u_i(t) \\ i\in[1,N]}} \; J_M &= \sum_{i=1}^{N} J_{Mi} = \sum_{i=1}^{N} \{ \varphi_i(x_i(t_f)) + \int_{t_0}^{t_f} [L_i(x_i,u_i,\gamma_i) \\
&\quad - \lambda_i^T u_i - \beta_i^T x_i - \sum_{j=1}^{N} \Omega_j^T M_{ji} x_i] \, dt \}
\end{aligned}
$$

subject to

$$
\begin{aligned}
\dot{x}_i &= f_i(x_i(t),u_i(t),\alpha_i(t)) \\
x_i(t_0) &= x_{io}
\end{aligned}
$$

under specified $\lambda_i(t)$, $\beta_i(t)$, $\Omega_i(t)$, $\alpha_i(t)$ and $\gamma_i(t)$.

The above analysis gives rise to the following hierarchical (two-level) algorithm, which, assuming convergence, achieves the correct optimal solution of the large-scale ROP4, via repeated solutions of MMOP4.

Algorithm 4.2.1: Hierarchical DISOPE algorithm

Data: f_i, L_i, φ_i, t_0, t_f and means for calculating f_i^*, L_i^*, $i \in [1,N]$.

Step 0: At level 2, compute a nominal solution $u_i^0(t)$, $x_i^0(t)$, $p_i^0(t)$.

Set $\iota=0$, $v_i^0(t)=u_i^0(t)$, $z_i^0(t)=x_i^0(t)$, $\hat{p}_i^0(t)=p_i^0(t)$, $i\in[1,N]$.

Step 1: At level 2, compute $w_i^\iota(t)$, $i\in[1,N]$ from (4,6).

Step 2: At level 2, compute $\alpha_i(t)$, $\gamma_i(t)$, $i\in[1,N]$ from (4,15) and send to level 1.

Step 3: At level 2, compute $\Omega_j(t)$, $j\in[1,N]$ from (4,14) and $\lambda_i(t)$, $\beta_i(t)$, $i\in[1,N]$ from (4,13), and send to level 1.

Step 4: At level 1, under specified $\lambda_i(t)$, $\beta_i(t)$, $\Omega_i(t)$ $\alpha_i(t)$ and $\gamma_i(t)$ solve MMOP4 to obtain $u_i^{\iota+1}(t)$, $x_i^{\iota+1}(t)$, $p_i^{\iota+1}(t)$, $i\in[1,N]$ and send to level 2.

Step 5: At level 2, update the estimate for the optimal solution of ROP4. In order to provide a mechanism for regulating convergence, a relaxation method similar to (2,21) may be employed to satisfy (4,16). If $v^{\iota+1}(t)=v^\iota(t)$ and $w^\iota(t)=Mx^{\iota+1}(t)$ within a defined tolerance, stop, else set $\iota=\iota+1$ and continue from step 1.

4.3 CASE WITH LINEAR-QUADRATIC MODEL-BASED PROBLEM

4.3.1 Formulation

For computational advantage, a linear model-based dynamic function f_i and quadratic weighting functions L_i and φ_i may be chosen. Considering that $\alpha_i(t)$ and $\gamma_i(t)$ enter as shift parameters we have:

$$\begin{aligned} L_i(x_i,u_i,\gamma_i) &= \tfrac{1}{2}x_i(t)^\tau Q_i x_i(t) + \tfrac{1}{2}u_i(t)^\tau R_i u_i(t) + \gamma_i(t) \\ \varphi_i(x_i(t_f)) &= \tfrac{1}{2}x_i(t_f)^\tau \Phi_i x_i(t_f) \\ f_i(x_i,u_i,\alpha_i) &= A_i x_i(t) + B_i u_i(t) + \alpha_i(t) \end{aligned} \qquad (4.18)$$

where $\Phi_i \geq 0$, $Q_i \geq 0$ and $R_i > 0$ are symmetric weighting matrices of the appropriate dimensions, A_i and B_i are matrices which represent a linear model of f_i^*.

Using (4,18) and adding augmentation terms, as was done in Section 2.3.1, equation (4,8) becomes:

$$H_i = \tfrac{1}{2}x_i^T Q_i x_i + \tfrac{1}{2}u_i^T R u_i + \gamma_i + p_i^T(Ax_i + Bu_i + \alpha_i) - \lambda_i^T u - \beta_i^T x \\ + \tfrac{1}{2}r_{i,1}\|u_i - v_i\|^2 + \tfrac{1}{2}r_{i,2}\|x_i - z_i\|^2 - \sum_{j=1}^{N} \Omega_j^T M_{ji} x_i \qquad \textbf{(4,19)}$$

Applying the optimality conditions (4,9), (4,10), (4,11) and (4,12) the following TPBVP is obtained:

$$\dot{x}_i = A_i x_i(t) + B_i \bar{R}_i^{-1}(B_i^T p_i(t) - \bar{\lambda}_i(t)) + \alpha_i(t) \\ \dot{p}_i = -\bar{Q}_i x_i(t) - A_i^T p_i(t) + \bar{\beta}_i(t) \qquad \textbf{(4,20)}$$

with border conditions:

$$x_i(t_0) = x_{io} \\ p_i(t_f) = \Phi_i x_i(t_f) \qquad \textbf{(4,21)}$$

where

$$\bar{R}_i = R_i + r_{i,1} I_{m_i} \\ \bar{Q}_i = Q_i + r_{i,2} I_{n_i} \qquad \textbf{(4,22)}$$

$$\bar{\lambda}_i(t) = \lambda_i(t) + r_{i,1} v_i(t) \\ \bar{\beta}_i(t) = \beta_i(t) + r_{i,2} z_i(t) + \sum_{j=1}^{N} M_{ji}^T \Omega_j(t) \qquad \textbf{(4,23)}$$

It is easy to notice that the structure of the resultant TPBVP (4,21) is identical to that obtained in the centralized case (See equation (2,29)). Therefore, Procedure 2.3.1 may be applied to solve MMOP4 for each subsystem.

The linear-quadratic formulation enables the augmented multipliers $\bar{\lambda}_i(t)$ and $\bar{\beta}_i(t)$ to be written as:

$$\begin{aligned}
\bar{\lambda}_i(t) &= [\frac{\partial f_i}{\partial v_i} - \frac{\partial f_i^*}{\partial v_i}]^T \hat{p}_i(t) + [\bar{R}_i v_i - \nabla_{v_i} L_i^*] \\
\bar{\beta}_i(t) &= [\frac{\partial f_i}{\partial z_i} - \frac{\partial f^*}{\partial z_i}]^T \hat{p}_i(t) + [\bar{Q}_i z_i - \nabla_{z_i} L_i^*] + \sum_{j=1}^{N} M_{ji}^T \Omega_j(t)
\end{aligned} \qquad \textbf{(4,24)}$$

The calculation of parameter $\alpha_i(t)$ becomes, noting that it is not necessary to calculate $\gamma_i(t)$:

$$\alpha_i(t) = f_i^*(z_i(t), v_i(t), t) + C_i w_i - A_i z_i(t) - B_i v_i(t) \qquad \textbf{(4,25)}$$

4.3.2 Hierarchical DISOPE algorithm with LQ model-based problem

The following algorithm requires a nominal solution to start the iterations. A recommended starting point is to use the solution of MMOP4 under $\alpha_i(t) = 0$, $\bar{\lambda}_i(t) = 0$, $\bar{\beta}_i(t) = 0$, $r_{i,1} = r_{i,2} = 0$ and $\Omega_j(t) = 0$, $i \in [1,N]$, $j \in [1,N]$, $t \in [t_0, t_f]$ (relaxed MMOP4).

Algorithm 4.3.2: Hierarchical DISOPE algorithm with LQ model-based problem

---

Data: A_i, B_i, Q_i, ,R_i, Φ_i, t_0, t_f, N, $r_{i,1}$, $r_{i,2}$ and means for calculating f_i^*, L_i^*, $i \in [1,N]$.

Step 0: At level 2, Compute or choose a nominal solution $u_i^0(t)$, $x_i^0(t)$, $p_i^0(t)$. Set $\iota = 0$, $v_i^0(t) = u_i^0(t)$, $z_i^0(t) = x_i^0(t)$, $\hat{p}_i^0(t) = p_i^0(t)$, $i \in [1,N]$.

Step 1: At level 2, compute $w_i(t)$, $i \in [1,N]$ from (4.6).

Step 2: At level 2, compute $\alpha_i(t)$, $i\in[1,N]$ from (4,25) and send to level 1. This is called the *parameter estimation step.*

Step 3: At level 2, compute $\Omega_j(t)$, $j\in[1,N]$ from (4,14) and $\bar{\lambda}_i(t)$, $\bar{\beta}_i(t)$, $i\in[1,N]$, from (4,23) and send to level 1.

Step 4: At level 1, under specified $\bar{\lambda}_i(t)$, $\bar{\beta}_i(t)$, $\Omega_i(t)$, $\alpha_i(t)$, $v_i(t)$, $z_i(t)$ solve MMOP4 to obtain $u_i^{\iota+1}(t)$, $x_i^{\iota+1}(t)$, $p_i^{\iota+1}(t)$, $i\in[1,N]$ and send to level 2. Procedure 2.3.1 may be used to obtain the solution . This is called the *system optimisation step.*

Step 5: At level 2, test convergence and update the estimate for the optimal solution of ROP4. A relaxation method similar to (2,21) may be employed to satisfy (4,16). If $v^{\iota+1}(t)=v^{\iota}(t)$ and $w^{\iota}(t)=Mx^{\iota+1}(t)$ within a defined tolerance, stop, else set $\iota=\iota+1$ and continue from step 1.

The convergence of the algorithm in step 5 can be evaluated by using the following set of 2-norms and comparing each of them with given tolerances ε_v and ε_w, respectively:

Control variation norm:

$$\|v^{\iota+1}-v^{\iota}\|_2=\sqrt{\frac{1}{\Delta t}\int_{t_0}^{t_f}\sum_{j=1}^{N}\|v_j^{\iota+1}(t)-v_j^{\iota}(t)\|^2dt} \qquad \textbf{(4,26)}$$

Interaction norm:

$$\|w^{\iota}-Mx^{\iota+1}\|_2=\sqrt{\frac{1}{\Delta t}\int_{t_0}^{t_f}\sum_{i=1}^{N}\|w_i^{\iota}(t)-\sum_{j=1}^{N}M_{ij}x_j^{\iota+1}\|^2 dt} \qquad \textbf{(4,27)}$$

where Δt is the numerical integration step.

Notice that $\alpha_i(t)$, $\bar{\lambda}_i(t)$ and $\bar{\beta}_i(t)$, $i\in[1,N]$ take into account not only the model-reality differences, but also the influence of the interactions, so that step 3 for the *ith* subsystem is independent from the other subsystems. Furthermore, the algorithm lends itself to parallel processing as the optimisation step for each subsystem can be solved independently.

Figure 4.3.2 shows the information exchange between the two levels.

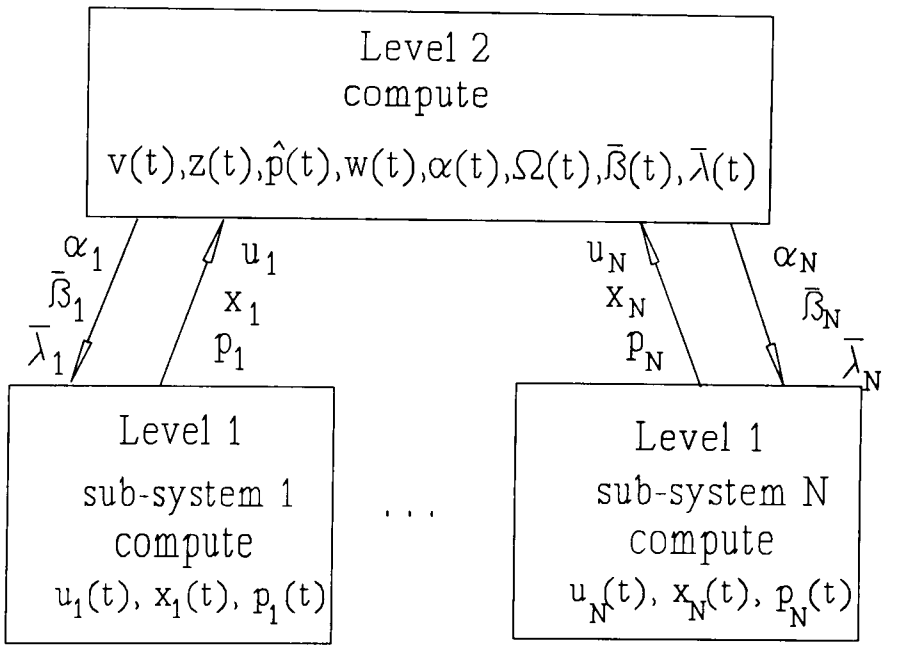

Figure 4.3.2: Information exchange in Algorithm 4.3.2

4.4 SIMULATION EXAMPLE

Algorithm 4.3.2 was implemented in the C++ programming language based on previous work on centralized DISOPE (see Chapter 2). The program implemented solves each subsystem in a sequential way (no parallel processing is used). Only the following overall variables require memory storage at level 2 during the iterations:

$u(t)$, $x(t)$, $p(t)$, $v(t)$, $z(t)$, $\hat{p}(t)$ and $w(t)$. The following simulations were run on a 486DX-based IBM compatible microcomputer with 33 MHz clock speed.

Example 4.4.1: Seventh order nonlinear system

This example consists of the optimal control of a seventh order nonlinear system. The numerical integration step used was $\Delta t = 0.02$ and the tolerances specified for convergence were $\varepsilon_v = \varepsilon_w = 0.05$. Relaxation gains and convexification factors were set to the default values one and zero, respectively.

The overall real optimal control problem is the following seventh order nonlinear problem:

$$\min_{U(t)} \frac{1}{2}\int_0^1 [X^T Q^* X + U^T R^* U]\,dt$$

subject to

$$\begin{aligned}
\dot{X}_1 &= -5X_1 + 0.2X_2 + 0.5X_3 + 0.1X_5 + 0.5X_6 + X_1X_2 + 0.1U_1 \\
\dot{X}_2 &= -2X_2 + 0.5X_4 - 0.5X_5 + 0.2X_6 - 0.1X_7 + X_1^3 + 0.1U_1 \\
\dot{X}_3 &= 0.1X_2 - 1.5X_3 + 0.5X_5 + 0.1X_6 + X_3^2 + X_4^2 + 0.2U_2 \\
\dot{X}_4 &= 0.2X_2 - 0.5X_3 - X_4 + 0.2X_5 + 0.2U_2 \\
\dot{X}_5 &= 0.2X_1 + 0.17X_3 - X_5 + X_7 + X_5X_6 \\
\dot{X}_6 &= 0.1X_1 - 0.2X_2 - X_4 - 0.5X_6 \\
\dot{X}_7 &= 0.4X_1 + 0.1X_2 - X_3 - 0.5X_5 - X_7 + 0.1U_3 \\
X(0) &= [1.0, 0.8, 0.5, 0.6, 1.5, 1, 1, 2]^T
\end{aligned}$$

where

$$\begin{aligned}
Q^* &= diag(1,1,1,1,1,1,1) \\
R^* &= diag(0.1, 0.1, 0.1)
\end{aligned}$$

The overall system was decomposed into three subsystems, giving the following decomposed ROP:

ROP:

$$\min_{\substack{u_i(t) \\ i\in[1,3]}} \frac{1}{2}\sum_{i=1}^{3} \int_0^1 [x_i^{\tau} Q_i^{*} x_i + u_i^{\tau} R_i^{*} u_i]\, dt$$

subject to

subsystem 1:

$$\dot{x}_{1,1} = -5x_{1,1} + 0.2x_{1,2} + x_{1,1}x_{1,2} + 0.1u_{1,1} + w_{1,1}$$

$$\dot{x}_{1,2} = -2x_{1,2} + x_{1,1}^3 + 0.1u_{1,1} + w_{1,2}$$

$$x_1(0) = [1.0, 0.8]^{\tau}$$

subsystem 2:

$$\dot{x}_{2,1} = -1.5x_{2,1} + x_{2,1}^2 + x_{2,2}^2 + 0.2u_{2,1} + w_{2,1}$$

$$\dot{x}_{2,2} = -0.5x_{2,1} - x_{2,2} + 0.2u_{2,1} + w_{2,2}$$

$$x_2(0) = [0.5, 0.6]^{\tau}$$

subsystem 3:

$$\dot{x}_{3,1} = -x_{3,1} + x_{3,3} + x_{3,1}x_{3,2} + w_{3,1}$$

$$\dot{x}_{3,2} = -0.5x_{3,2} + w_{3,2}$$

$$\dot{x}_{3,3} = -0.5x_{3,1} - x_{3,3} + 0.1u_{3,1} + w_{3,3}$$

$$x_3(0) = [1.5, 1.0, 1.2]^{\tau}$$

where

$$Q_1^{*} = diag(1,1);\ R_1^{*} = 0.1$$

$$Q_2^{*} = diag(1,1);\ R_2^{*} = 0.1$$

$$Q_3^{*} = diag(1,1,1);\ R_3^{*} = 0.1$$

$$[x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}, x_{3,1}, x_{3,2}, x_{3,3}] = [X_1, X_2, X_3, X_4, X_5, X_6, X_7]$$

and the interaction vectors are given by:

$$
\begin{aligned}
w_1 &= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} x_2 + \begin{bmatrix} 0.1 & 0.5 & 0 \\ -0.5 & 0.2 & -0.1 \end{bmatrix} x_3 \\
w_2 &= \begin{bmatrix} 0 & 0.1 \\ 0 & 0.2 \end{bmatrix} x_1 + \begin{bmatrix} 0.5 & 0.1 & 0 \\ 0.2 & 0 & 0 \end{bmatrix} x_3 \\
w_3 &= \begin{bmatrix} 0.2 & 0 \\ 0.1 & -0.2 \\ 0.4 & 0.1 \end{bmatrix} x_1 + \begin{bmatrix} 0.17 & 0 \\ 0 & -1 \\ -1 & 0 \end{bmatrix} x_2
\end{aligned}
\qquad \textbf{(4,28)}
$$

The model-based problem is an LQ approximation of ROP.

MOP:

$$
\min_{\substack{u_i(t) \\ i \in [1,3]}} \frac{1}{2} \sum_{i=1}^{3} \int_0^1 [x_i^{\top} Q_i x_i + u_i^{\top} R_i u_i + 2\gamma_i(t)]\, dt
$$

subject to

subsystem 1:

$$
\dot{x}_1 = \begin{bmatrix} -5 & 0.2 \\ 0 & -2 \end{bmatrix} x_1 + \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} u_1 + \alpha_1
$$

$$
x_1(0) = [1.0, 0.8]^{\top}
$$

subsystem 2:

$$
\dot{x}_2 = \begin{bmatrix} -1.5 & 0 \\ -0.5 & -1 \end{bmatrix} x_2 + \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} u_2 + \alpha_2
$$

$$
x_2(0) = [0.5, 0.6]^{\top}
$$

subsystem 3:

$$\dot{x}_3 = \begin{bmatrix} -1 & 0 & 1 \\ 0 & -0.5 & 0 \\ -0.5 & 0 & -1 \end{bmatrix} x_3 + \begin{bmatrix} 0 \\ 0 \\ 0.1 \end{bmatrix} u_3 + \alpha_3$$

$$x_3(0) = [1.5, 1.0, 1.2]^T$$

where

$$Q_1 = diag(1,1);\ R_1 = 0.1$$
$$Q_2 = diag(1,1);\ R_2 = 0.1$$
$$Q_3 = diag(1,1,1);\ R_3 = 0.1$$

For the sake of comparison, ROP was also solved in a centralized way (without decomposition). Table 4.4.1 shows the performance of the algorithm for each case (centralized and hierarchical) and the final performance index J^*.

Case	Number of DISOPE iterations	CPU time (s)	J^*
Centralized	6	134	2.210431
Hierarchical	6	112	2.210553

Table 4.4.1: Performances for example 4.4.1

Figures 4.4.1.1 to 4.4.1.3 show the final state responses for each subsystem. Figures 4.4.1.4 to 4.4.1.6 show the computed optimal control signals for each subsystem. Figures 4.4.1.7 and 4.4.1.8 show the convergence behaviour of the hierarchical algorithm.

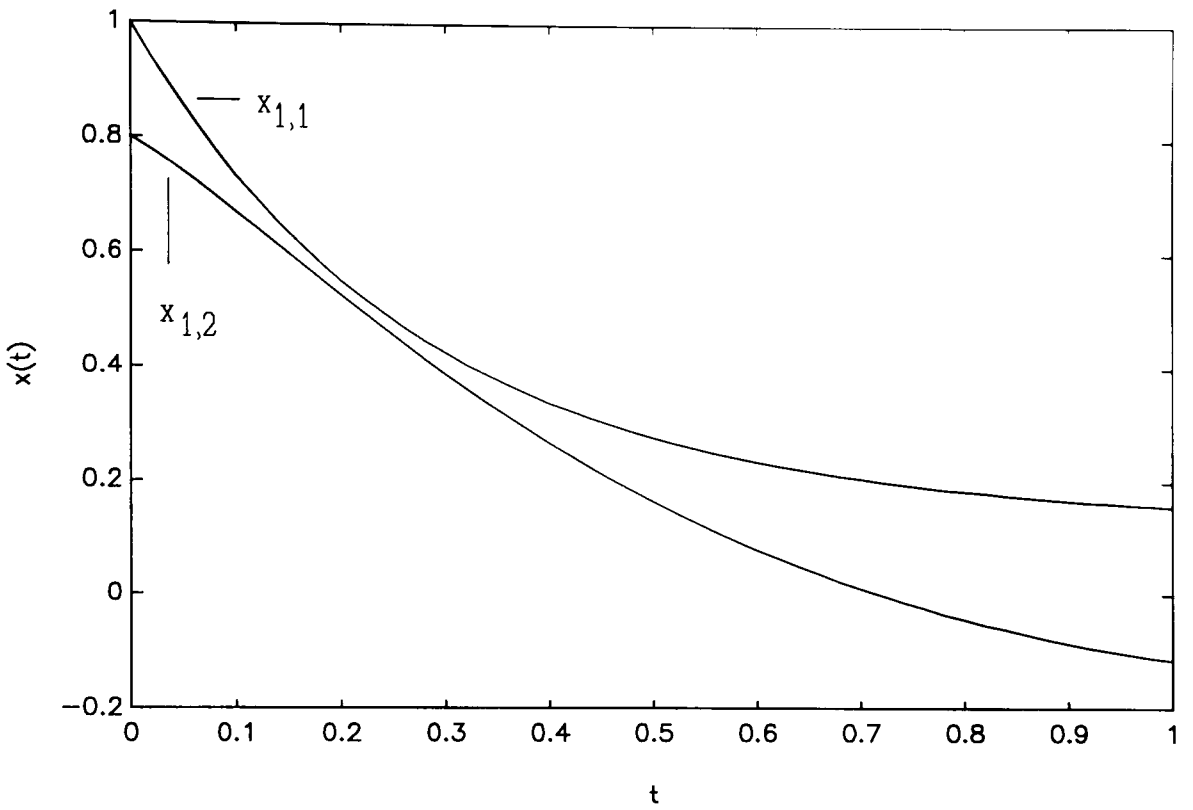

Figure 4.4.1.1: Example 4.4.1, subsystem 1 state vector

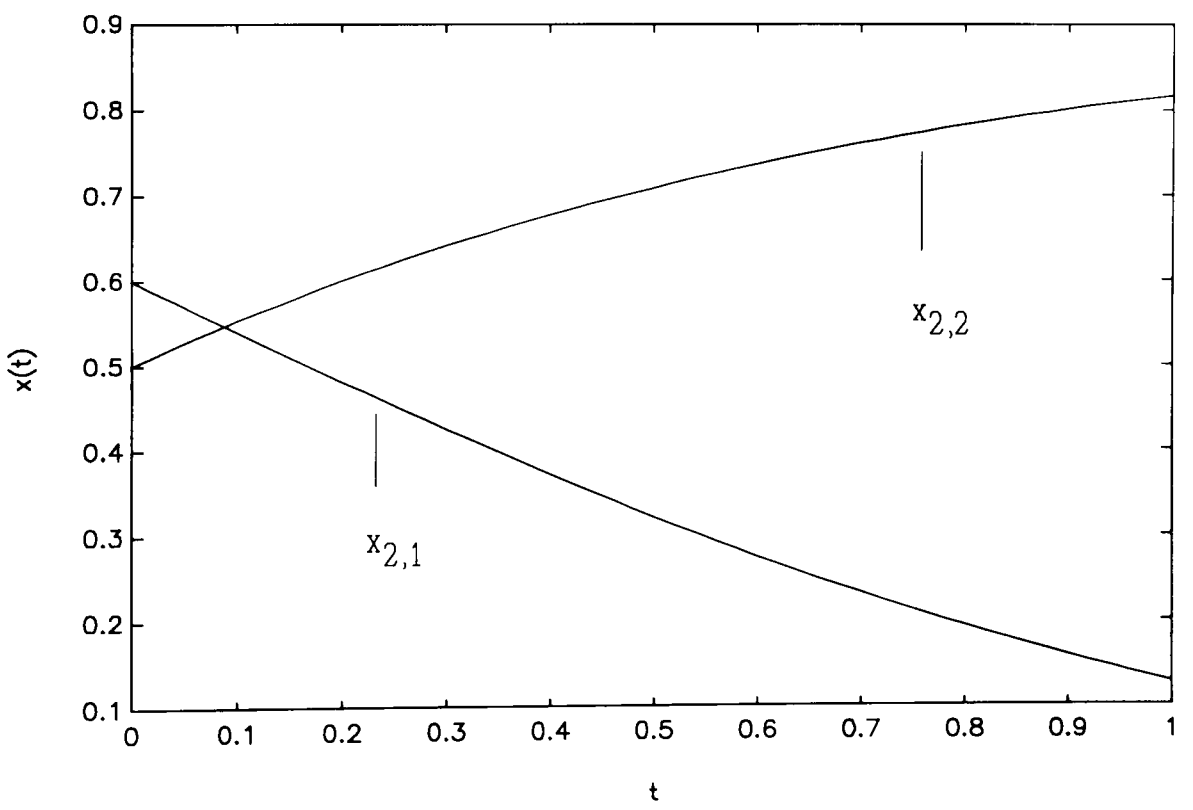

Figure 4.4.1.2: Example 4.4.1, subsystem 2 state vector

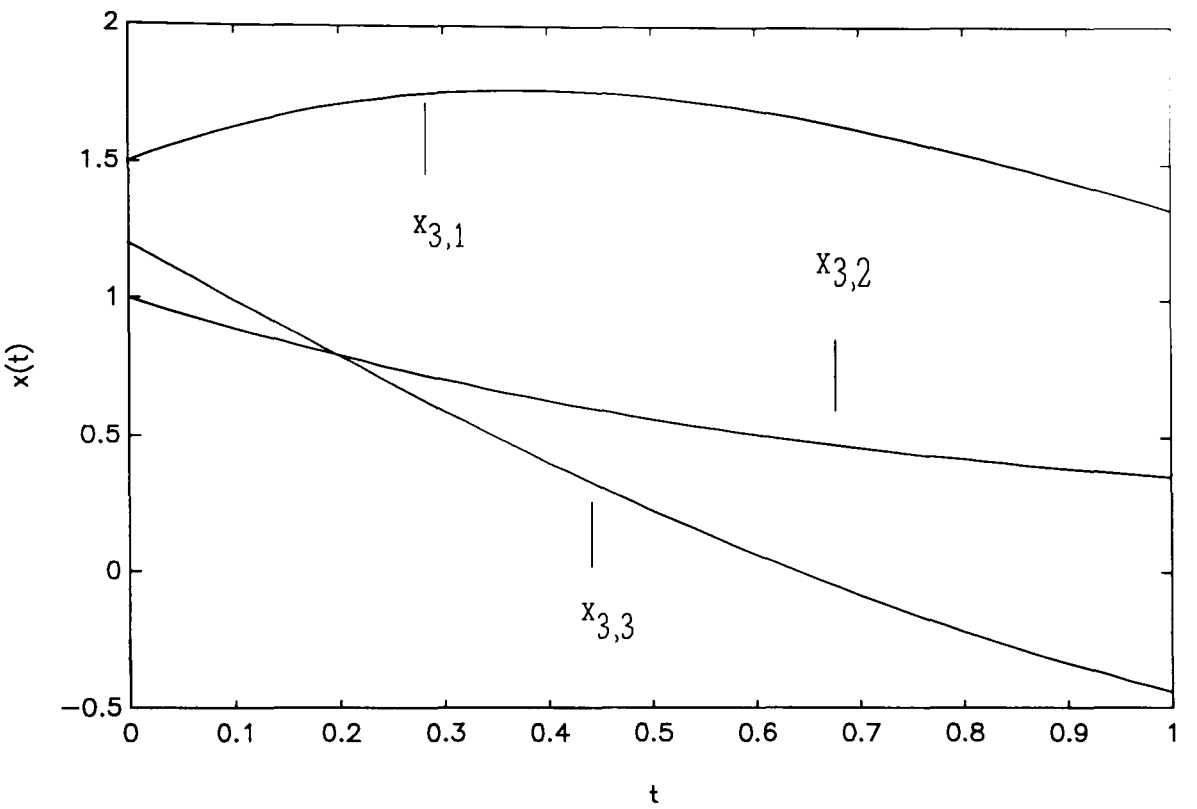

Figure 4.4.1.3: Example 4.4.1, subsystem 3 state vector

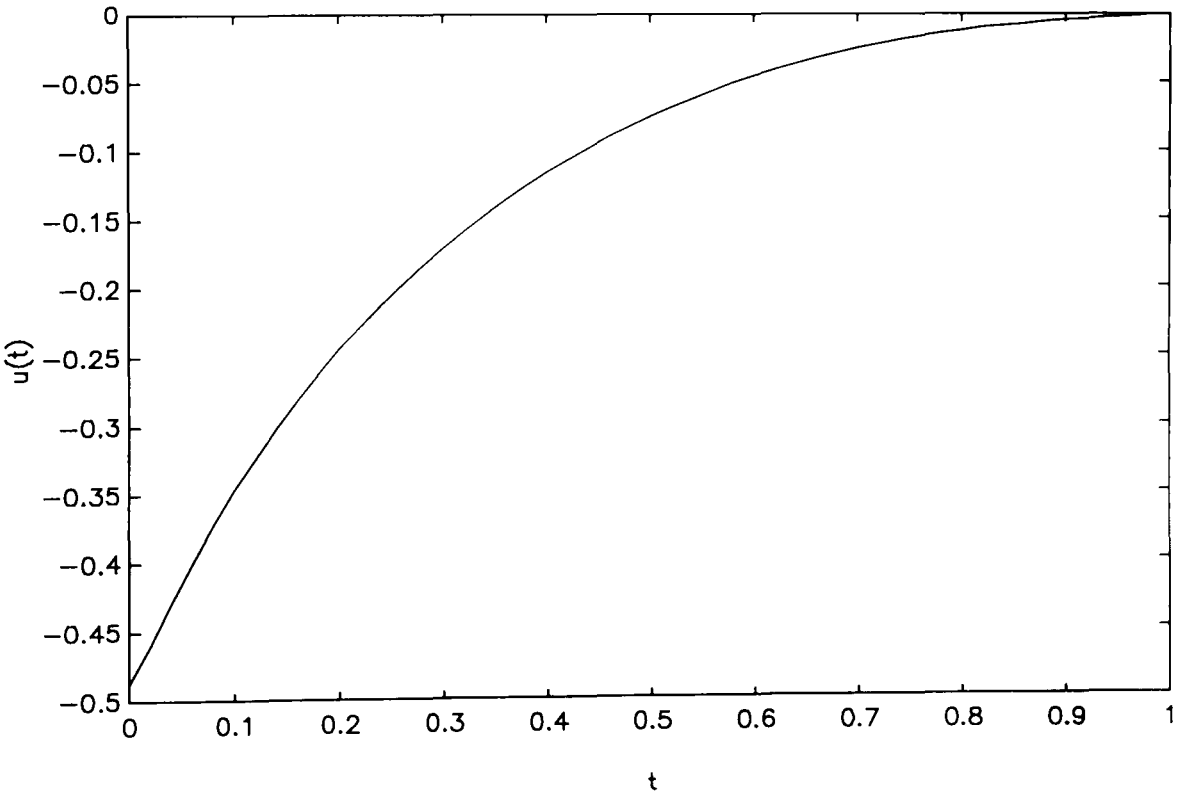

Figure 4.4.1.4: Example 4.4.1, subsystem 1 final control signal

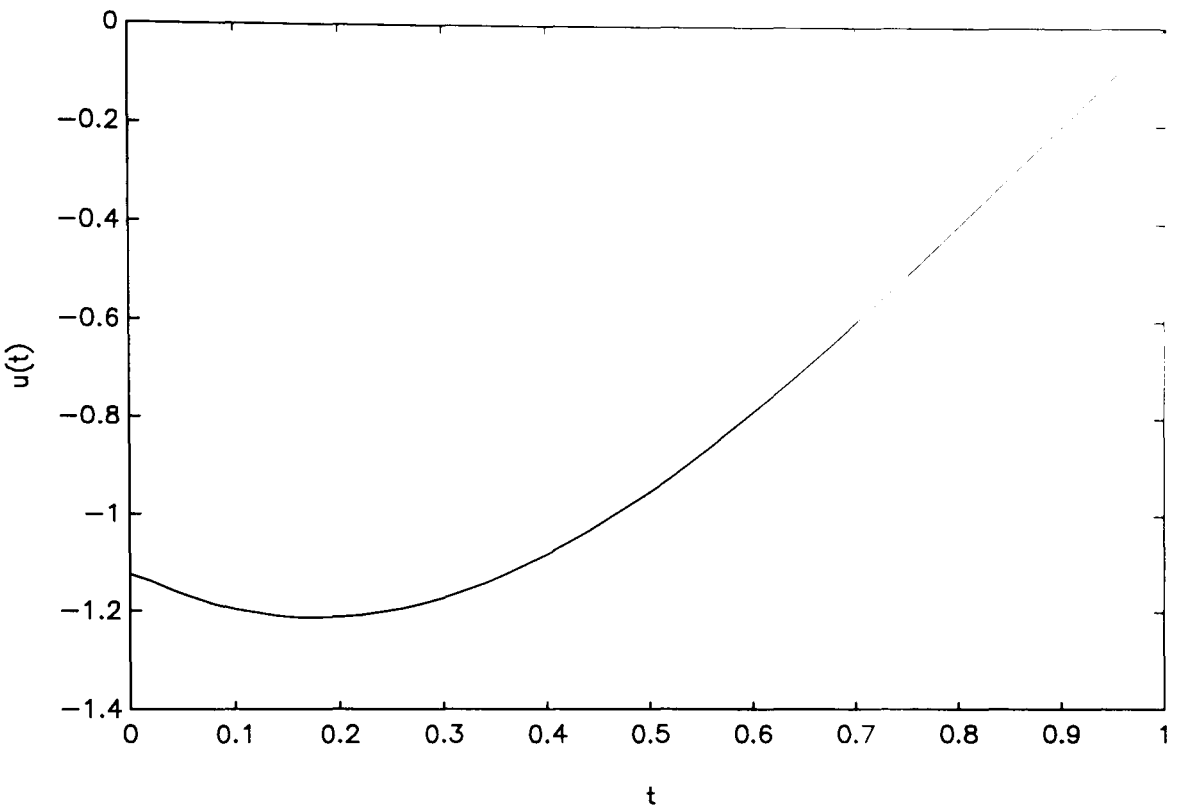

Figure 4.4.1.5: Example 4.4.1, subsystem 2 final control signal

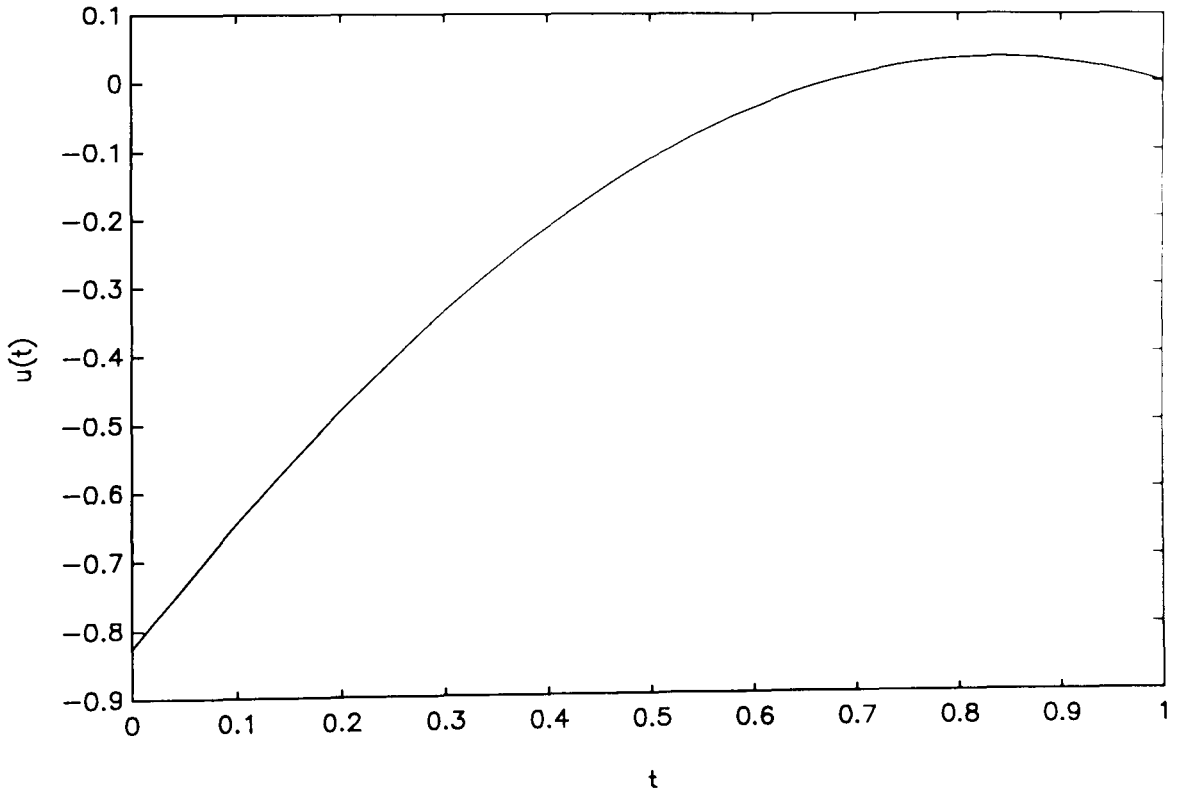

Figure 4.4.1.6: Example 4.4.1, subsystem 3 final control signal

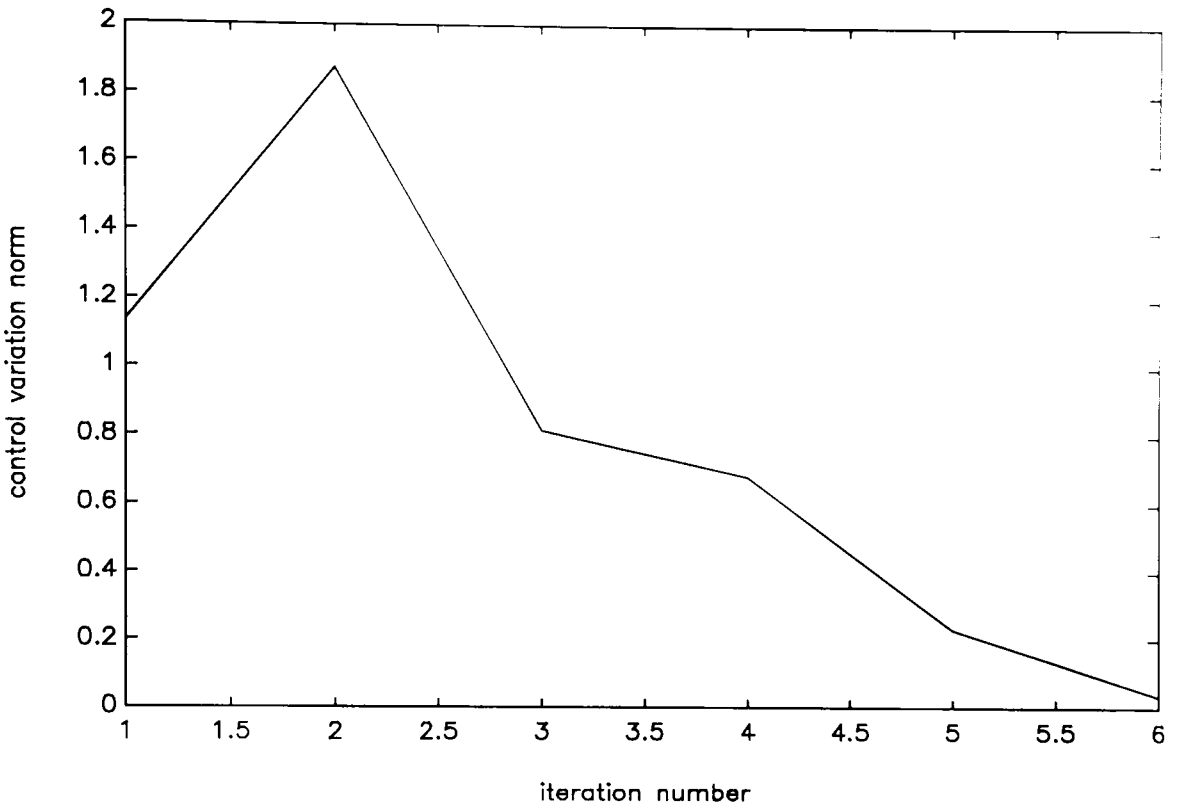

Figure 4.4.1.7: Example 4.4.1, control variation norm vs. iteration (hierarchical case)

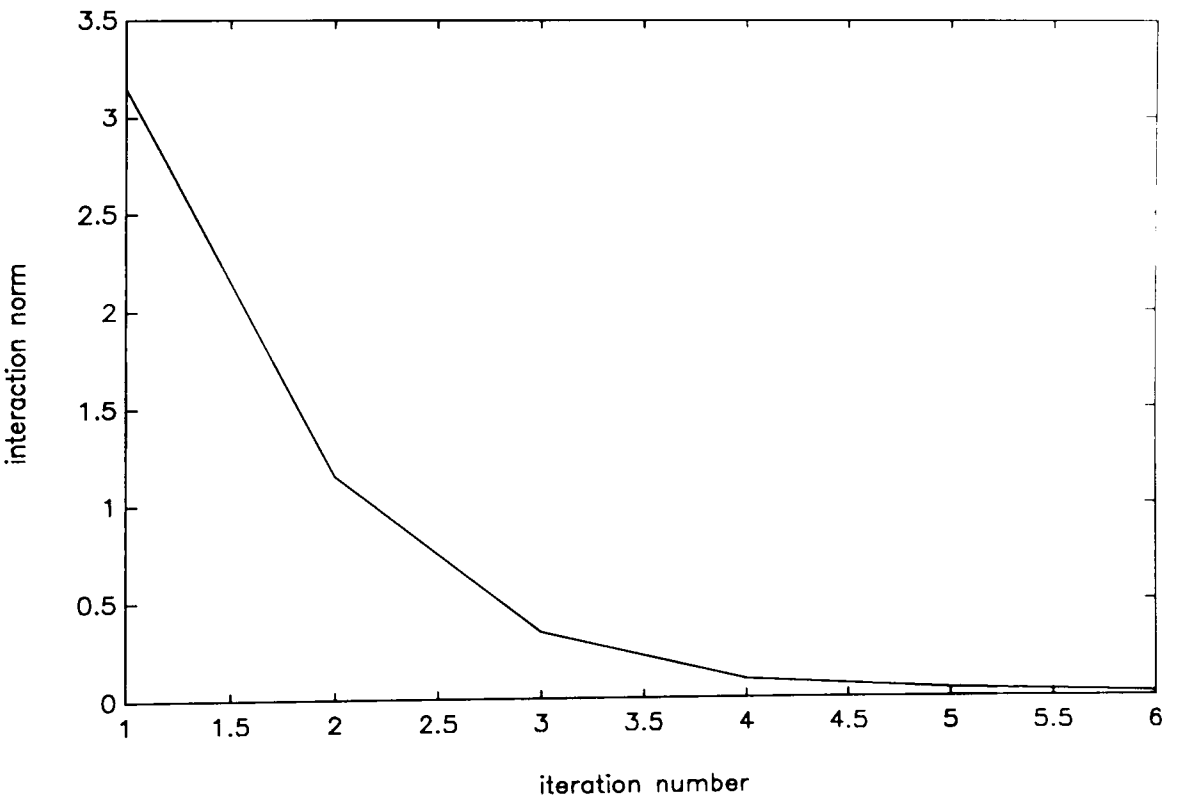

Figure 4.4.1.8: Example 4.4.1, interaction norm vs. iteration (hierarchical case)

It can be seen that the hierarchical algorithm developed and implemented (Algorithm 4.3.2) achieved the same solution than that obtained in the centralized case. Moreover, it can be checked that the solutions obtained satisfy (within the tolerances specified for convergence) the optimality conditions of the overall real optimal control problem. Therefore, the hierarchical algorithm achieved the correct optimal solutions in spite of model reality-differences. Furthermore, the CPU times obtained were lower in the hierarchical case than in the centralized case, which indicates that parallel processing would reduce substantially the computational times.

4.5 SUMMARY

An algorithm for hierarchical optimal control of large-scale systems with model-reality differences has been developed. A version of the hierarchical DISOPE algorithm with linear-quadratic model-based problem was developed and implemented in software. The technique handles large-scale continuous time nonlinear systems with non-quadratic performance indexes. The implemented algorithm was tested with one simulation example. The algorithm achieved the correct optimal solutions in spite of model-reality differences. It is suitable for parallel or distributed processing, as the calculations for each subsystem can be done independently. The algorithm as implemented may yield substantial computational savings in terms of memory storage.

CHAPTER 5

DISCRETE-TIME DISOPE ALGORITHM

In this chapter, an algorithm for solving nonlinear discrete-time optimal control problems with model-reality differences is developed. A version of the algorithm with a linear-quadratic model-based problem is developed and implemented in software. A discretization scheme, which avoids the use of crude approximations when discretizing continuous time dynamic systems, is introduced. The algorithm implemented is tested with three simulation examples. The research work presented in this chapter is also described in (Becerra, 1993b; Becerra and Roberts, 1994a).

5.1 DIGITAL COMPUTER CONTROL

With the increasing sophistication and decreasing cost of microprocessors, more control schemes are being implemented digitally. In these schemes, the control input is switched between different values at discrete-time steps. The control signal is normally held constant between such switchings by a zero-order hold. Such controls are usually designed using a discretized version of the continuous plant. There are also processes which are discrete in nature and can only be controlled by using discrete-time controllers (Åström and Wittenmark, 1990; Leigh, 1992; Franklin *et al*, 1990).

5.2 DISCRETE-TIME DISOPE ALGORITHM

5.2.1 Problem formulation and solution approach

Suppose that the real plant dynamics are described by the following nonlinear time-varying difference equation:

$$x(k-1) = f^*(x(k), u(k), k) \tag{5,1}$$

where $f^* : \Re^n \times \Re^m \times \Re \rightarrow \Re^n$ represents a set of discrete-time state equations which describe the process with state $x(k) \in \Re^n$ and control input $u(k) \in \Re^m$. Further assume that the following performance index has been chosen:

$$J^* = \varphi(x(N_f)) + \sum_{N_0}^{N_f-1} L^*(x(k), u(k), k) \tag{5,2}$$

where $[N_0, N_f]$ is the fixed time interval of interest, $\varphi : \Re^n \rightarrow \Re$ is a scalar valued terminal weighting function and $L^* : \Re^n \times \Re^m \times \Re \rightarrow \Re$ is a discrete performance (or weighting) function.

If the state of the system at the initial time N_0 is assumed known (being measured or estimated), with value $x(N_0) = x_0$, and if no constraints on the values of control and state variables are taken into consideration, apart from the dynamic constraint (5,1), the discrete-time real optimal control problem (ROP5) can be formulated as follows:

ROP5

$$\min_{\substack{u(k) \\ k \in [N_0, N_f-1]}} J^* = \varphi(x(N_f)) + \sum_{k=N_0}^{N_f-1} L^*(x(k), u(k), k)$$

subject to

$$x(k+1) = f^*(x(k), u(k), k)$$
$$x(N_0) = x_0$$

Define the following function (the Hamiltonian):

$$H^*(x(k), u(k), p(k), k) = L^*(x(k), u(k), k) + p(k+1)^T f^*(x(k), u(k), k) \tag{5,3}$$

where $p(k) \in \Re^n$ is a Lagrange multiplier function usually termed as the *costate*.

The necessary optimality conditions of the ROP5 are as follows (see (Lewis, 1986a: 27-30), for the derivation):

Stationarity:

$$\nabla_{u(k)} H^* = 0 \tag{5,4}$$

Costate equation:

$$\nabla_{x(k)} H^* - p(k) = 0 \tag{5,5}$$

State equation:

$$\nabla_{p(k+1)} H^* - x(k+1) = 0 \tag{5,6}$$

Boundary conditions:

$$x(N_0) = x_0$$
$$p(N_f) = \nabla_{x(k)} \varphi(x(k))\big|_{k=N_f} \tag{5,7}$$

Instead of solving ROP5, the following, possibly simplified, discrete-time model-based optimal control problem (MOP5) is considered:

MOP5

$$\min_{\substack{u(k) \\ k\in[N_0,N_f-1]}} \quad J_m = \varphi(x(N_f)) + \sum_{k=N_0}^{N_f-1} L(x(k),\ u(k),\ \gamma(k))$$

subject to

$$x(k+1) = f(x(k),\ u(k),\ \alpha(k))$$
$$x(N_0) = x_0$$

where state and control vectors have the same dimensions as in ROP5, J_m is a model-based performance index, $L : \Re^n \times \Re^m \times \Re \rightarrow \Re$ is a discrete weighting function and perhaps a simplification of a known L^*, $f : \Re^n \times \Re^m \times \Re^r \rightarrow \Re$ is an approximate

dynamic model of f^*, $\gamma(k)\in\Re$ and $\alpha(k)\in\Re^r$ are discrete parameters. The role of $\gamma(k)$ and $\alpha(k)$ will be to take into account model reality-differences *in value*.

Following a similar reasoning to that used in Section 2.2.1, we now integrate system optimisation and parameter estimation by defining an expanded optimal control problem (EOP5) which, in spite of being model-based, is made equivalent to ROP5 by adding appropriate equality constraints on state equations and discrete weighting function values. Furthermore, state and control variables are separated between parameter estimation and optimisation steps by introducing the new state and control variables $z(k)$ and $v(k)$, respectively.

EOP5

$$\min_{\substack{u(k)\\ k\in[N_0,N_f-1]}} \quad J_e = \varphi(x(N_f)) + \sum_{k=N_0}^{N_f-1} L(x(k),\ u(k),\ \gamma(k))$$

subject to

$$x(k+1) = f(x(k),\ u(k),\ \alpha(k))$$
$$x(N_0) = x_0$$

$$f(z(k),v(k),\alpha(k)) = f^*(z(k),v(k),k)$$
$$L(z(k),v(k),\gamma(k)) = L^*(z(k),v(k),k)$$

$$u(k) = v(k)$$
$$x(k) = z(k)$$

Adjoining all the equality constraints to the performance index by using Lagrange multipliers, we obtain the following augmented performance index J_e':

$$J_e' = \varphi(x(N_f)) + \sum_{k=N_0}^{N_f-1} [\ L(x(k),u(k),\gamma(k)) + p(k+1)^T(f(x(k),u(k),\alpha(k)) - x(k+1))$$
$$+ \lambda(k)^T(v(k)-u(k)) + \beta(k)^T(z(k)-x(k)) + \mu(k)^T(f^*(z(k),v(k),k) - f(z(k),v(k),\alpha(k)))$$
$$+ \xi(k)(L^*(z(k),v(k),k) - L(z(k),v(k),\gamma(k)))\ ] \qquad (5.8)$$

where $p(k)\in\Re^n$, $\lambda(k)\in\Re^m$, $\beta(k)\in\Re^n$, $\mu(k)\in\Re^m$ and $\xi(k)\in\Re$ are Lagrange multipliers.

Define the augmented Hamiltonian function as:

$$H = L(x(k),u(k),\gamma(k)) + p^{\top}(k+1)f(x(k),u(k),\alpha(k)) - \lambda(k)^{\top}u(k) - \beta(k)^{\top}x(k) \qquad \textbf{(5,9)}$$

then we can re-write (5,8) as

$$\begin{aligned} J_e' = [\varphi(x(N_f))] + \sum_{k=N_0}^{N_f-1} [\, H - p(k+1)^{\top}x(k+1) \\ + \lambda(k)^{\top}v(k) + \beta(k)^{\top}z(k) + \mu(k)^{\top}(f^*(z(k),v(k),k) - f(z(k),v(k),\alpha(k))) \\ + \xi(k)(L^*(z(k),v(k),k) - L(z(k),v(k),\gamma(k))) \,] \end{aligned} \qquad \textbf{(5,10)}$$

Now, it is desired to examine the increment in J_e' due to increments in all the variables. It is assumed that the final time N_f is fixed. This increment is given by:

$$\begin{aligned} dJ_e' = \nabla_{x(N_f)}\varphi(x(N_f))^{\top}dx(N_f) + \sum_{k=N_0}^{N_f-1} \{\nabla_{u(k)}H^{\top}du(k) + [\nabla_{x(k)}H - p(k)]^{\top}dx(k) \\ + [\,\lambda(k)^{\top} + \mu(k)^{\top}(f^*_{v(k)} - f_{v(k)}) + \xi(k)(\nabla_{v(k)}L^* - \nabla_{v(k)}L)]^{\top}dv(k) \\ + [\beta(k)^{\top} + \mu(k)^{\top}(f^*_{z(k)} - f_{z(k)}) + \xi(k)(\nabla_{z(k)}L^* - \nabla_{z(k)}L)]^{\top}dz(k) \\ + [\nabla_{\alpha(k)}H^{\top} - \mu(k)^{\top}f_{\alpha(k)}]d\alpha(k) + [\nabla_{\gamma(k)}H - \xi(k)\nabla_{\gamma(k)}L]^{\top}d\gamma(k) \\ + [\nabla_{p(k+1)}H - x(k+1)]^{\top}dp(k+1) + [\nabla_{\lambda(k)}H + v(k)]^{\top}d\lambda(k) \\ + [\nabla_{\beta(k)}H + z(k)]^{\top}d\beta(k) + [f^* - f]^{\top}d\mu(k) + [L^* - L]d\xi(k) \,\} \end{aligned} \qquad \textbf{(5,11)}$$

According to the Lagrange multiplier theory, at a constrained minimum this increment dJ_e' should be zero. Setting to zero the coefficients of the independent increments in (5,11), and concluding by inspection that $\xi(k) = 1$ and $\mu(k) = p(k+1)$, $k\in[N_0,N_f-1]$ the following necessary optimality conditions are obtained:

Stationarity:

$$\nabla_{u(k)} H = 0 \tag{5,12}$$

Costate equation:

$$\nabla_{x(k)} H - p(k) = 0 \tag{5,13}$$

State equation:

$$\nabla_{p(k+1)} H - x(k+1) = 0 \tag{5,14}$$

Boundary conditions:

$$\begin{aligned} x(N_0) &= x_0 \\ p(N_f) &= \nabla_{x(k)} \varphi(x(k))\big|_{k=N_f} \end{aligned} \tag{5,15}$$

Multiplier equations:

$$\begin{aligned} \lambda(k) &= [\frac{\partial f}{\partial v(k)} - \frac{\partial f^*}{\partial v(k)}]^T \hat{p}(k+1) + [\nabla_{v(k)} L - \nabla_{v(k)} L^*] \\ \beta(k) &= [\frac{\partial f}{\partial z(k)} - \frac{\partial f^*}{\partial z(k)}]^T \hat{p}(k+1) + [\nabla_{z(k)} L - \nabla_{z(k)} L^*] \end{aligned} \tag{5,16}$$

plus the following equality constraints

$$\begin{aligned} f(z(k), v(k), \alpha(k)) &= f^*(z(k), v(k), k) \\ L(z(k), v(k), \gamma(k)) &= L^*(z(k), v(k), k) \end{aligned} \tag{5,17}$$

$$\begin{aligned} v(k) &= u(k) \\ z(k) &= x(k) \\ \hat{p}(k) &= p(k) \end{aligned} \tag{5,18}$$

where $k \in [N_0, N_f - 1]$, and $\hat{p}(k)$ has been introduced as a costate separation variable. We assume that the structure of f and L is such that given $v(k)$ and $z(k)$, $k \in [N_0, N_f - 1]$ the values of $\alpha(k)$ and $\gamma(k)$, $k \in [N_0, N_f - 1]$ can be uniquely determined from (5,17). Notice that optimality conditions (5,12), (5,13) and (5,14)

are model-based, and that $\lambda(k)$ and $\beta(k)$ $k\in[N_0,N_f-1]$ carry information on model-reality differences, *in curvature*, as opposed to $\alpha(k)$ and $\gamma(k)$ $k\in[N_0,N_f-1]$ which carry information on model-reality differences *in value*. Following a similar reasoning to that used in Section 2.2.2, we conclude that given values of $\alpha(k)$, $\gamma(k)$, $\lambda(k)$ and $\beta(k)$, $k\in[N_0,N_f-1]$, the solution of the following problem satisfies the definition of the augmented Hamiltonian (5,9) and optimality conditions (5,12), (5,13), (5,14) and (5,15). We call it discrete-time modified model-based problem (MMOP5) defined as follows:

MMOP5

$$\min_{\substack{u(k)\\ k\in[N_0,N_f-1]}} \quad J_M = \varphi(x(N_f)) + \sum_{k=N_0}^{N_f-1} [\, L(x(k),u(k),\gamma(k)) - \lambda(k)^T u(k) - \beta(k)^T x(k) \,]$$

subject to

$$x(k+1) = f(x(k),\ u(k),\ \alpha(k))$$

$$x(N_0) = x_0$$

The above analysis gives rise to the following discrete-time DISOPE algorithm which, assuming convergence, achieves the solution of ROP5 via repeated solutions of MMOP5 (Becerra, 1993b).

Algorithm 5.2.1: Discrete-time DISOPE algorithm

Data — f, L, φ, x_0, N_0, N_f, and means for calculating f^* and L^*.

Step 0 — Compute or choose a nominal solution $u^0(k)$, $x^0(k)$ and $p^0(k)$. Set $i=0$, $v^0(k)=u^0(k)$, $z^0(k)=x^0(k)$, $\hat{p}^0(k)=p^0(k)$.

Step 1 — Compute the parameters $\alpha^i(k)$, $\gamma^i(k)$ to satisfy (5,17). This is called the *parameter estimation step.*

Step 2 — Compute the multipliers $\lambda^i(k)$ and $\beta^i(k)$ from (5,16).

Step 3 With specified $\alpha(k)$, $\gamma(k)$, $\lambda(k)$ and $\beta(k)$ solve the discrete-time modified model-based optimal control problem MMOP5 to obtain $u^{i+1}(k)$, $x^{i+1}(k)$ and $p^{i+1}(k)$. This is called the *system optimisation step*.

Step 4 This step tests convergence and updates the estimate for the optimal solution of ROP5. In order to provide a mechanism for regulating convergence, a simple relaxation method is employed to satisfy (5,18). This is:

$$\begin{aligned} v^{i+1}(k) &= v^{i}(k) + k_v(u^{i+1}(k) - v^{i}(k)) \\ z^{i+1}(k) &= z^{i}(k) + k_z(x^{i+1}(k) - z^{i}(k)) \\ \hat{p}^{i+1}(k) &= \hat{p}^{i}(k) + k_p(p^{i+1}(k) - \hat{p}^{i}(k)) \end{aligned} \qquad \textbf{(5,19)}$$

where k_v, k_z and $k_p \in (0,1]$ are scalar gains. If $v^{i+1}(k) = v^{i}(k)$, $k \in [N_0, N_f-1]$ within a given tolerance stop, else set $i=i+1$ and continue from step 1.

5.2.2 Performance index augmentation

Variable augmentation has been used in continuous time DISOPE (see Section 2.2.2) and has resulted in improved algorithm stability and convergence in difficult cases (Roberts, 1993a). Augmentation may also be applied in discrete-time DISOPE by adding convexification terms to the performance index of EOP5. This is, J_e becomes:

$$\begin{aligned} J_e = \varphi(x(N_f)) + \sum_{k=N_0}^{N_f-1} [\, & L(x(k), u(k), \gamma(k)) \\ & + \tfrac{1}{2} r_1 \|u(k)-v(k)\|^2 + \tfrac{1}{2} r_2 \|x(k)-z(k)\|^2 \,] \end{aligned} \qquad \textbf{(5,20)}$$

where r_1 and r_2 are given scalar convexification factors.

The definition of the augmented Hamiltonian is changed to take into account the new terms:

$$H = L(x(k),u(k),\gamma(k)) + p(k+1)^{\top} f(x(k),u(k),\alpha(k)) - \lambda(k)^{\top} u(k) - \beta(k)^{\top} x(k) + \frac{1}{2} r_1 \|u(k)-v(k)\|^2 + \frac{1}{2} r_2 \|x(k)-z(k)\|^2 \quad \textbf{(5,21)}$$

By using a similar analysis to that used in Section 5.2.1 it is possible to find that the model-based optimality conditions obtained in Section 5.2.1 continue to be valid. Furthermore, the main change in Algorithm 5.2.1 is that the solution of MMOP5 requires information on $v(k)$ and $z(k)$, since the performance index in MMOP5 becomes:

$$J_M = \varphi(x(N_f)) + \sum_{N_0}^{N_f-1} [\, L(x(k),\, u(k),\, \gamma(k)) - \lambda(k)^{\top} u(k) - \beta(k)^{\top} x(k) + \frac{1}{2} r_1 \|u(k)-v(k)\|^2 + \frac{1}{2} r_2 \|x(k)-z(k)\|^2 \,] \quad \textbf{(5,22)}$$

5.2.3 Terminal state constraints

In some problems we are interested in restricting functions of the terminal state to have prescribed values, that is:

$$\psi(x(N_f),N_f) = 0 \quad \textbf{(5,23)}$$

for a given function $\psi : \Re^n \times \Re \rightarrow \Re^q$. Equation (5,23) is an additional equality constraint to ROP5, MOP5, EOP5 and MMOP5. It can be treated by using the Lagrange multiplier theory (see, for example, Bryson and Ho, 1975). As a consequence, a new boundary condition for the costate is obtained as follows:

$$p(N_f) = \left[\nabla_{x(k)} \varphi + \frac{\partial \psi^{\top}}{\partial x(k)} \nu \right]_{k=N_f} \quad \textbf{(5,24)}$$

where $\nu \in \Re^q$ is a Lagrange multiplier vector to be found so that the additional necessary condition (5,23) is satisfied. Notice that in terminal state constrained problems some reachability conditions must be satisfied for a solution to exist (see, for example, Lewis (1986a) for a discussion on the LQ case).

5.3 CASE WITH LINEAR QUADRATIC MODEL-BASED PROBLEM

5.3.1 Formulation

If the model-based problem (MOP5) is chosen as a linear-quadratic approximation of ROP5, then noniterative methods (Lewis, 1986a) can be used for the model-based computations. As was done in Chapter 2, for computational advantage, a linear model-based dynamic function f and quadratic weighting functions L and φ may be chosen. Considering that $\alpha(k)$ and $\gamma(k)$ enter as shift parameters we have:

$$\begin{aligned} L(x(k),u(k),\gamma(k)) &= \tfrac{1}{2}x(k)^{\top}Qx(k) + \tfrac{1}{2}u(k)^{\top}Ru(k) + \gamma(k) \\ \varphi(x(N_f)) &= \tfrac{1}{2}x(N_f)^{\top}\Phi x(N_f) \\ f(x(k),u(k),\alpha(k)) &= Ax(k) + Bu(k) + \alpha(k) \end{aligned} \qquad \textbf{(5,25)}$$

where $\Phi \geq 0$, $Q \geq 0$ and $R > 0$ are symmetric weighting matrices of the appropriate dimensions, A and B are matrices which represent a linear model of f^*.

By using (5,25), and including the variable augmentation discussed in Section 5.2.2 the linear-quadratic MMOP5 can be written as:

$$\min_{\substack{u(k) \\ k\in[N_0,N_f-1]}} \quad J_M = \tfrac{1}{2}x(N_f)^{\top}\Phi x(N_f) + \sum_{k=N_0}^{N_f-1} [\ \tfrac{1}{2}x(k)^{\top}Qx(k) + \tfrac{1}{2}u(k)^{\top}Ru(k) + \gamma(k)$$
$$- \lambda(k)^{\top}u(k) - \beta(k)^{\top}x(k) + \tfrac{1}{2}r_1\|u(k)-v(k)\|^2 + \tfrac{1}{2}r_2\|x(k)-z(k)\|^2\ ]$$

subject to

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + \alpha(k) \\ x(N_0) &= x_0 \end{aligned}$$

The corresponding augmented Hamiltonian function is:

$$\begin{aligned} H &= \tfrac{1}{2}(x(k)^{\top}Qx(k) + u(k)^{\top}Ru(k)) + \gamma(k) \\ &+ p(k+1)^{\top}(Ax(k) + Bu(k) + \alpha(k)) - \lambda(k)^{\top}u(k) - \beta(k)^{\top}x(k) \\ &+ \tfrac{1}{2}r_1\|u(k)-v(k)\|^2 + \tfrac{1}{2}r_2\|x(k)-z(k)\|^2 \end{aligned} \qquad \textbf{(5,26)}$$

Applying the model-based optimality conditions (5,12), (5,13), (5,14) and (5,15) with H given by (5,26), we obtain the control law:

$$u(k) = -\bar{R}^{-1}(B^{\top}p(k+1) - \bar{\lambda}(k)) \qquad \textbf{(5,27)}$$

and, in addition, the following TPBVP:

$$\begin{aligned} x(k+1) &= Ax(k) - B\bar{R}^{-1}(B^{\top}p(k+1) - \bar{\lambda}(k)) + \alpha(k) \\ p(k) &= \bar{Q}x(k) + A^{\top}p(k+1) - \bar{\beta}(k) \end{aligned} \qquad \textbf{(5,28)}$$

with border conditions:

$$\begin{aligned} x(N_0) &= x_0 \\ p(N_f) &= \Phi x(N_f) \end{aligned} \qquad \textbf{(5,29)}$$

where $\bar{R} = R + r_1 I_m$ and $\bar{Q} = Q + r_2 I_n$ are augmented weighting matrices and $\bar{\lambda}(k) = \lambda(k) + r_1 \nu(k)$, and $\bar{\beta}(k) = \beta(k) + r_2 z(k)$ are augmented multipliers.

The linear TPBVP (5,28) can be solved by using the backward sweep method (Bryson and Ho, 1975; Lewis, 1986a). The key is to assume the relationship between costate and state as $p(k) = S(k)x(k) + h(k)$, where $S(k)$ is a nxn matrix and $h(k) \in \Re^n$. This gives rise to the following noniterative solution procedure (see Appendix D for the derivation):

Procedure 5.3.1: Solution of discrete-time linear-quadratic MMOP5

Step a: Solve backwards from $k = N_f - 1$ to N_0 the following *difference* equations, with terminal conditions $S(N_f) = \Phi$ and $h(N_f) = 0$:

$$\begin{aligned} S(k) &= \bar{Q} + A^{\top}S(k+1)(A - BG(k)) \\ G(k) &= [\bar{R} + B^{\top}S(k+1)B]^{-1}B^{\top}S(k+1)A \\ h(k) &= (A - BG(k))^{\top}h(k+1) + (A - BG(k))^{\top}S(k+1)\alpha(k) \\ &\qquad - \bar{\beta}(k) + G(k)^{\top}\bar{\lambda}(k) \end{aligned}$$

Step b: Compute the driving input $g(k)$, $k \in [N_0, N_f - 1]$ from:

$$g(k) = [\bar{R} + B^{\top}S(k+1)B]^{-1}[-B^{\top}S(k+1)\alpha(k) - B^{\top}h(k+1) + \bar{\lambda}(k)]$$

Step c: Compute the state $x(k)$, $k\in[N_0,N_f]$ by solving from the initial condition $x(N_0) = x_0$ the following difference equation:

$$x(k+1) = (A - BG(k))x(k) + Bg(k) + \alpha(k)$$

Step d: Compute the costate $p(k)$, $k\in[N_0,N_f]$ from:

$$p(k) = S(k)x(k) + h(k)$$

Step e: Compute the control input $u(k)$, $k\in[N_0,N_f-1]$ from the control law:

$$u(k) = -G(k)x(k) + g(k)$$

The linear-quadratic formulation enables the augmented multipliers $\bar{\lambda}(k)$ and $\bar{\beta}(k)$, $k\in[N_0,N_f-1]$ to be expressed as (see equation (5,16)):

$$\begin{aligned}\bar{\lambda}(k) &= [\frac{\partial f}{\partial v(k)} - \frac{\partial f^*}{\partial v(k)}]^T \hat{p}(k+1) + [\bar{R}v(k) - \nabla_{v(k)} L^*] \\ \bar{\beta}(k) &= [\frac{\partial f}{\partial z(k)} - \frac{\partial f^*}{\partial z(k)}]^T \hat{p}(k+1) + [\bar{Q}z(k) - \nabla_{z(k)} L^*]\end{aligned} \qquad \textbf{(5,30)}$$

while the calculation of parameter $\alpha(k)$, $k\in[N_0,N_f-1]$ becomes (see equation (5,17)), noting that it is not necessary to calculate $\gamma(k)$:

$$\alpha(k) = f^*(z(k),v(k),k) - Az(k) - Bv(k) \qquad \textbf{(5,31)}$$

5.3.2 Terminal state constraints

Terminal state constraints of the type $x_i(N_f)=0$, $i\in[1,q]$ will be taken into consideration. This kind of constraint can be written as:

$$\psi(x(N_f),N_f)) = Cx(N_f) = 0 \qquad \textbf{(5,32)}$$

where $C = [I_q|0]$ is a qxn matrix. Notice that a terminal constraint of the type $x_i(N_f)=x_{if}$, $i\in[1,q]$ can be achieved by a straightforward shift change of state

variable. The resulting TPBVP is identical to (5,28), but with boundary conditions (see equation (5,24)):

$$\begin{aligned} x(N_0) &= x_0 \\ p(N_f) &= \Phi x(N_f) + C^{\top}\nu \end{aligned} \tag{5.33}$$

where $\nu \in \Re^q$ is a Lagrange multiplier to be determined such that (5,32) is satisfied. The solution of MMOP5 taking into account the terminal state constraint is again based on the backward sweep method (Bryson and Ho, 1975). The key is to assume the relationship between costate and state as $p(k) = S(k)x(k) + h(k) + F(k)\nu$, and express the terminal state function as $\psi = F(k)^{\top}x(k) + W(k)\nu + \eta(k) = Cx(N_f)$, where $S(k)$ is a nxn matrix, $F(k)$ is a nxq matrix, $W(k)$ is a qxq matrix, $h(k) \in \Re^n$, and $\eta(k) \in \Re^q$. The resulting noniterative solution procedure is as follows (see Appendix E for the derivation):

Procedure 5.3.2: Solution of LQ terminal constrained discrete-time MMOP5

Step a: Solve backwards from $k = N_f - 1$ to N_0 the following *difference* equations, with terminal conditions $S(N_f) = \Phi$ and $h(N_f) = 0$:

$$\begin{aligned} S(k) &= \bar{Q} + A^{\top}S(k+1)(A - BG(k)) \\ G(k) &= [\bar{R} + B^{\top}S(k+1)B]^{-1}B^{\top}S(k+1)A \\ h(k) &= (A - BG(k))^{\top}h(k+1) + (A - BG(k))^{\top}S(k+1)\alpha(k) \\ &\qquad - \bar{\beta}(k) + G(k)^{\top}\bar{\lambda}(k) \end{aligned}$$

Step b: Solve backwards from $k = N_f - 1$ to N_0 the following difference equations, with terminal conditions $F(N_f) = [0 \; I_q]^{\top}$, $W(N_f) = 0$, and $\eta(N_f) = 0$:

$$F(k) = (A - BG(k))^{\top} F(k+1)$$

$$W(k) = W(k+1) - F(k+1)^{\top}[I_n + B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}B\bar{R}^{-1}B^{\top}F(k+1)$$

$$\eta(k) = \eta(k+1) + F(k+1)^{\top}[I_n + B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}[B\bar{R}^{-1}\bar{\lambda}(k) - B\bar{R}^{-1}B^{\top}h(k+1) + \alpha(k)]$$

Step c: Compute the multiplier ν and driving input $g(k) \in \Re^m$, $k \in [N_0, N_f - 1]$ from:

$$\nu = -W(N_0)^{-1}(F(N_0)^{\top}x_0 - \eta(N_0))$$

$$g(k) = [\bar{R} + B^{\top}S(k+1)B]^{-1}[-B^{\top}S(k+1)\alpha(k) - B^{\top}F(k+1)\nu - B^{\top}h(k+1) + \bar{\lambda}(k)]$$

Step d: Compute the state $x(k)$, $k \in [N_0, N_f]$ by solving from the initial condition $x(N_0) = x_0$ the following difference equation:

$$x(k+1) = (A - BG(k))x(k) + Bg(k) + \alpha(k)$$

Step e: Compute the costate $p(k)$, $k \in [N_0, N_f]$ from:

$$p(k) = S(k)x(k) + F(k)\nu + h(k)$$

Step f: Compute the control input $u(k)$, $k \in [N_0, N_f - 1]$ from the control law:

$$u(k) = -G(k)x(k) + g(k)$$

5.3.3 Discrete-time DISOPE algorithm with LQ model-based problem

DISOPE requires a nominal solution to start the iterations. A recommended starting point is to use the solution of MMOP5 under $\alpha(k) = 0$, $\bar{\lambda}(k) = 0$, $\bar{\beta}(k) = 0$, $k \in [N_0, N_f - 1]$, $r_1 = r_2 = 0$ (relaxed MMOP5).

From the above analysis, the following discrete-time DISOPE algorithm with linear-quadratic model-based problem has been proposed (Becerra, 1993b).

Algorithm 5.3.3: Discrete-time DISOPE algorithm with LQ model-based problem

---

Data — A, B, Q, R, Φ, r_1, r_2, k_v, k_z, k_p, x_0, N_0, N_f, q and means for calculating f^* and L^*.

Step 0 — Compute or choose a nominal solution $u^0(k)$, $x^0(k)$ and $p^0(k)$. Set $i=0$, $v^0(k)=u^0(k)$, $k\in[N_0,N_f-1]$, $z^0(k)=x^0(k)$, $\hat{p}^0(k)=p^0(k)$, $k\in[N_0,N_f]$.

Step 1 — Compute the parameter $\alpha^i(k)$ to satisfy (5,31). This is called the *parameter estimation step.*

Step 2 — Compute the augmented multipliers $\bar{\lambda}^i(k)$ and $\bar{\beta}^i(k)$ from (5,30).

Step 3 — With specified $\alpha(k)$, $\bar{\lambda}(k)$ and $\bar{\beta}(k)$ solve the modified model-based optimal control problem MMOP5 (by using Procedure 5.3.1 if $q=0$ or Procedure 5.3.2 if $q>0$) to obtain $u^{i+1}(k)$, $x^{i+1}(k)$ and $p^{i+1}(k)$. This is called the *system optimisation step.*

Step 4 — This step tests convergence and updates the estimate for the optimal solution of ROP5. The simple relaxation method (5,19) is employed to satisfy (5,18). If $v^{i+1}(k) = v^i(k)$, $k\in[N_0,N_f-1]$ within a given tolerance stop, else set $i=i+1$ and continue from step 1.

In practice, the achievement of the equality $v^{i+1}(k) = v^i(k)$, $k\in[N_0,N_f-1]$ may be evaluated by using the following 2-norm (*control variation norm* between iterates):

$$\|v^{i+1}-v^i\|_2 = \sqrt{\sum_{N_0}^{N_f-1} \|v^{i+1}(t)-v^i(t)\|^2} \qquad \textbf{(5,34)}$$

and comparing its value with a given small tolerance ε_1.

5.4 EXACT DISCRETIZATION OF CONTINUOUS TIME SYSTEMS

It is a common practice (Jamshidi, 1983; Teo *et al*, 1991; Singh, 1980) when formulating a nonlinear discrete-time optimal control problem of a continuous system described by the differential equation $\dot{x} = f_c(x(t), u(t), t)$ to discretize with sampling time T the continuous time differential equation by its first order (or Euler) approximation $x(k+1) = x(k) + Tf_c(x(t_k), u(t_k), t_k)$. However, for this discretization scheme to be a good approximation of the continuous time system, it is required that the sampling time be very small in comparison with the dynamics of the system and even so, it can diverge from the actual solution (Press et al, 1992).

To overcome the low accuracy of the first order approximation, an *exact discretization* of the differential equation can be implemented as follows:

$$x(k+1) = x(k) + \int_{t_k}^{t_{k+1}} f_c(x(\tau), u(\tau), \tau)\, d\tau \qquad \textbf{(5,35)}$$

noticing that because of the zero order hold, the control signal is held constant between sampling times $u(\tau) = u(k),\ \tau \in [t_k, t_{k+1}]$. Equation (5,35) denotes the integration of the continuous state equation from $x(t_k) = x(k)$, given $u(k)$, over one sampling interval to obtain $x(k+1)$.

The term *exact* here denotes the high accuracy that may be achieved by using a good and well tuned ordinary differential equation (ODE) solver. A similar approach has been proposed by Sage and White (1977), who suggest the exact discretization of *both* the continuous dynamics and a continuous performance index. Notice, however, that in the scheme proposed here the performance index is formulated in discrete-time and that MMOP5 continues to be discrete.

5.5 SIMULATION EXAMPLES

Algorithm 5.3.3 was implemented in the C++ programming language using object oriented and modular programming techniques. All the derivatives (jacobian matrices, gradients) were computed by using the Central Difference Formula (Press

et al, 1992). The exact discretization scheme was implemented as a C++ module where the integration in (5,35) can be carried out by a 4th order fixed step size Runge-Kutta integrator, a 5th order adaptive step size Runge-Kutta integrator, or a stiff integrator (Press et al, 1992) depending on the accuracy requirements, the numerical conditioning of the differential equation and the computational time restrictions.

The following examples were run on an IBM compatible 486-DX based machine with 33 MHz clock speed.

Example 5.5.1: nonlinear discrete-time system

Consider the following example proposed by (Mahmoud et al, 1978) and used later as a benchmark to compare solutions of nonlinear optimal control problems (Papageorgiou and Smith, 1980). The problem consists in the minimization of a quadratic performance index subject to nonlinear second order discrete-time dynamic constraints. The dynamic equation in MOP consists in a linearization of the real dynamics about the origin. Relaxation gains and convexification factors were set to the default values one and zero, respectively. The tolerance specified for convergence was $\varepsilon_v = 0.05$.

ROP:

$$\min_{u(k)} \sum_{k=0}^{50} \{ 0.1x_1(k)^2 + 0.1x_2(k)^2 + 0.2u_1(k)^2 + 0.1u_2(k)^2 \}$$

subject to

$$x_1(k+1) = 0.9x_1(k) + 0.1x_2(k) + 0.1u_1(k)$$

$$x_2(k+1) = 0.2x_1(k) + 0.1x_2(k) - 0.1x_2(k)^2 + 0.1u_2(k)$$

$$x(0) = [10 \quad 4.5]^T$$

MOP:

$$\min_{u(k)} \sum_{k=0}^{50} \{ 0.1x_1(k)^2 + 0.1x_2(k)^2 + 0.2u_1(k)^2 + 0.1u_2(k)^2 - \gamma(k) \}$$

subject to

$$x(k+1) = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.1 \end{bmatrix} x(k) + \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} u(k) + \alpha(k)$$

$$x(0) = [10 \quad 4.5]^T$$

Table 5.5.1 shows the algorithm's performance, where J_0^* is the initial performance index and J^* is the final performance index. Figures 5.5.1.1 and 5.5.1.2 show the final state responses and control signals, respectively. Figures 5.5.1.3 and 5.5.1.4 illustrate the convergence behaviour of the algorithm.

No. iterations	CPU (s)	J_0^*	J^*
5	12	31.0091	30.9781

Table 5.5.1: Algorithm's performance, example 5.5.1

The results presented here may be compared with those published by Papageourgiou and Smith (1980).

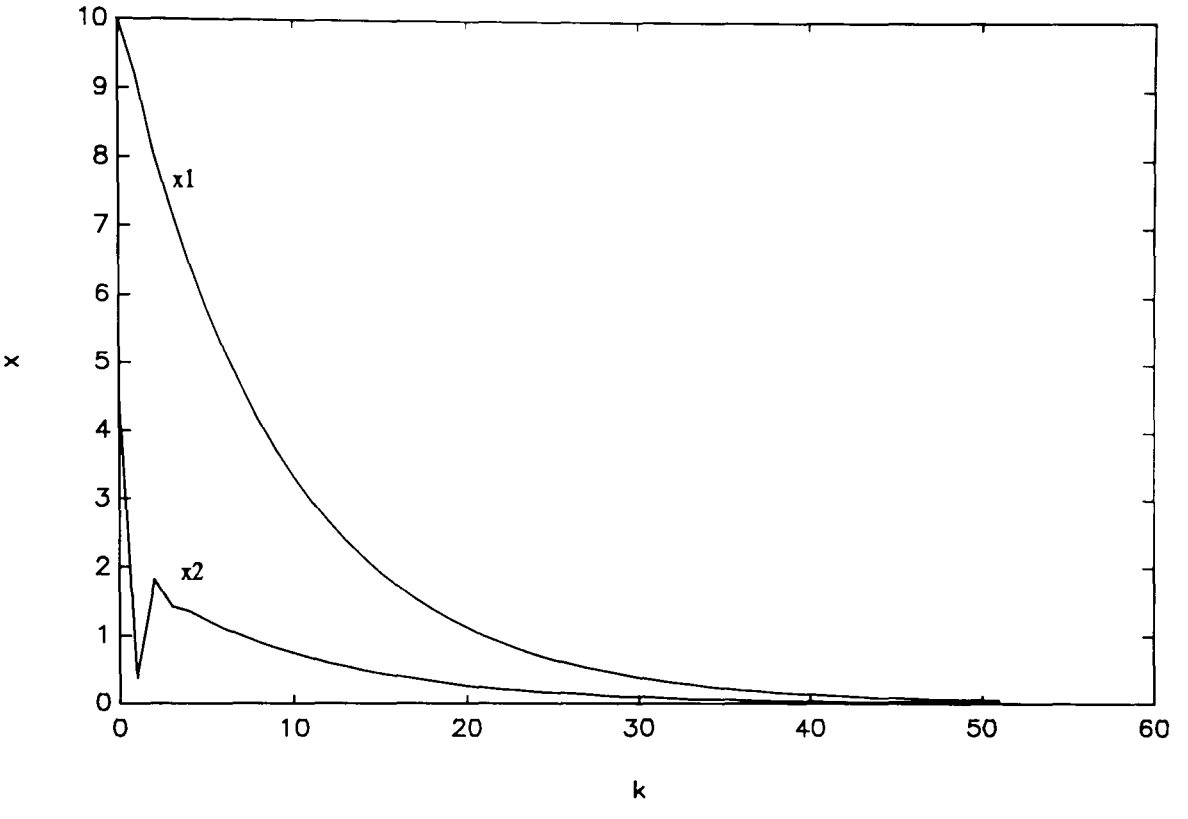

Figure 5.5.1.1: Example 5.5.1, final state response

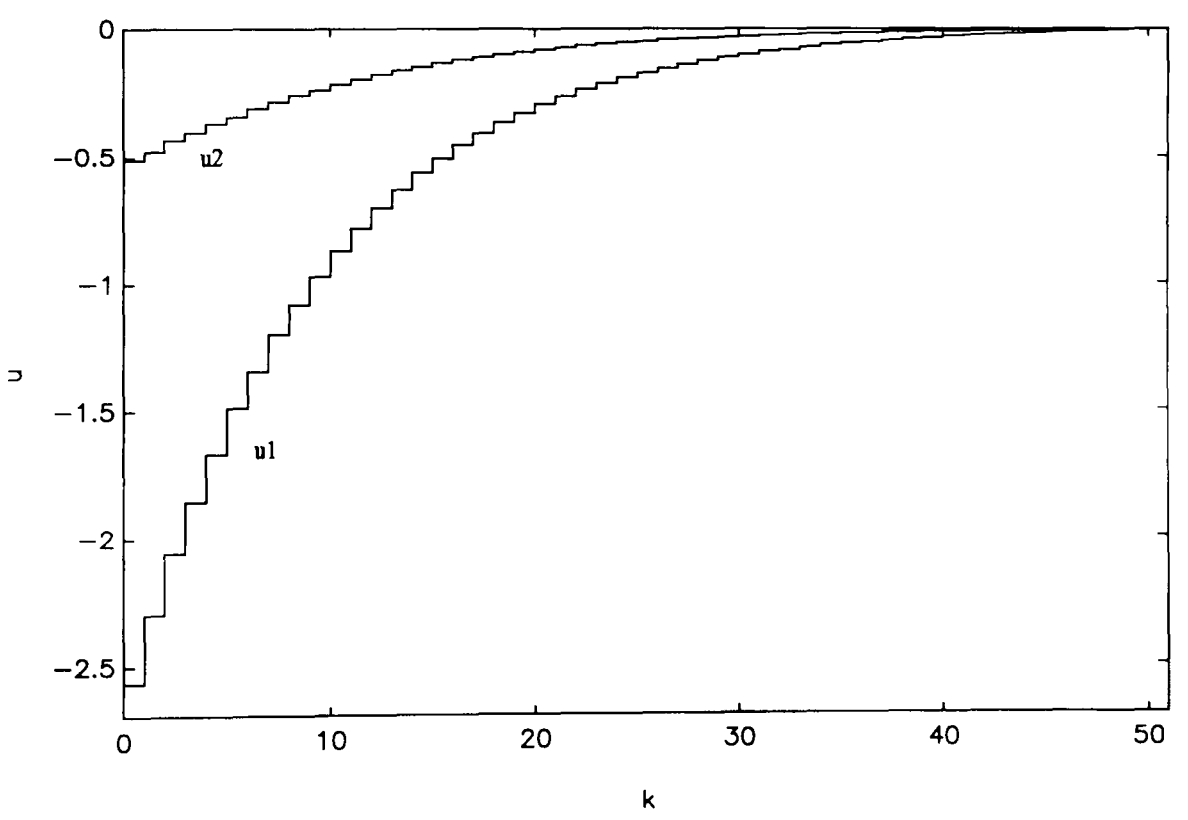

Figure 5.5.1.2: Example 5.5.1, final control vector

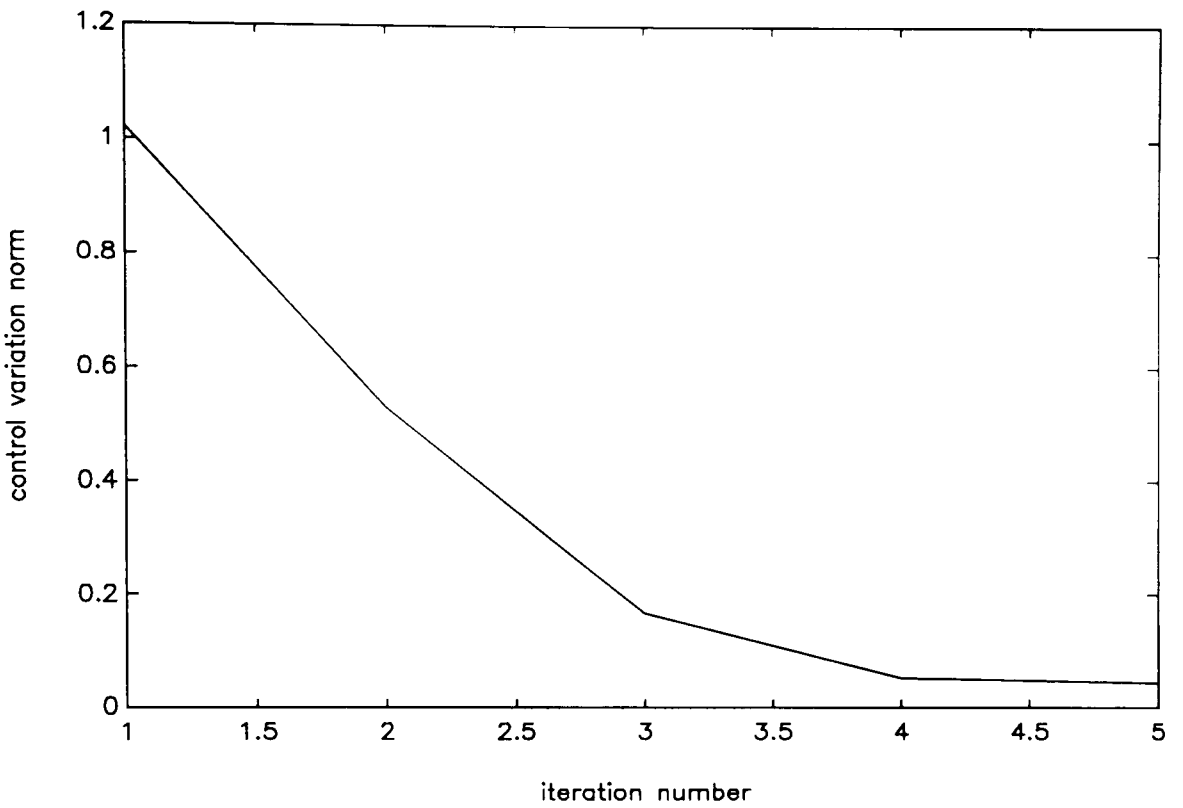

Figure 5.5.1.3: Example 5.5.1, control variation norm vs. iterations

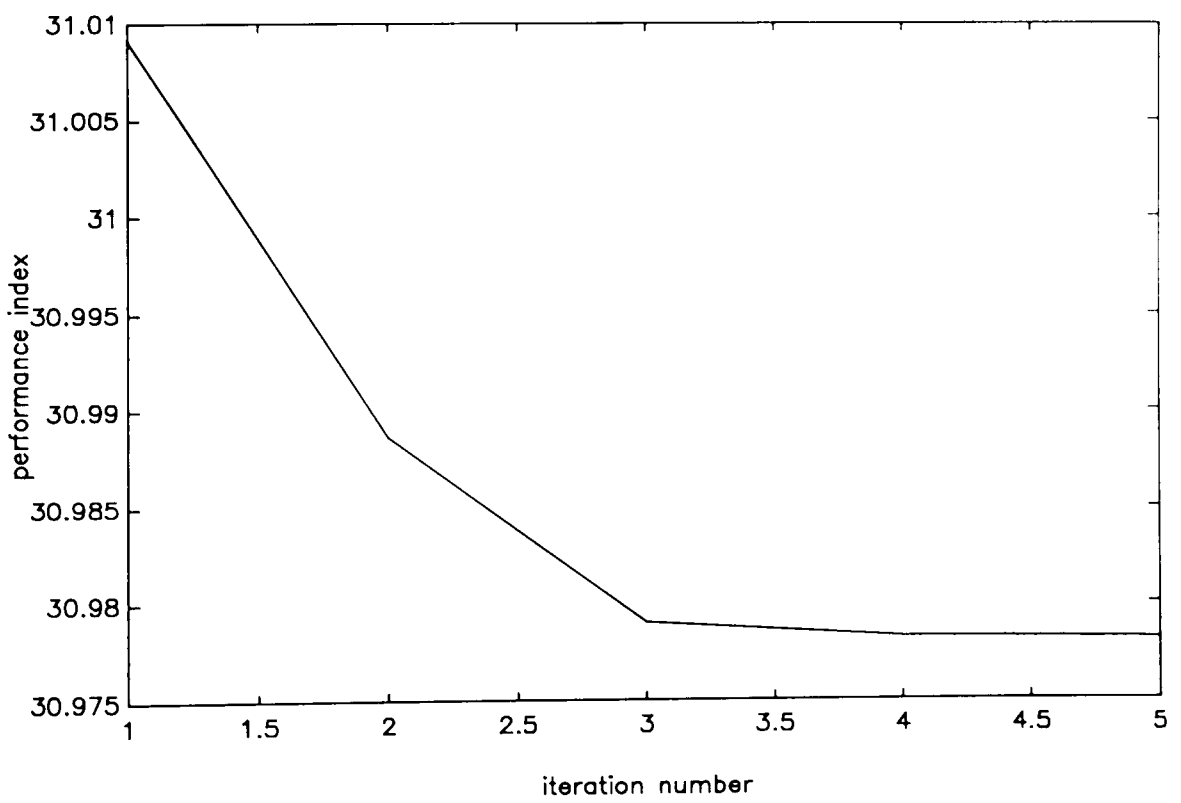

Figure 5.5.1.4: Example 5.5.1 performance index vs. iterations

Example 5.5.2: continuous stirred tank reactor (CSTR) with bounded control

This example consists in a discrete-time version of example 3.4.1, where the control signal is bounded between upper and lower levels $-1 \le u(t) \le 1$. Control bounds were handled by using the variable transformation technique described in Section 3.3. The model-based dynamics have been chosen as a linearization about the origin. The exact discretization scheme was used for handling the continuous dynamics, using 4th order Runge-Kutta integration. The numerical integration step between sampling times was $\Delta t = 0.004$, the control signal sample interval was $T = 0.02$, and the tolerance specified for convergence was $\varepsilon_v = 0.01$. The relaxation gains k_z, and k_p were both set to 1 and the convexification factor r_2 was set to zero. The values of k_v and r_1 were set to 0.9 and 0.5, respectively.

ROP:

$$\min_{u(k)} \sum_{k=0}^{38} (x_1(k)^2 + x_2(k)^2 + 0.1u(k)^2)$$

subject to

$$x(k+1) = x(k) + \int_{t_k}^{t_{k+1}} f_c(x(\tau), u(k), \tau)\, d\tau$$

$$x(0) = [0.05 \quad 0]^T$$

$$-1 \le u(k) \le 1$$

where the continuous dynamics represented by $\dot{x} = f_c(x, u, t)$ are given by:

$$\dot{x}_1 = -(x_1 + 0.25) + (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right) - (1 + u)(x_1 + 0.25)$$

$$\dot{x}_2 = 0.5 - x_2 - (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right)$$

MOP:

$$\min_{u(k)} \sum_{k=0}^{38} (x_1(k)^2 + x_2(k)^2 + 0.1u(k)^2 - \gamma(k))$$

subject to

$$x(k+1) = \begin{bmatrix} 1.087412 & 0.02046 \\ -0.127875 & 0.959537 \end{bmatrix} x(k) + \begin{bmatrix} -0.005216 \\ 0.000317 \end{bmatrix} u(k) + \alpha(k)$$

$$x(0) = [0.05 \quad 0]^T$$

$$-1 \leq u(t) \leq 1$$

Table 5.5.2 shows the algorithm's performance. Figures 5.5.2.1 and 5.5.2.2 show the final state responses and control signals, respectively. Figures 5.5.2.3 and 5.5.2.4 illustrate the convergence behaviour of the algorithm.

No. iterations	CPU (s)	J_0^*	J^*
9	48	2.685687	1.444583

Table 5.5.2: Algorithm's performance, example 5.5.2

It is important to make comparisons with example 3.4.1. We may remark that the performance index in example 5.5.2 is proportional to a first order approximation to the performance index in example 5.5.2. We may notice that the optimal discrete-time control signal shown in Figure 5.5.2.2 is close to its continuous counterpart, shown in Figure 3.4.1.1. Moreover, the CPU time per iteration is shorter in example 5.5.2.2. This indicates that if the sampling time increases, the computational savings using Algorithm 5.3.3 and the exact discretization scheme increase. An increase in the sampling time will be limited in practical cases by the desired resolution in the control signal and by other factors influencing the choice of the sampling interval (Åström and Wittenmark, 1990). Notice that in Procedure 2.3.1 we have to solve differential equations (which may require short integration steps and might even be

numerically stiff) while in Procedure 5.3.1 we have to solve *difference* equations (which do not present such computational drawbacks).

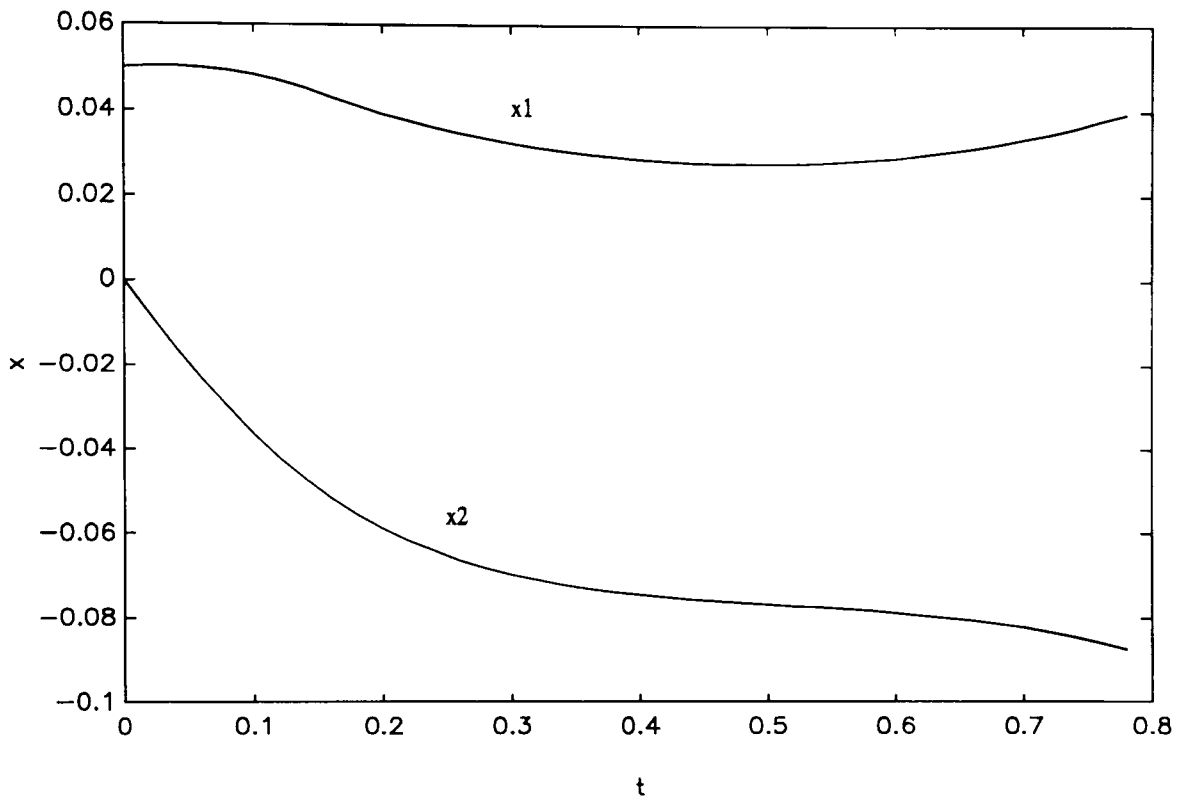

Figure 5.5.2.1: Example 5.5.2, final state response

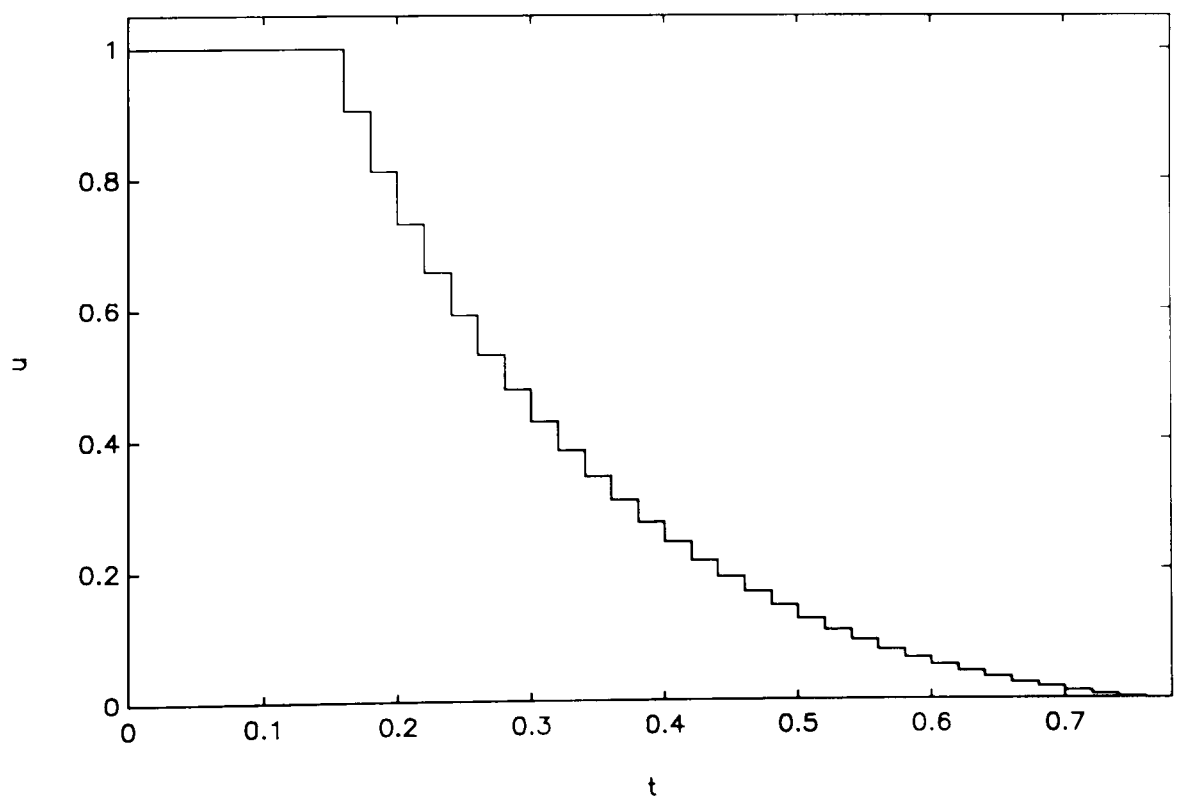

Figure 5.5.2.2: Example 5.5.2, final control signal

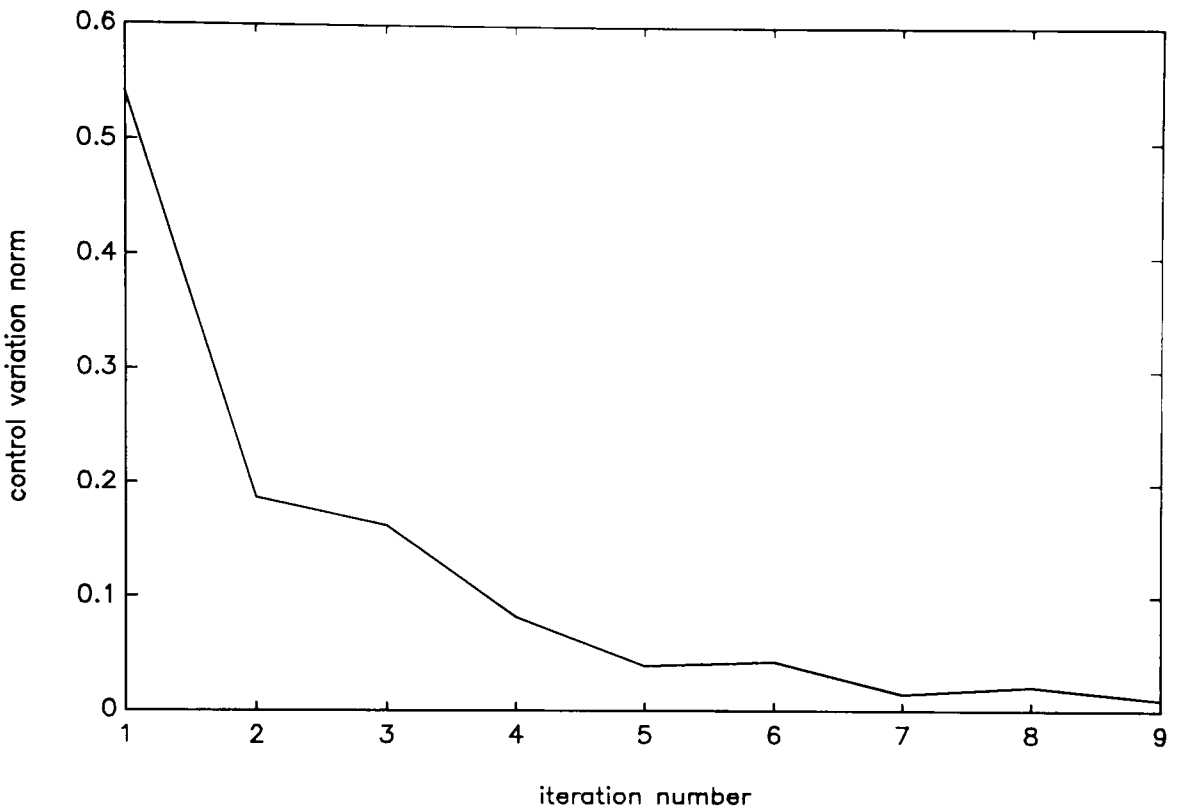

Figure 5.5.2.3: Example 5.5.2, control variation norm vs. iterations

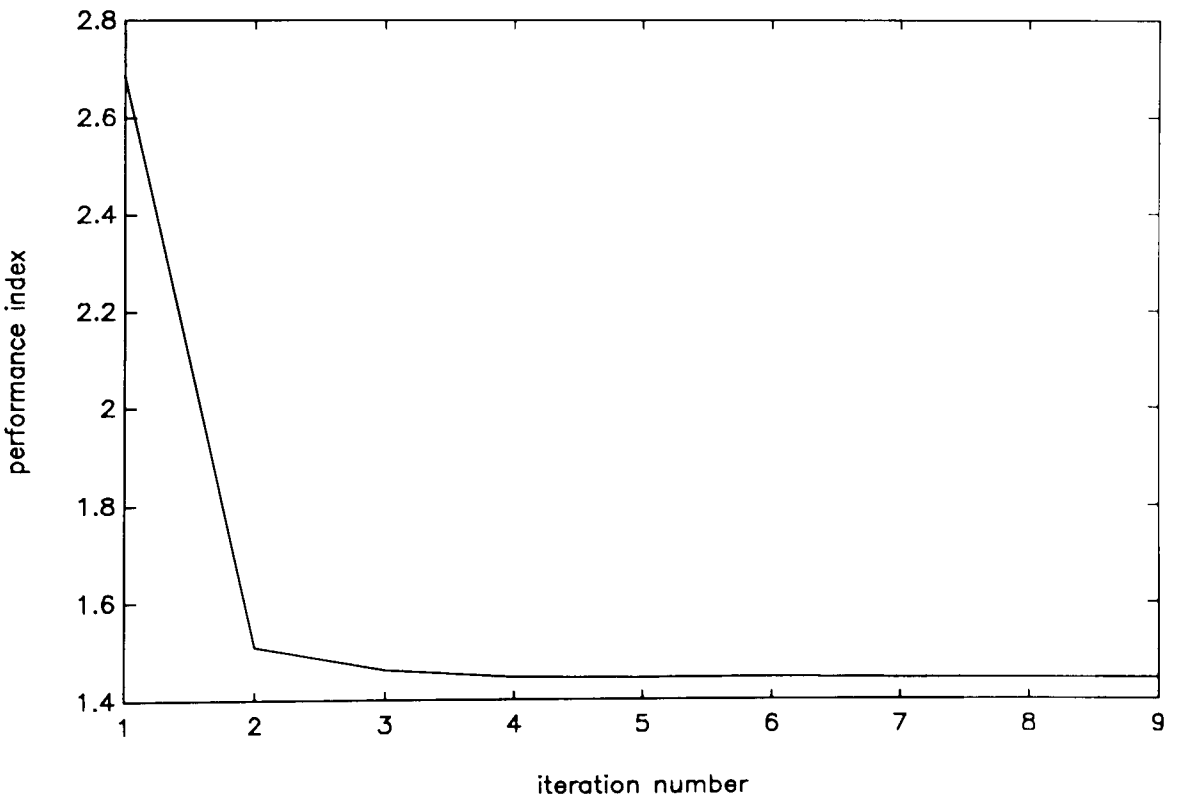

Figure 5.5.2.4: Example 5.5.2 performance index vs. iterations

Example 5.5.3: CSTR with bounded control and terminal constraints

This example is identical to example 5.5.2, with the added difficulty of a terminal constraint on the values of both states:

$$x(39) = [0\ 0]^T$$

Table 5.5.3 shows the algorithm's performance as well as the nominal ($x_1^0(39)$) and final ($x_1(39)$) values of the terminal states. Figures 5.5.3.1 and 5.5.3.2 show the final state responses and control signals, respectively. Figure 5.5.3.3 illustrates the convergence behaviour of the algorithm.

No. Iterations	CPU (s)	$x^0(39)$	$x(39)$	J^*
38	186	$\begin{bmatrix} 0.427761 \\ -0.469444 \end{bmatrix}$	$\begin{bmatrix} 0.001190 \\ -0.001335 \end{bmatrix}$	2.643502

Table 5.5.3: Algorithm's performance, example 5.5.3

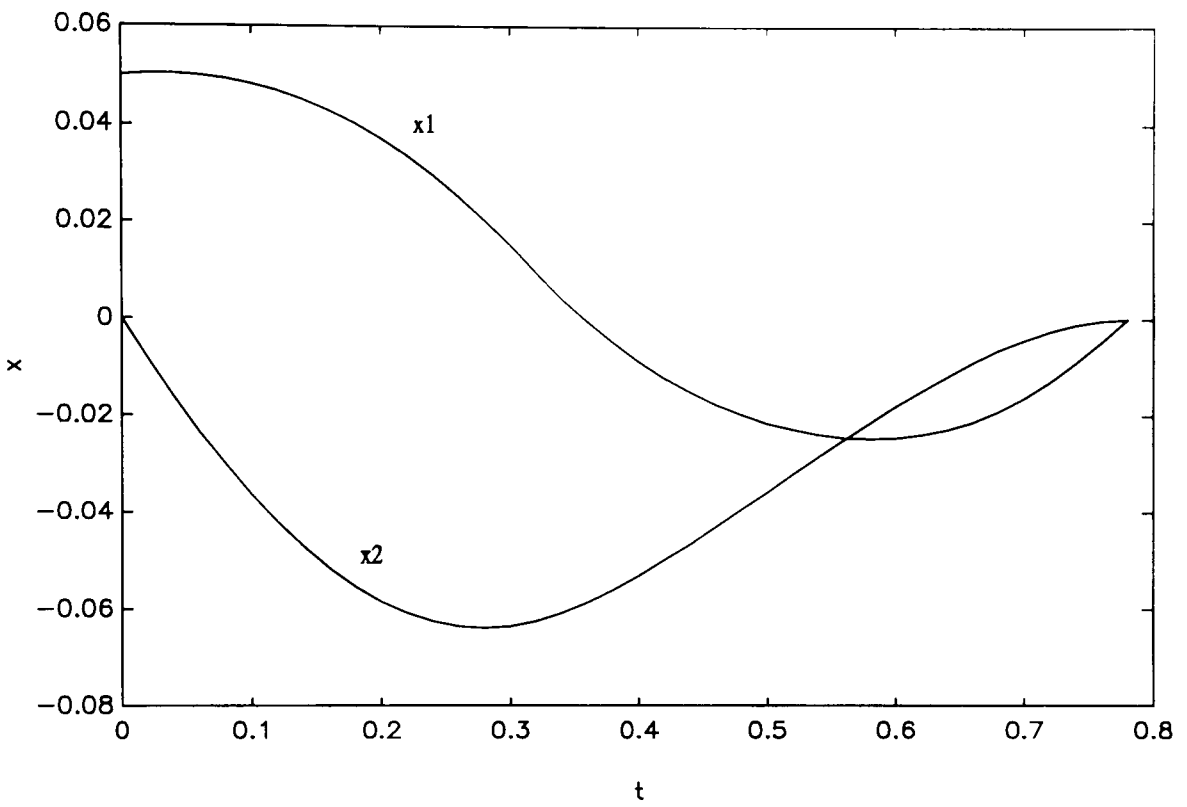

Figure 5.5.3.1: Example 5.5.3, final state response

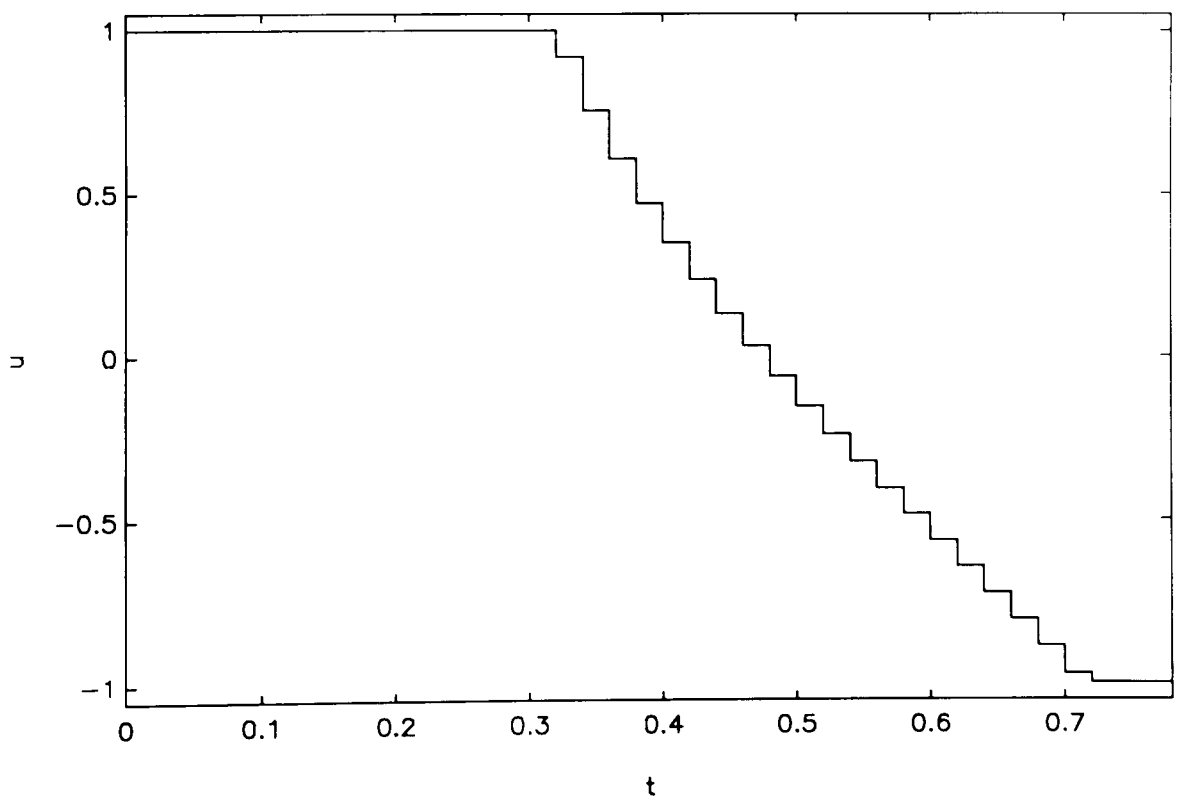

Figure 5.5.3.2: Example 5.5.3, final control signal

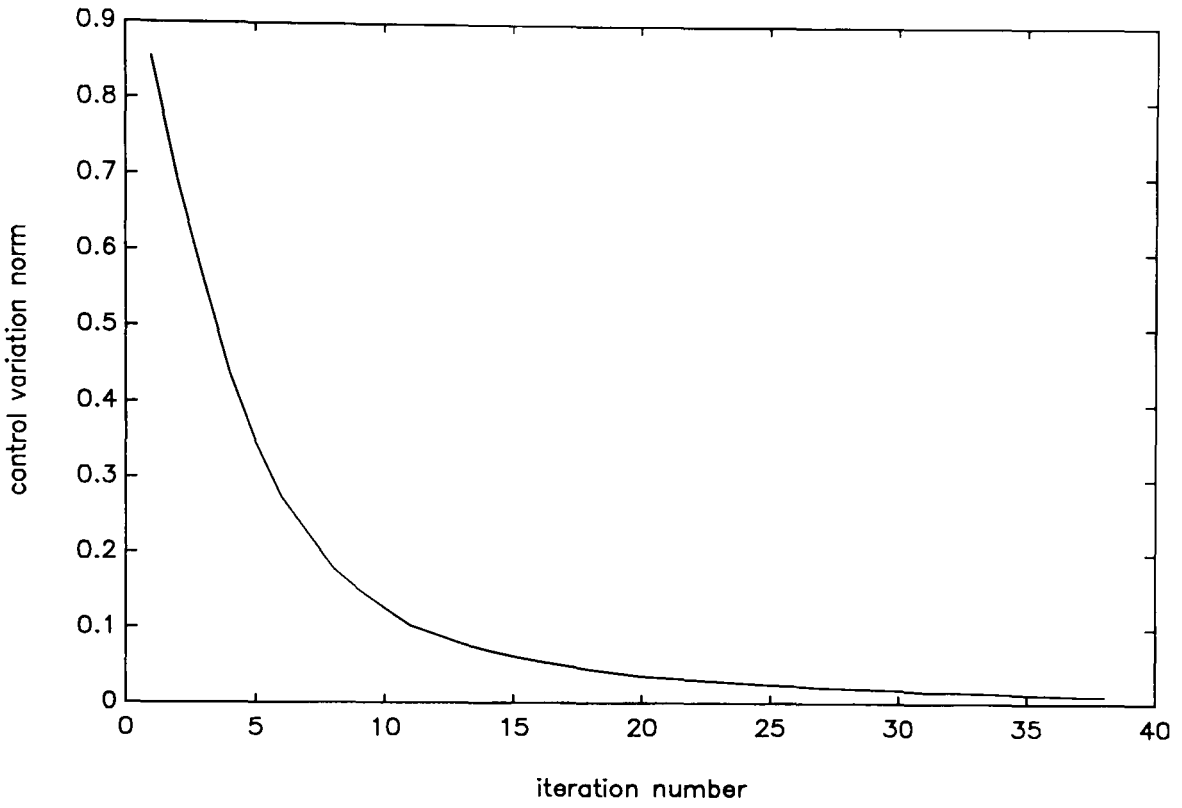

Figure 5.5.3.3: Example 5.5.3, control variation norm vs. iterations

5.6 SUMMARY

An algorithm has been presented for the solution of discrete-time optimal control problems where there are differences from reality, either intentionally in order to facilitate the solution of complex nonlinear problems or due to uncertainties, and the model used in the computations.

A version of the discrete-time DISOPE algorithm which uses a linear model and a quadratic performance criterion has been developed and implemented in the C++ programming language.

The *exact discretization scheme* has been introduced for optimising continuous time systems using the implementation of the discrete-time DISOPE algorithm, which allows the use of standard (and convenient) discrete-time linear-quadratic calculations at the model-based level.

The implemented algorithm has been tested with three simulation examples with model-reality differences. The results show that the real optimal solution is obtained in spite of the differences between the real and model-based problems. The algorithm as implemented is capable of handling nonlinear discrete-time optimal

control problems with terminal state equality constraints, non-quadratic performance indexes and multiple control inputs.

CHAPTER 6

SET-POINT TRACKING DISOPE ALGORITHM

In this chapter, the discrete time DISOPE algorithm developed in Chapter 5 is applied to the solution of the nonlinear tracking optimal control problem. A version of the algorithm with a linear-quadratic model-based problem is developed and implemented in software. The algorithm implemented is tested with simulation examples. The research work presented in this chapter is also described in (Becerra, 1993d; Becerra and Roberts, 1994b)

6.1 TRACKING OPTIMAL CONTROL

When the output variables of a system are required to follow or track a reference trajectory over a time horizon while minimizing a given performance index, a so-called tracking optimal control problem is formulated. Such controls are important, for example, in the control of spacecraft and robot arms (Lewis, 1986a) and in the formulation of predictive controllers (Soeterboek, 1992).

6.2 SET-POINT TRACKING DISOPE ALGORITHM

6.2.1 Formulation in discrete time

The reader is referred to Chapter 5 for the development of the discrete-time DISOPE algorithm. It must be emphasized that what follows is a particular case of the discrete time DISOPE algorithm (Algorithm 5.2.1) in which the performance index has a particular structure and is expressed in terms of an output reference trajectory which the system output is desired to follow. Therefore, the discrete time DISOPE algorithm remains basically unaltered, what changes is the formulation.

Consider the real optimal control problem (ROP5) defined in Section 5.2.1 Assume that the performance measure functions can be expressed in terms of the control vector, an output vector $y(k) \in \Re^{n_o}$ given by

$$y(k)=\zeta(x(k)) \tag{6,1}$$

where $\zeta : \Re^n \rightarrow \Re^{n_o}$ is the real output function, and a (known) reference trajectory $r_o(k) \in \Re^{n_o}$, $k \in [N_0, N_f]$, in such a way that the output is required to follow or track the reference trajectory. The necessary optimality conditions of ROP5 are given in Section 5.2.1. Instead of solving ROP5, the possibly simplified model-based optimal control problem (MOP5) defined in Section 5.2.1 is considered. Furthermore, MOP5 can be chosen as a linear-quadratic approximation of ROP5 where there are standard procedures for its solution (Lewis, 1986a, Sage and White, 1977). This provides a computational advantage. Assume now

$$\varphi = \tfrac{1}{2}(Cx(N_f)-r_0(N_f))^{\top}\Phi(Cx(N_f)-r_0(N_f)) \tag{6,2}$$

$$L = \tfrac{1}{2}(Cx(k)-r_0(k))^{\top}Q(Cx(k)-r_0(k))+\tfrac{1}{2}u(k)^{\top}Ru(k)+\gamma(k) \tag{6,3}$$

$$f = Ax(k)+Bu(k)+\alpha(k) \tag{6,4}$$

where the term $Cx(k)$ is a linear approximation of the real output function ζ, and C, Φ, Q, R, A, and B are matrices of the appropriate dimensions. Hence, including the augmentation terms described in Section 5.2.2, the augmented Hamiltonian (5,9) may be written as

$$\begin{aligned} H &= \tfrac{1}{2}(Cx(k)-r_o(k))^{\top}Q(Cx(k)-r_o(k)) + \tfrac{1}{2}u(k)^{\top}Ru(k) \\ &+p(k+1)^{\top}(Ax(k)+Bu(k)+\alpha(k)) - \lambda(k)^{\top}u(k) - \beta(k)^{\top}x(k) \\ &+ \tfrac{1}{2}r_1\|u(k)-v(k)\|^2 + \tfrac{1}{2}r_2\|x(k)-z(k)\|^2 \end{aligned} \tag{6,5}$$

Thus, based on the definition of MMOP5 given in Section 5.2.1 the modified model- based optimal control problem with reference trajectory information (MMOP6), whose iterative solution provides, after convergence, the solution of ROP5 is then defined as:

MMOP6

$$\min_{u(k)} \left\{ \tfrac{1}{2}(Cx(N_f)-r_o(N_f))^{\top}\Phi(Cx(N_f)-r_0(N_f)) - \sum_{k=N_0}^{N_f-1} \left[\tfrac{1}{2}(Cx(k)-r_0(k))^{\top}Q(Cx(k)-r_0(k)) \right.\right.$$

$$\left.\left. +\tfrac{1}{2}u(k)^{\top}Ru(k)+\gamma(k)-\lambda(k)^{\top}u(k)-\beta(k)^{\top}x(k)+\tfrac{1}{2}r_1\|u(k)-v(k)\|^2+\tfrac{1}{2}r_2\|x(k)-z(k)\|^2 \right] \right\}$$

subject to

$$x(k+1)=Ax(k)+Bu(k)+\alpha(k)$$
$$x(N_0)=x_o$$

Applying the model-based optimality conditions (5,12), (5,13), (5,14) and (5,15) with H given by (6,5), we obtain the control law:

$$u(k) = -\bar{R}^{-1}(B^{\top}p(k+1) - \bar{\lambda}(k)) \qquad \textbf{(6,6)}$$

and, in addition, the following TPBVP:

$$\begin{aligned} x(k+1) &= Ax(k) - B\bar{R}^{-1}(B^{\top}p(k+1) - \bar{\lambda}(k)) + \alpha(k) \\ p(k) &= \bar{Q}x(k) + A^{\top}p(k+1) - \bar{\beta}(k) \end{aligned} \qquad \textbf{(6,7)}$$

with border conditions:

$$\begin{aligned} x(N_0) &= x_0 \\ p(N_f) &= C^{\top}\Phi(Cx(N_f)-r_0(N_f)) \end{aligned} \qquad \textbf{(6,8)}$$

where the augmented weighting matrices $\bar{R}$ and $\bar{Q}$ are given by:

$$\begin{aligned} \bar{R} &= R + r_1 I_m \\ \bar{Q} &= C^{\top}QC + r_2 I_n \end{aligned} \qquad \textbf{(6,9)}$$

and the augmented multipliers $\bar{\lambda}(k)$ and $\bar{\beta}(k)$ are expressed as:

$$\begin{aligned} \bar{\lambda}(k) &= \lambda(k) + r_1 v(k) \\ \bar{\beta}(k) &= \beta(k) + r_2 z(k) + C^{\top}Q r_o(k) \end{aligned} \qquad \textbf{(6,10)}$$

It is observed that the structure of TPBVP (6,7) is identical to that of TPBVP (5,28), but (6,7) has a different boundary condition on the costate and is based on a different definition for $\bar{\beta}(k)$. The solution is obtained by the sweep method (Lewis, 1986a, Bryson and Ho, 1975). The key is to assume the relationship between costate and state as $p(k) = S(k)x(k) + h(k)$, where $S(k)$ is a *nxn* matrix and $h(k) \in \Re^n$. This gives rise to the following noniterative solution procedure (see Appendix F for the derivation):

Procedure 6.2.1: Solution of set-point tracking MMOP6

Step a: Solve backwards from $k = N_f - 1$ to N_0 the following *difference* equations, with terminal conditions $S(t_f) = C^T \Phi C$ and $h(t_f) = -C^T \Phi r_0(N_f)$:

$$S(k) = \bar{Q} + A^T S(k+1)(A - BG(k))$$

$$G(k) = [\bar{R} + B^T S(k+1)B]^{-1} B^T S(k+1)A$$

$$h(k) = (A - BG(k))^T h(k+1) + (A - BG(k))^T S(k+1)\alpha(k) - \bar{\beta}(k) + G(k)^T \bar{\lambda}(k)$$

Step b: Compute the driving input $g(k)$, $k \in [N_0, N_f - 1]$ from:

$$g(k) = [\bar{R} + B^T S(k+1)B]^{-1}[-B^T S(k+1)\alpha(k) - B^T h(k+1) + \bar{\lambda}(k)]$$

Step c: Compute the state $x(k)$, $k \in [N_0, N_f]$ by solving from the initial condition $x(N_0) = x_0$ the following difference equation:

$$x(k+1) = (A - BG(k))x(k) + Bg(k) + \alpha(k)$$

Step d: Compute the costate $p(k)$, $k \in [N_0, N_f]$ from:

$$p(k) = S(k)x(k) + h(k)$$

Step e: Compute the control input $u(k)$, $k \in [N_0, N_f - 1]$ from the control law:

$$u(k) = -G(k)x(k) + g(k)$$

The linear-quadratic formulation enables the augmented multipliers $\bar{\lambda}(k)$ and $\bar{\beta}(k)$, $k\in[N_0,N_f-1]$ to be expressed as (see equation (5,16)):

$$\begin{aligned}\bar{\lambda}(k) &= [B-\frac{\partial f^*}{\partial v(k)}]^T\hat{p}(k+1) + [\bar{R}v(k)-\nabla_{v(k)}L^*]\\ \bar{\beta}(k) &= [A-\frac{\partial f^*}{\partial z(k)}]^T\hat{p}(k+1) + [\bar{Q}z(k)-\nabla_{z(k)}L^*]\end{aligned} \qquad \textbf{(6,11)}$$

while the calculation of parameter $\alpha(k)$, $k\in[N_0,N_f-1]$ becomes (see equation (5,17)), noting that it is not necessary to calculate $\gamma(k)$:

$$\alpha(k) = f^*(z(k),v(k),k) - Az(k) - Bv(k) \qquad \textbf{(6,12)}$$

DISOPE requires a nominal solution to start the iterations. A recommended starting point is to use the solution of MMOP6 under $\alpha(k)=0$, $\bar{\lambda}(k)=0$, $\bar{\beta}(k)=0$, $k\in[N_0,N_f-1]$, $r_1 = r_2 = 0$ (relaxed MMOP6).

The above analysis enables us to formulate the discrete time DISOPE algorithm (Algorithm 5.2.1) as a tracking optimal control algorithm with a linear-quadratic model-based problem.

Algorithm 6.2.1: Set-point tracking DISOPE algorithm with LQ model-based problem

---

Data: A, B, Q, R, Φ, r_1, r_2, k_v, k_z, k_p, x_0, N_0, N_f, q, $r_0(k)$, $k\in[N_0,N_f]$ and means for calculating f^*, L^* and ζ.

Step 0: Compute or choose a nominal solution $u^0(k)$, $x^0(k)$ and $p^0(k)$. Set $i=0$, $v^0(k)=u^0(k)$, $k\in[N_0,N_f-1]$, $z^0(k)=x^0(k)$, $\hat{p}^0(k)=p^0(k)$, $k\in[N_0,N_f]$.

Step 1: Compute the parameter $\alpha^i(k)$ to satisfy (6,12). This is called the *parameter estimation step.*

Step 2: Compute the augmented multipliers $\bar{\lambda}^i(k)$ and $\bar{\beta}^i(k)$ from (6,11).

Step 3: With specified $\alpha(k)$, $\bar{\lambda}(k)$ and $\bar{\beta}(k)$, solve the MMOP6 by using Procedure 6.2.1. This is called the *system optimisation step.*

Step 4: This step tests convergence and updates the estimate for the optimal solution of ROP5. The simple relaxation method (5,19) may employed to satisfy (5,18). If $v^{i+1}(k) = v^{i}(k)$, $k \in [N_0, N_f-1]$ within a given tolerance stop, else set $i=i+1$ and continue from step 1.

In practice, the achievement of the equality $v^{i+1}(k) = v^{i}(k)$, $k \in [N_0, N_f-1]$ may be evaluated by computing the control variation norm (5,34) and comparing its value with a given small tolerance ε_v.

6.3 INCREMENTAL CONTROL WEIGHTING

With the purpose of removing zero steady state error for constant reference trajectories incremental control weighting may be introduced in the performance index by using a term of the form $\Delta u(k)^{\top} R \Delta u(k)$, where $\Delta u(k) = u(k)-u(k-1)$ (Soeterboek, 1992). Even though this kind of term is not directly taken into account in the LQ model-based problem, where a term of the form $u(k)^{\top} R u(k)$ is considered, the iterations of DISOPE would deal with it as a *model-reality difference* between L and L^*.

6.4 SIMULATION EXAMPLES

Algorithm 6.2.1 was implemented in the C++ programming language using object oriented and modular programming techniques, making use of existing code generated for the implementation of discrete-time DISOPE (see Sections 5.4 and 5.5). The exact discretization scheme introduced in Section 5.4 was used for the calculations. The following simulations were run on a IBM compatible 486-DX based machine with 33 MHz clock speed.

Example 6.4.1: Exothermal CSTR

This example consists of a first order irreversible chemical reaction carried out under exothermal conditions in a continuous stirred tank reactor (CSTR). Here x_1 represents the dimensionless concentration, x_2 represents the dimensionless temperature (controlled variable), the control variable u is the dimensionless cooling jacket temperature. This reactor has a complex dynamic behaviour and represents a challenging control problem (Sistu and Bequette, 1992). The control signal is bounded between upper and lower levels $-1 \le u(t) \le 2$. Control bounds were handled by using the variable transformation technique described in Section 3.3. There is a set-point change from r_0=0.8859 to r_0=2 at $t = 1.5$, and the reactor is required to track the set-point by minimizing a quadratic performance index. The differential equation is discretized in an exact way as explained in Section 3.2. The control signal sample interval was $T = 0.25$. Here t is a dimensionless time variable.

ROP:

$$\min_{u(k)} \; \tfrac{1}{2}\Phi\,(y(32)-r_0(32))^2+\tfrac{1}{2}\sum_{k=0}^{31}\,[\;(y(k)-r_0(k))^2+0.05\,(P(q^{-1})\,u(k))^2\;]$$

subject to

$$x(k+1) = x(k) + \int_{t_k}^{t_{k+1}} f_c(x(\tau),u(k),\tau)\,d\tau$$

$$x(0) = [0.8560 \quad 0.8859]^T$$

$$-1 \le u(k) \le 2$$

where

$$y(k) = x_2(k)$$

$$r_0(k) = \begin{cases} 0.8859 & k<6 \\ 2.0000 & k\geq 6 \end{cases}$$

$P(q^{-1})$ is a polynomial in the backward shift operator q^{-1}, and the continuous dynamics represented by $\dot{x} = f_c(x,u,t)$ are given by:

$$\dot{x}_1 = -0.072 x_1(t) \exp\left(\frac{x_2(t)}{1+x_2(t)/20}\right) + (1 - x_1(t))$$

$$\dot{x}_2 = 0.576 x_1(t) \exp\left(\frac{x_2(t)}{1+x_2(t)/20}\right) - 1.3 x_2(t) + 0.3 u(t)$$

MOP:

$$\min_{u(k)} \tfrac{1}{2}\Phi(y(32)-r_0(32))^2 + \tfrac{1}{2}\sum_{k=0}^{31} [\ (y(k)-r_0(k))^2 + 0.05u(k)^2 + 2\gamma(k)\ ]$$

subject to

$$x(k+1) = A x(k) + B u(k) + \alpha(k)$$

$$x(0) = [0.8560\ \ 0.8859]^T$$

Here we will distinguish five different cases in order to investigate the influence of the integration method, the linearization point to compute the model-based matrices A and B, the effect of using quadratic incremental control weighting and the influence of the terminal weighting, as indicated in Table 6.4.1.1. In cases (a), (b), (d) and (e) the integration step used was $\Delta t = 0.08333$. Table 6.4.1.2 shows the algorithm's performance in every case. Figures 6.4.1.1 to 6.4.1.3 show the output response for cases (a), (b) and (e), respectively. Figures 6.4.1.4 to 6.4.1.6 show the final control signal for cases (a), (b) and (e), respectively. Figures 6.4.1.7 and 6.4.1.8 show the convergence behaviour of the algorithm for cases (a), (b), (d) and (e).

Case	Integration method	Linearization Point	$P(q^{-1})$	Φ
a	Fixed step size 4th order Runge-Kutta	$x = x_0 \quad u = 0$	1	0
b	Fixed step size 4th order Runge-Kutta	$x = x_0 \quad u = 0$	$1 - q^{-1}$	0
c	Adaptive step size 5th order Runge-Kutta	$x = x_0 \quad u = 0$	$1 - q^{-1}$	0
d	Fixed step size 4th order Runge-Kutta	$x = [0 \ 0]^T \quad u = 0$	1	0
e	Fixed step size 4th order Runge-Kutta	$x = x_0 \quad u = 0$	1	100

Table 6.4.1.1: Description of cases in example 6.4.1

Case	No. of Iterations	CPU (s)	J_0^*	J^*
a	16	49	115.831345	1.121004
b	42	127	113.351143	0.493216
c	37	247	113.352859	0.493183
d	82	233	137.353577	1.120998
e	17	51	117.634392	1.136164

Table 6.4.1.2: Algorithm's performance, example 6.4.1

The values of matrices A and B, according to the linearization point used are:

linearization about the origin $x=[0\ 0]^T\ u=0$

$$A = \begin{bmatrix} 0.764661 & -0.001555 \\ 0.108802 & 0.734551 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0.064705 \end{bmatrix}$$

linearization about initial condition $x=x_0\ \ u=0$

$$A = \begin{bmatrix} 0.742285 & -0.027646 \\ 0.282072 & 0.935978 \end{bmatrix} \qquad B = \begin{bmatrix} -0.0011037 \\ 0.0725700 \end{bmatrix}$$

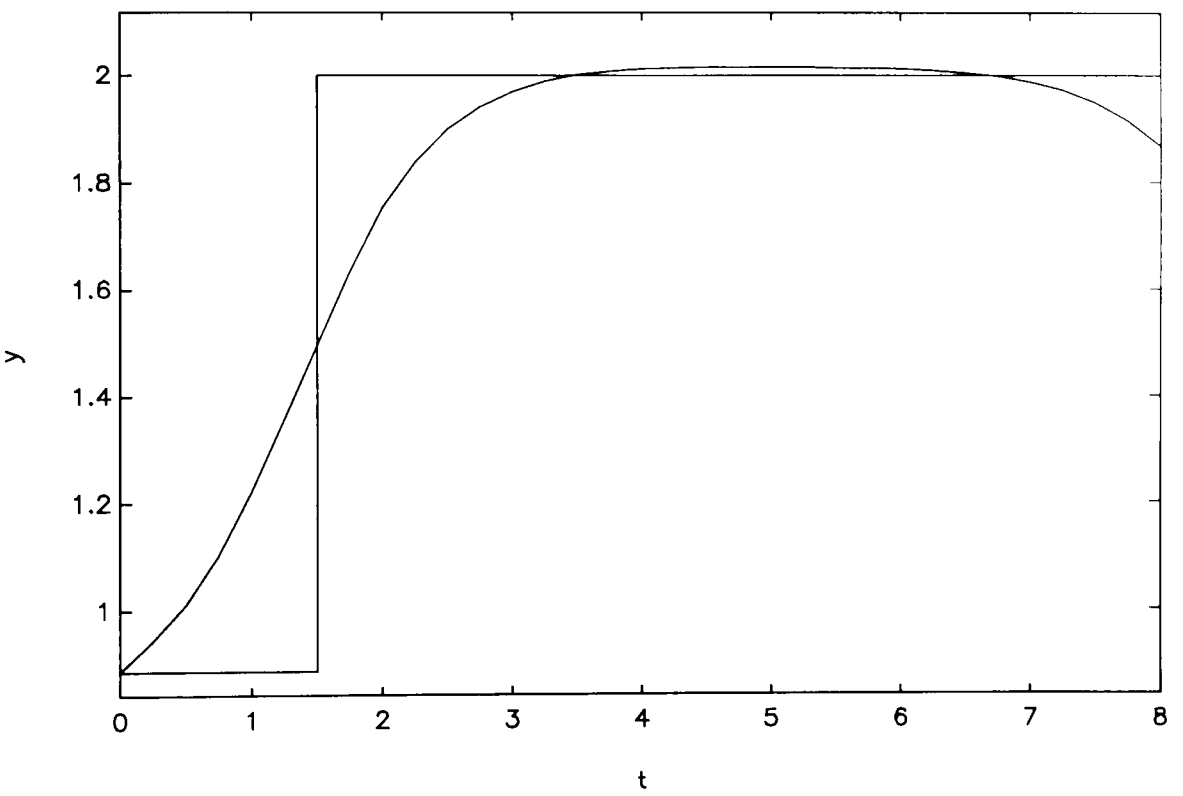

Figure 6.4.1.1: Example 6.4.1.a, dimensionless temperature and set-point

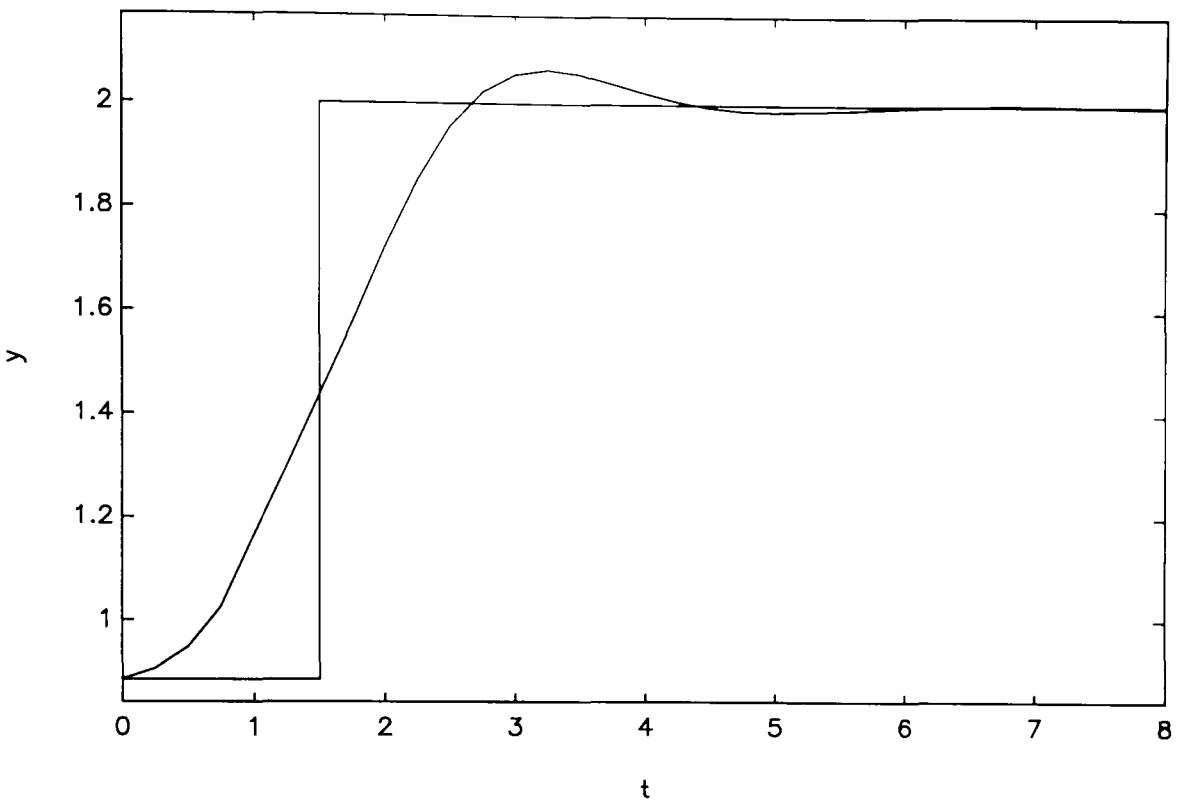

Figure 6.4.1.2: Example 6.4.1.b, dimensionless temperature and set-point

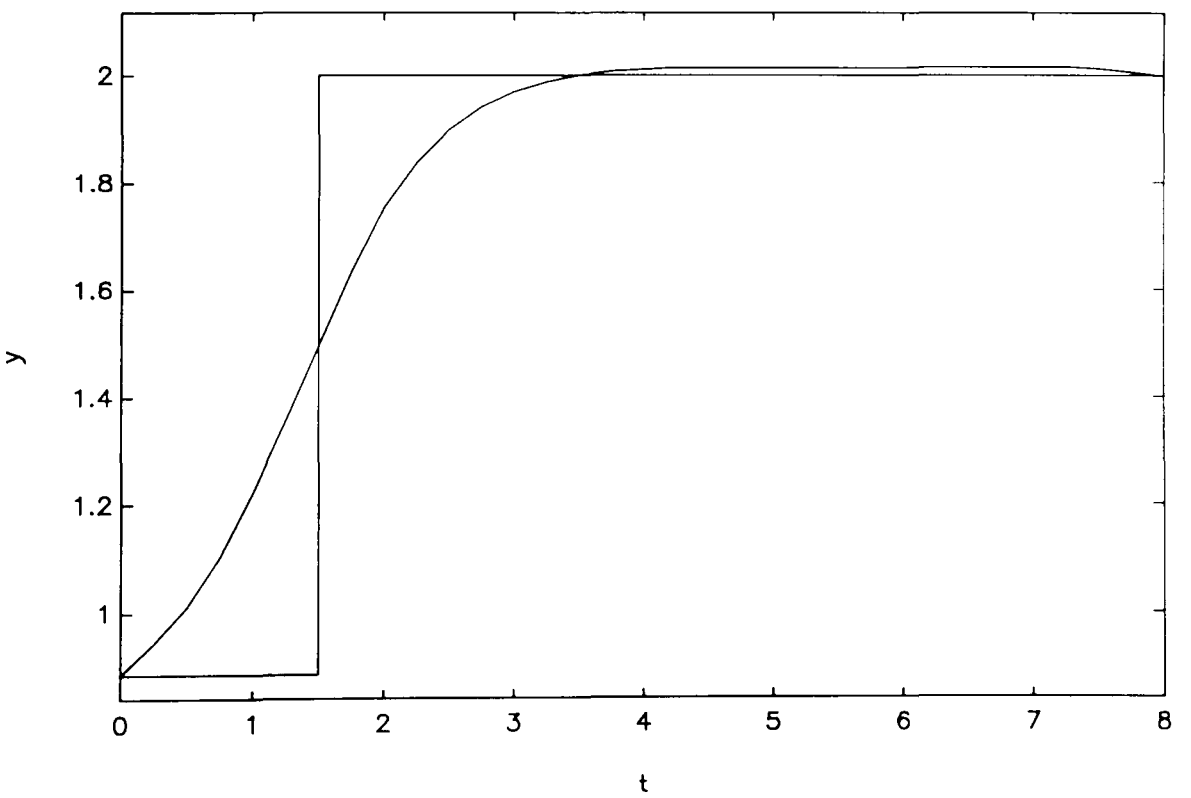

Figure 6.4.1.3: Example 6.4.1.e, dimensionless temperature and set-point

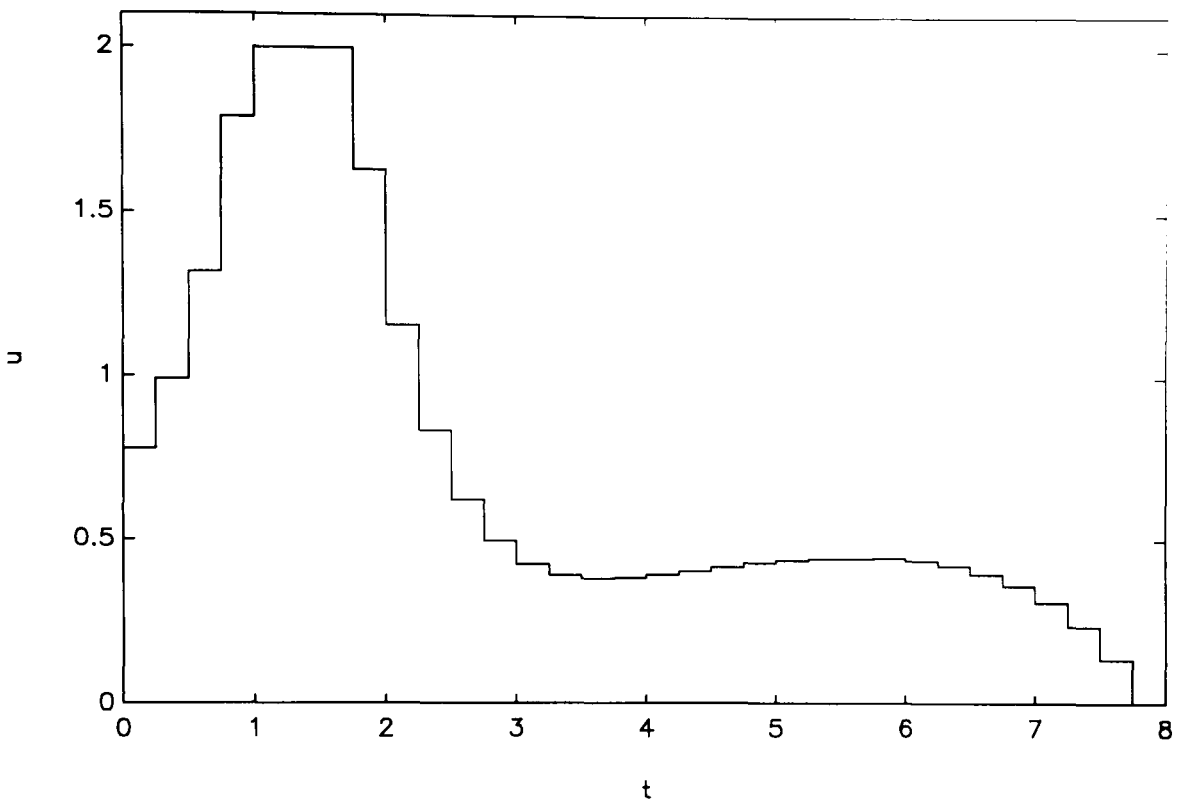

Figure 6.4.1.4: Example 6.4.1.a, final control signal

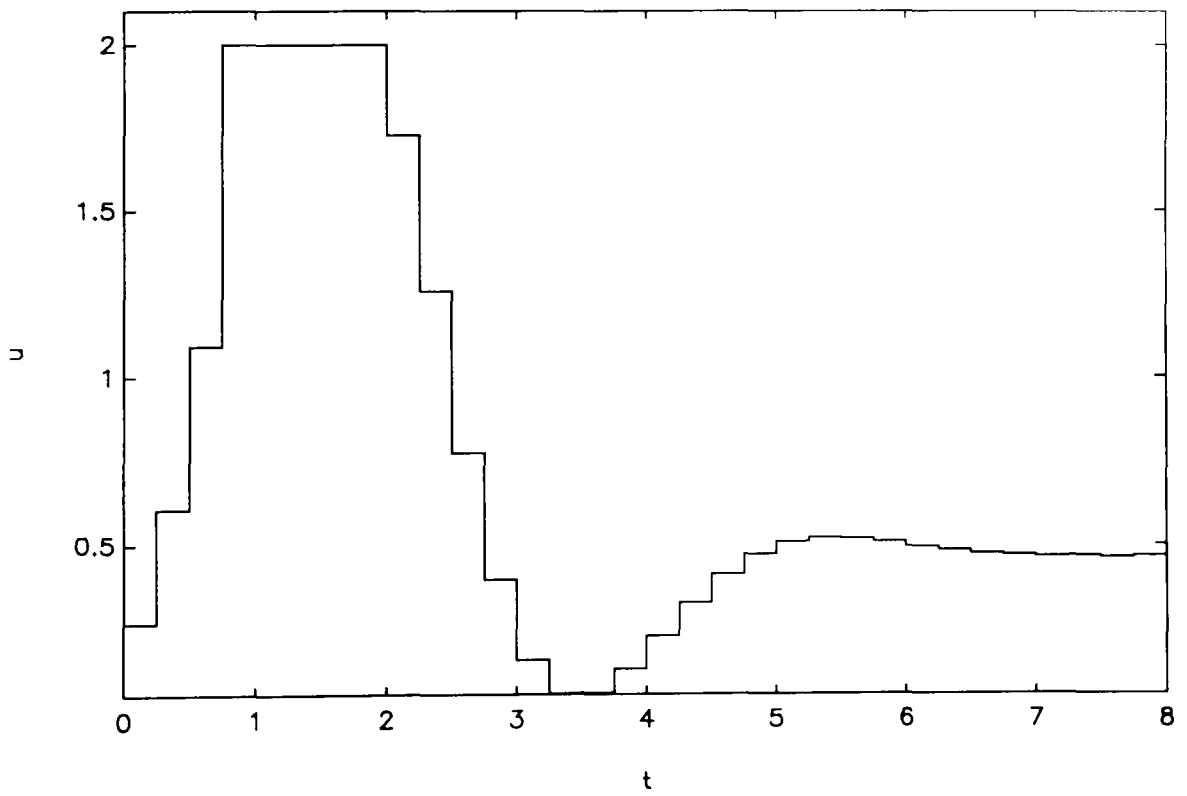

Figure 6.4.1.5: Example 6.4.1.b, final control signal

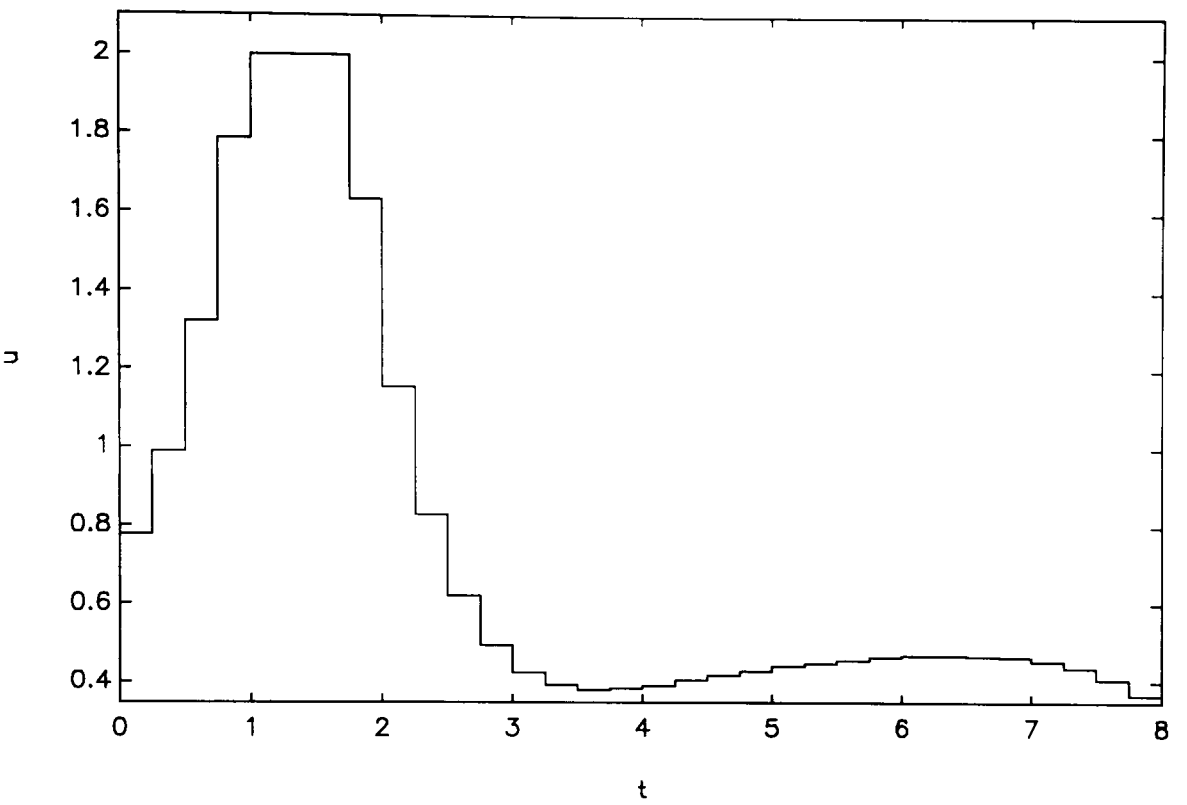

Figure 6.4.1.6: Example 6.4.1.e, final control signal

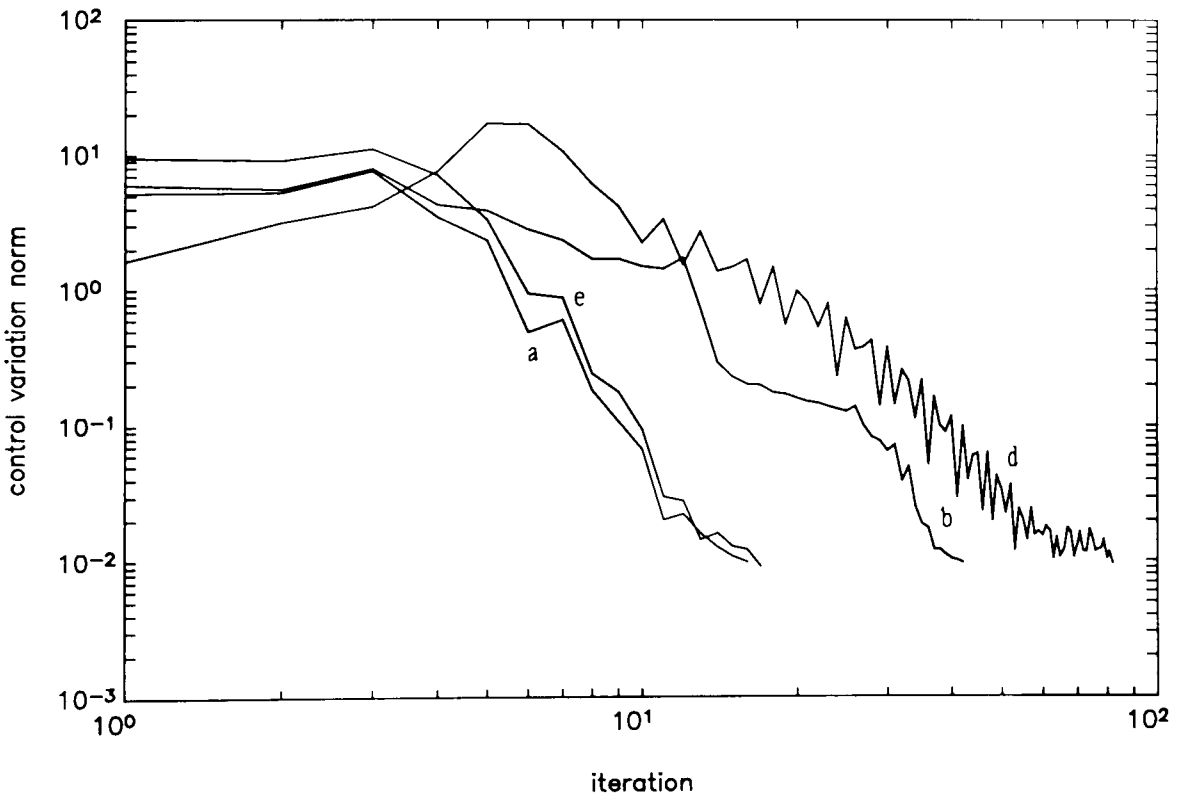

Figure 6.4.1.7: Example 6.4.1, control variation norm vs. iterations

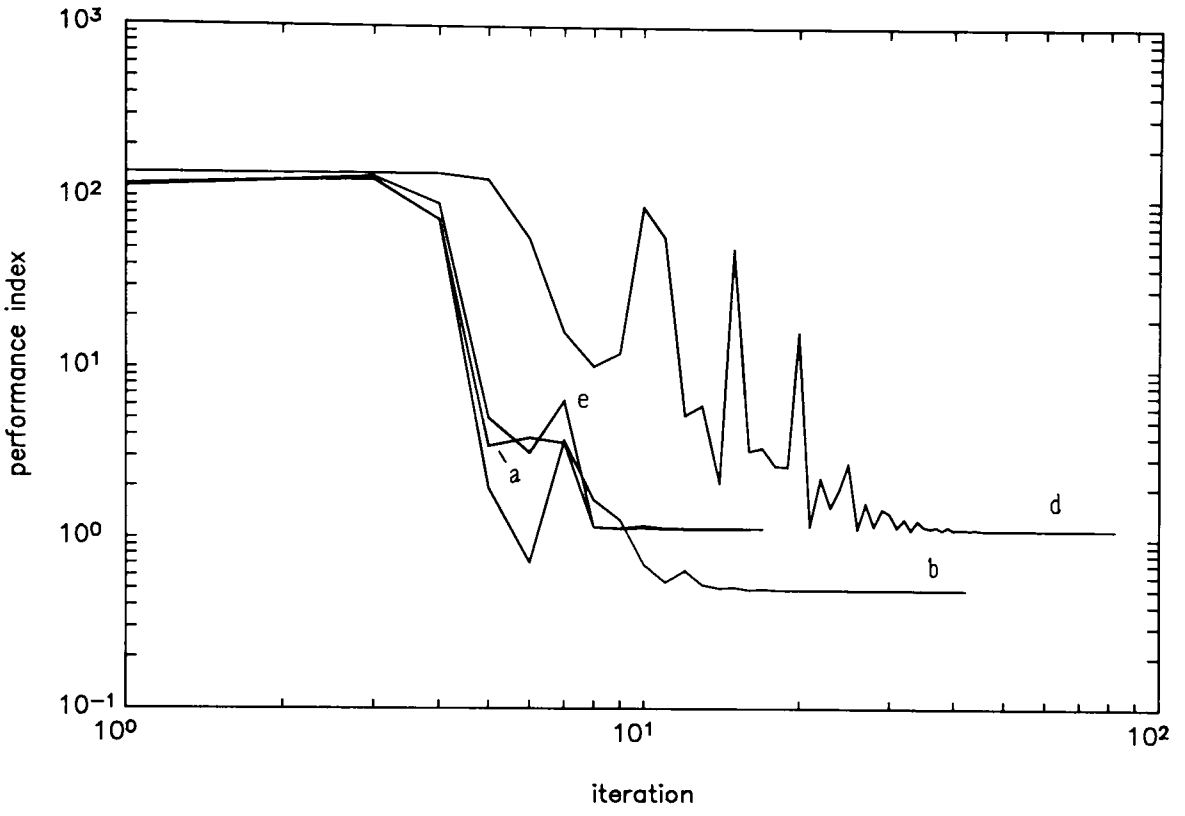

Figure 6.4.1.8: Example 6.4.1 performance index vs. iterations

It is observed that in case (a) the output signal seems to have reached a steady state with a certain off-set from the final set-point (see Figure 6.4.1.1), but there is a deviation from that value near the end of the optimisation horizon. Similarly, the control signal becomes zero at the end of the horizon (see Figure 6.4.1.4). This occurs because the final set-point value is not an equilibrium point of the system and the absolute value of the control signal is quadratically weighted at the end of the horizon.

If we weight the terminal set-point deviation (case e) but keep the quadratic absolute control weighting, then the output signal seems to have reached a steady state with a certain off-set from the final set-point (see Figure 6.4.1.3), but near the end of the time horizon the off-set disappears.

On the other hand, when we introduce quadratic incremental control weighting in case (b), the output signal reaches a steady-state value with zero off-set (see Figure 6.4.1.2). Similarly, the control signal reaches a (non zero) steady state value (see Figure 6.4.1.5). Notice, however, that the number of iterations for convergence increases when the control increments are used in the performance index (compare cases (b) and (a)).

It is observed in Table 6.4.1.2 that the use of the adaptive step size Runge-Kutta integrator (case (c)) increases the CPU time per iteration, with the advantages that the accuracy of integration is specified by the user and, furthermore, it is not necessary to specify an integration step. There is a minor difference in the number of iterations for convergence in cases (b) and (c), but the differences in the final computed values of states and control signals are not significant in this case. Although it is claimed that adaptive-step size integration reduces computational costs (Press *et al*, 1992), this is not applicable in this case due to the length of the integration range required by the exact discretization scheme (only 1 sample interval). It must be emphasized that the choice of the integration method is a function of the particular problem being solved and the availability of different ODE solvers in a general implementation is important (Strand, 1991).

Finally, the results indicate that the choice of the linearization point for computing the model-based dynamic matrices A and B affects the convergence behaviour of Algorithm 6.2.1. If we choose the linearization point as the origin $x = [0 \quad 0]^T$ (case (d)), rather that the initial state condition $x = x_0$ (cases (a), (b) and (e)), the convergence behaviour of the algorithm deteriorates significantly (see Figures 6.4.1.8 and 6.4.1.9 and compare cases (a) and (d)). This result will be used in Chapter 9, where the application of Algorithm 6.2.1 in a nonlinear predictive control scheme is described.

6.5 SUMMARY

A version of the discrete-time DISOPE algorithm which uses a set-point tracking formulation has been developed and implemented in software. The algorithm solves iteratively a model-based set-point tracking optimal control problem which has linear dynamics and a quadratic performance criterion, and the iterations converge to the solution of the nonlinear problem, denominated reality, of which the model-based problem is an approximation. The implementation has been tested with one nonlinear set-point tracking example. The results indicate that the real optimal solution is obtained in spite of the differences between the real and model-based problems. The influence of different factors have been investigated. Such factors are: the use of quadratic incremental control weighting in the performance index, the differential equation solver used, the linearization point for the model-based matrices,

and the use of terminal weighting of the set-point deviation. The algorithm as implemented is capable of handling nonlinear optimal control problems with non-quadratic performance indexes, multiple control inputs, nonlinear output functions and bounded controls.

CHAPTER 7

HANDLING OF STATE DEPENDENT CONSTRAINTS WITHIN THE DISOPE FRAMEWORK

In this chapter, the flexible structure of the DISOPE algorithm is exploited so as to handle optimal control problems with state dependent inequality constraints. The penalty relaxation technique is used, by which penalty functions are included in the real performance index. Provided the model-based problem is linear quadratic, the use of the penalty relaxation technique allows the solution of state constrained nonlinear optimal control problems by using standard linear quadratic methods in the model-based computations.

7.1 FORMULATION

It is well known that many real-life optimal control problems may have, in addition to constraints associated with the control variables, constraints associated with the trajectory followed by the state variables. This type of constraints is usually difficult to handle both from theoretical and computational points of view and sometimes their presence leads to ill-conditioning of the optimal control problem (Bryson and Ho, 1975; Teo *et al*, 1991). The ability of an optimal control algorithm to deal successfully and efficiently with this type of constraint is considered to be important.

Suppose that the state constrained real optimal control problem (SCROP) is defined as follows:

SCROP

$$\min_{\substack{u(k) \\ k\in[N_0,N_f-1]}} J_o^* = \sum_{k=N_0}^{N_f-1} L_o^*(x(k),u(k),k)$$

subject to

$$x(k+1) = f^*(x(k),u(k),k)$$
$$x(N_0) = x_0$$
$$\Psi(x(k)) \geq 0$$

where $f^* : \Re^n\times\Re^m\times\Re\rightarrow\Re^n$ represents a set of discrete-time state equations which describe the process with state $x(k)\in\Re^n$ and control input $u(k)\in\Re^m$, $L_o^* : \Re^n\times\Re^m\times\Re\rightarrow\Re$ is a discrete performance (or weighting) function and $\Psi : \Re^n\times\Re\rightarrow\Re^c$ is a set of state dependent inequality constraints to be satisfied during the fixed time interval of interest $[N_0,N_f-1]$.

The approach proposed in this chapter to handle state dependent inequality constraints within the DISOPE framework is the use of the penalty relaxation technique (Lin, 1993; Lin *et al*, 1989).

By using the penalty relaxation technique the original state constrained problem SCROP is transformed into an unconstrained optimal control problem such as ROP5 by adding penalty terms. We define the penalized discrete performance function as follows:

$$L^*(x(k),k) = L_o^*(x(k),k) + \rho\sum_{j=1}^{c}[P_\varepsilon(\Psi_j(x(k),k))]^2 \qquad \textbf{(7,1)}$$

where $\rho\in\Re$ is a large penalty factor and the smoothed function P_ε (Teo *et al*, 1991) is given by

$$P_\varepsilon(w) = \begin{cases} w & w \le -\varepsilon \\ -\dfrac{(w-\varepsilon)^2}{4\varepsilon} & -\varepsilon < w < \varepsilon \\ 0 & w \ge \varepsilon \end{cases} \qquad (7,2)$$

where $w \in \Re$ is a given argument and $\varepsilon \in \Re$ is a properly chosen small value. Each added penalty term becomes significant in the face of violations of the corresponding constraint along the time horizon, while its value is zero if the constraint is not violated at a given time.

Recall that DISOPE allows for model-reality differences between the model-based performance function L and the reality function L^*. We may use the penalized function L^* as reality, and then form the unconstrained ROP5 (which in this case would contain the penalty terms dependent on the state constraints and no terminal weighting, see Chapter 5) to be solved using DISOPE by iterating on a modified model-based problem such as MMOP5 (See Chapter 5). Provided the model-based problem is linear-quadratic, we may iterate using standard LQ methods (such as Procedure 5.3.1) for solving a path constrained optimal control problem.

The following DISOPE algorithm with state-dependent constraints, assuming convergence, achieves the solution of SCROP. The algorithm includes an strategy to decrease oscillations about the constrained optimum and improve convergence, which is based on increasing the control convexification factor r_1 whenever the performance index increases during the iterations. The state convexification factor r_2 is not changed during the iterations.

Algorithm 7.1.1: Discrete-time DISOPE algorithm with state-dependent constraints

---

Data — f, L, Ψ, x_0, N_0, N_f, ρ, ε^0, r_1, r_2, k_r, k_v, k_v, k_v and means for calculating f^* and L^*.

Step 0 — Compute or choose a nominal solution $u^0(k)$, $x^0(k)$ and $p^0(k)$. Set $i=0$, $v^0(k)=u^0(k)$, $k \in [N_0, N_f-1]$, $z^0(k)=x^0(k)$, $\hat{p}^0(k)=p^0(k)$, $k \in [N_0, N_f]$.

Step 1 Compute the real system state response $x_r^i(k)$ to $u^i(k)$. Then compute the penalized performance index J^{*i} given $x_r^i(k)$ and $u^i(k)$. If $J^{*i} > J^{*i-1}$, $i \geq 1$ increase the convexification factor r_1 as follows:

$$r_1 = k_r r_1, \quad k_r > 1$$

Step 2 Compute the parameters $\alpha^i(k)$, $\gamma^i(k)$ to satisfy (5,17). This is called the *parameter estimation step.*

Step 3 Compute the multipliers $\lambda^i(k)$ and $\beta^i(k)$ from (5,16).

Step 4 With specified $\alpha(k)$, $\gamma(k)$, $\lambda(k)$ and $\beta(k)$ solve the discrete-time modified model-based optimal control problem MMOP5 to obtain $u^{i+1}(k)$, $x^{i+1}(k)$ and $p^{i+1}(k)$. This is called the *system optimisation step.*

Step 5 This step tests convergence and updates the estimate for the optimal solution of ROP5. In order to provide a mechanism for regulating convergence, in addition to that given by convexification factors r_1, r_2, a simple relaxation method is employed to satisfy (5,18). This is:

$$\begin{aligned} v^{i+1}(k) &= v^i(k) + k_v(u^{i+1}(k) - v^i(k)) \\ z^{i+1}(k) &= z^i(k) + k_z(x^{i+1}(k) - z^i(k)) \\ \hat{p}^{i+1}(k) &= \hat{p}^i(k) + k_p(p^{i+1}(k) - \hat{p}^i(k)) \end{aligned} \qquad \textbf{(7,3)}$$

where k_v, k_z and $k_p \in (0,1]$ are scalar gains. If $v^{i+1}(k) = v^i(k)$, $k \in [N_0, N_f - 1]$ within a given tolerance stop, else set $i = i+1$ and continue from step 1.

7.2 SIMULATION EXAMPLE

The following example was run on a IBM compatible 486DX based machine with 33 MHz speed and is based on a C++ implementation of Algorithm 7.1.1.

Example 7.2.1: Linear system with time varying state constraint

This non-trivial example has been used by several authors to test state constrained optimal control algorithms (Teo *et al*, 1991). The problem may be expressed as follows:

$$\min_{u(k)} \quad J_o^* = \sum_{k=0}^{19} \frac{1}{2}(x_1(k)^2 + x_2(k)^2 + 0.005u(k)^2)$$

subject to

$$x(k+1) = x(k) + \int_{t_k}^{t_{k+1}} f_c(x(\tau), u(k), k)\, d\tau$$

$$x(0) = [0 \quad -1]^T$$

$$\Psi(x(k), k) = 8(0.05k - 0.5)^2 - 0.5 - x_2(k) \geq 0$$

where the continuous dynamics represented by $\dot{x} = f_c(x, u, t)$ are linear and given by:

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

and the sampling interval is $T = 0.05$

In order to find the solution of the above described state constrained optimal control problem we use Algorithm 7.1.1 with a real optimal control problem given by:

ROP

$$\min_{u(k)} \quad J^* = \sum_{k=0}^{19} \left[\frac{1}{2}(x_1(k)^2 + x_2(k)^2 + 0.005u(k)^2) + \rho P_e(\Psi(x(k), k))^2 \right]$$

subject to

$$x(k+1) = x(k) + \int_{t_k}^{t_{k+1}} f_c(x(\tau), u(k), k)\, d\tau$$

$$x(0) = [0 \quad -1]^T$$

The model-based dynamics are based on a zero-order hold discretization of the continuous dynamics (see, for instance, Åström and Wittenmark (1990)). The model-based problem given by:

MOP:

$$\min_{u(k)} \sum_{k=0}^{19} \tfrac{1}{2}(x_1(k)^2 + x_2(k)^2 + 0.005u(k)^2 + 2\gamma(k))$$

subject to

$$x(k+1) = \begin{bmatrix} 1.0 & 0.048771 \\ 0.0 & 0.951229 \end{bmatrix} x(k) + \begin{bmatrix} 0.001229 \\ 0.048771 \end{bmatrix} u(k) + \alpha(k)$$

$$x(0) = [0 \quad -1]^T$$

The nominal solution consisted of the solution of MOP with $\alpha(t) = 0$, $\gamma(t) = 0$. Table 7.2.1.1 shows the tuning parameters used for the solution. Table 7.2.1.2 shows the algorithm's performance. Figures 7.2.1.1 and 7.2.1.2 show the final state and control trajectories. Figures 7.2.1.3 and 7.2.1.4 show the convergence behaviour of the algorithm.

ε^0	r_1^0	r_2	k_r	ε_v	$k_v = k_z = k_p$	ρ
0.00001	2.0	0	2.0	0.01	1	100

Table 7.2.1.1: Tuning parameters for example 7.2.1

No. of iterations	Maximum constraint violation	Final performance index
50	0.0064	2.0131

Table 7.2.1.2: Performances for example 7.2.1

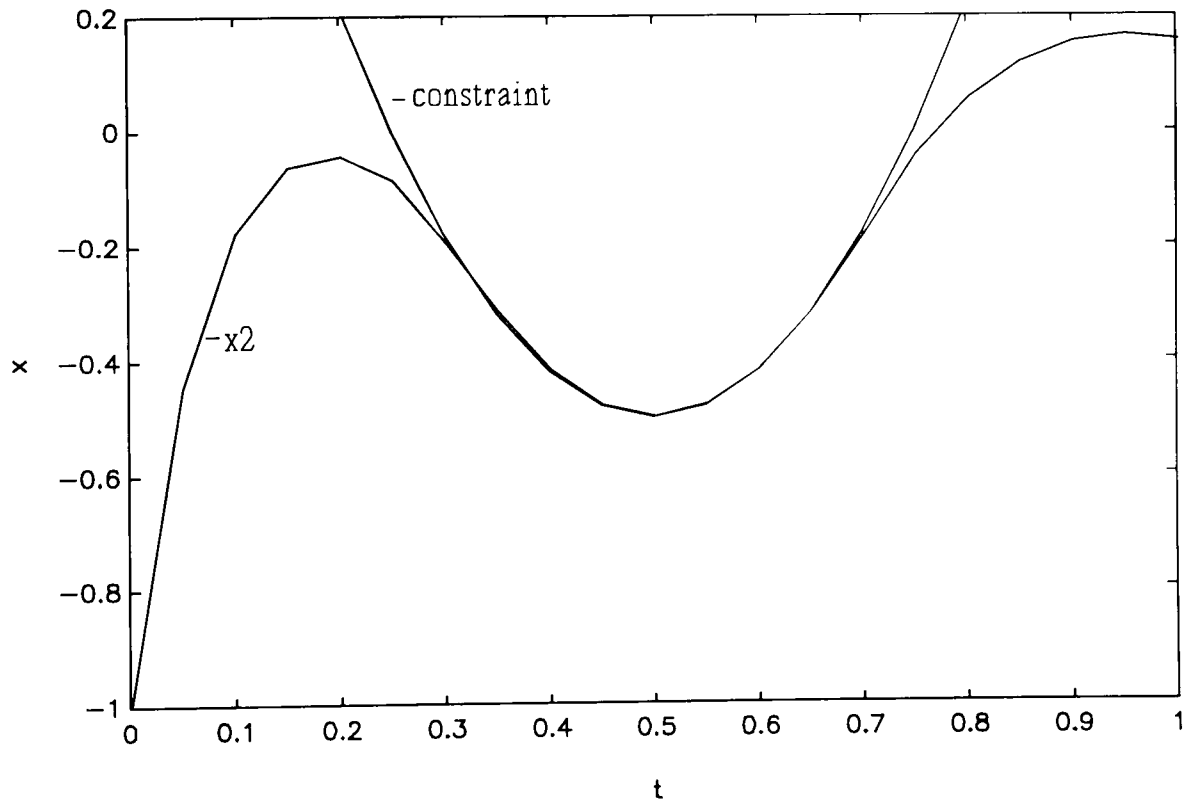

Figure 7.2.1.1: Example 7.2.1, constrained state and constraint

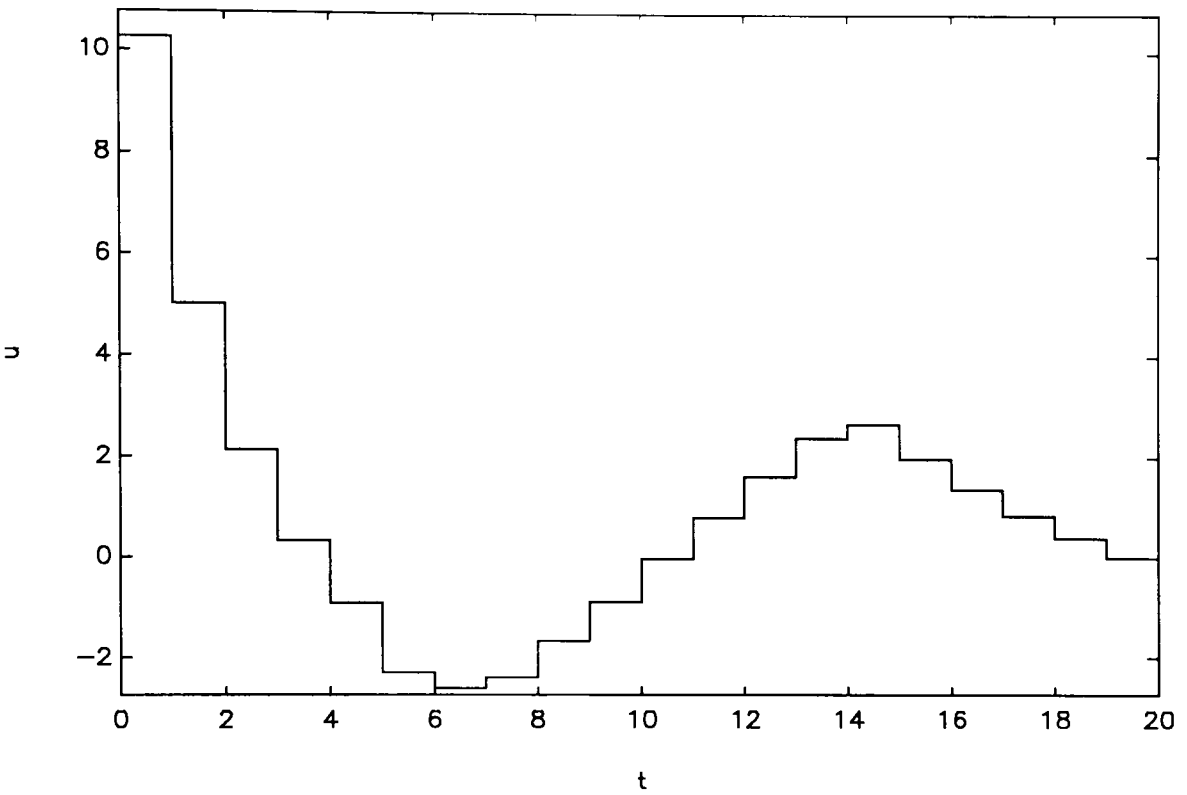

Figure 7.2.1.2: Example 7.2.1, final control signal

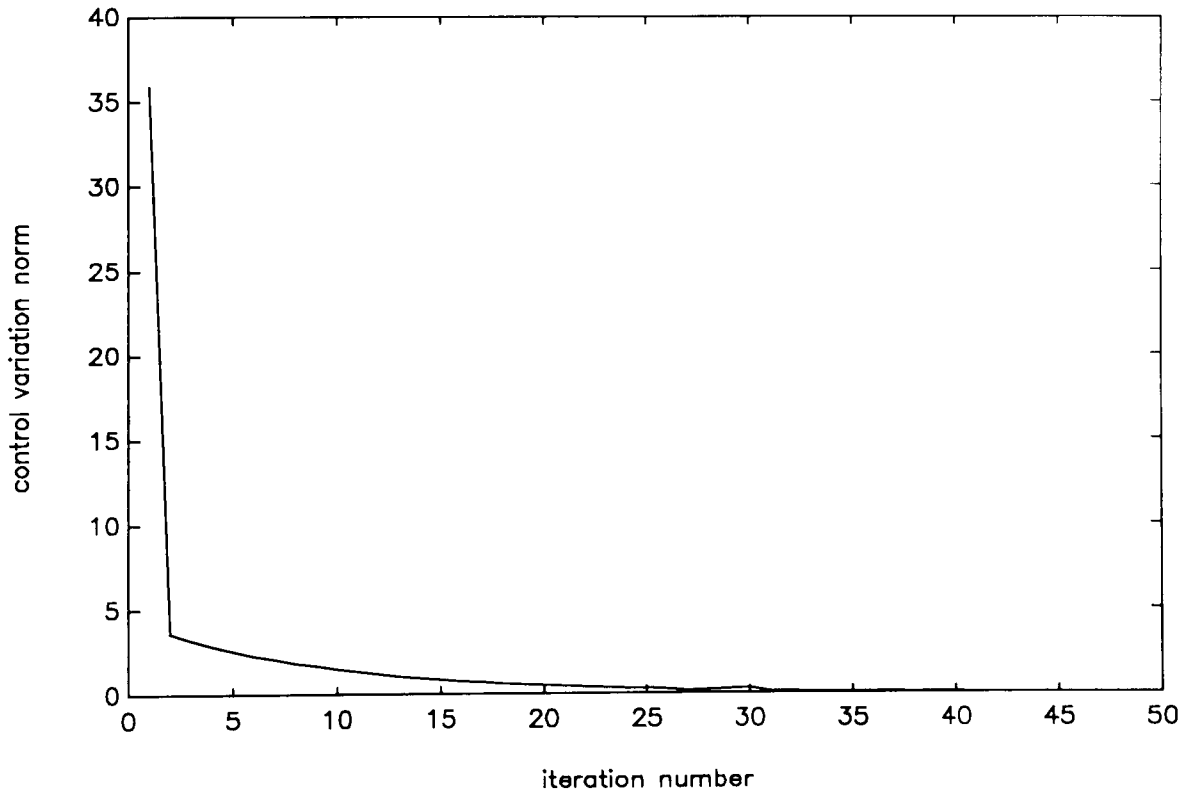

Figure 7.2.1.3: Example 7.2.1, control variation norm vs. iterations

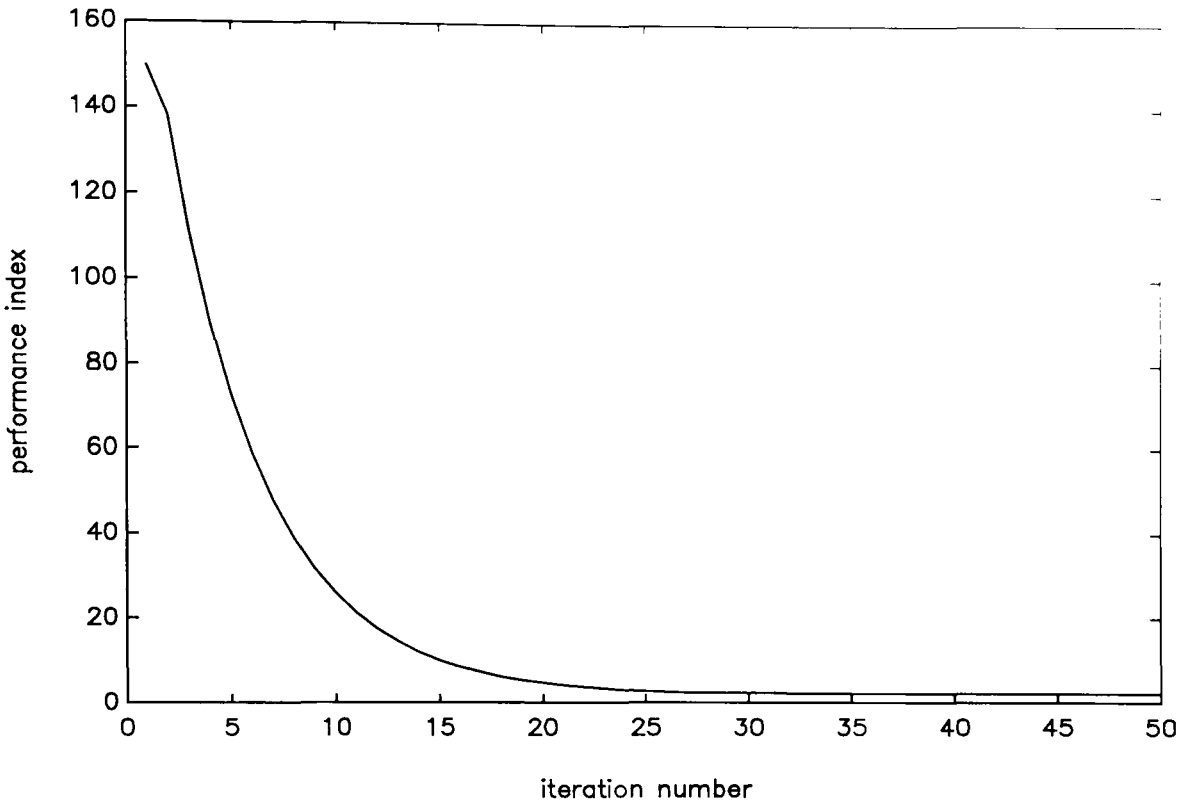

Figure 7.2.1.4: Example 7.2.1 penalized performance index vs.iterations

7.3 SUMMARY

An approach for handling state dependent inequality constraints within the DISOPE framework has been proposed. The method is based on the use of the penalty relaxation technique. Using the inherent flexibility of the DISOPE approach, penalty terms are added to the original performance index and, provided the model-based problem is linear-quadratic, the state constrained problem is solved by using iterative linear-quadratic methods.

A DISOPE algorithm with state-dependent constraints has been proposed which includes a strategy to decrease oscillations about the constrained optimum and improve convergence. The technique has been successfully tested with a simulation example.

CHAPTER 8

APPLICATION OF DISOPE IN THE OPTIMISATION OF BATCH PROCESSES

In this chapter, a DISOPE algorithm is designed for the optimisation of batch processes. The algorithm achieves the real dynamic optimum of the batch process in spite of deficiencies in the mathematical model used for the computations. A modification of the discrete-time DISOPE algorithm is introduced in order to integrate the iterations of DISOPE with the batchwise operation of the process, in such a way that every batch cycle corresponds with one iteration of the algorithm. The use of the shadow model concept is proposed in order to deal with the problem of accurate state and dynamic derivative estimation. The approach is illustrated with one simulation example.

8.1 BATCH PROCESSES

Industrial processes can be classified as continuous, discrete, or batch. How a particular process is classified depends on the way the output product is yielded: either in a continuous flow or in discrete batches or quantities. Batch processes are common in small scale processing of chemical products with high unit cost.

A process is considered to be batch if it consists of a sequence of steps or phases that must be carried out in a defined order. The culmination of this succession of steps creates a finite amount of finished product. The sequence needs to be repeated to produce additional amounts of product. Batch processes present interesting control problems due to their inherent dynamic nature.

A typical batch cycle can be described as follows (Rijnsdorp, 1991):

Step 1	Transportation and storage of raw materials
Step 2	Preparation of mixtures
Step 3	Initial charging into vessesls
Step 4	Transfer to initial conditions

Step 5 Transformation

Step 6 Transfer to final conditions

Step 7 Emptying and cleaning

Batch processes have optimum conditions that will yield the maximum product with minimum time and cost. The degrees of freedom for the determination of such optimum conditions are often a combination of the initial conditions, the set-point profile used during the transformation phase and the time allowed for this phase.

During the transformation phase some process variables have to follow a certain function of time so as to obtain good operating conditions. This is achieved by manipulating controller set-points during that period.

Procedures for determining acceptable set-point profiles include trial and error, previous experience, and the use of dynamic optimisation.

Optimal set-point profiles obtained by the use of optimal control theory based on a dynamic model of the batch process will only be optimal for the specific model and parameter values used in the optimisation. This mismatch between the model and the actual plant behaviour may result in sub-optimal performance when the model-based optimal profile is applied to the real process.

8.2 DISOPE APPROACH

We have seen in the previous chapters of this thesis that the DISOPE algorithm has been designed so that it handles the model-reality differences between the model used for the computations and the real plant in such a way that the correct dynamic optimum is achieved in a sequential way in spite of the deficiencies in the model.

The idea of applying DISOPE to the dynamic optimisation of batch processes arises naturally from knowledge on the structure of the algorithm and the basic principles on which it was originally developed.

An attempt to handle model-reality differences in batch process dynamic optimisation have been presented by Zafiriou and Zhu (1989), who use a dynamic version of the Two-step Method and, as a result of not dealing with the differences in the first order derivatives of the real process and the model used, is suboptimal.

It is possible to integrate the iterations of DISOPE towards the dynamic optimum with the batchwise operation of the process. This is achieved by introducing a modification on the basic algorithm such that the control profile obtained from model-based optimisation at each iteration is applied to the real process at every transformation phase. As has been discussed in Chapter 5 it is convenient to formulate the algorithm in discrete-time so that it is suitable for computer-control implementation. Then, assuming that the transformation time is fixed, the discrete-time DISOPE algorithm for batch processes is described as follows:

Algorithm 8.2: Discrete-time DISOPE algorithm for batch processes

---

Data — f, L, φ, x_0, N_0, N_f, and means for calculating f^* and L^*.

Step 0 — Compute or choose a nominal solution $v^0(k)$, and $\hat{p}^0(k)$. Set i=0,

Step 1 — During the transformation phase, apply the control profile $v^i(k)$ to the batch plant. Obtain the corresponding state response $z^i(k)$, $k\in[N_0,N_f]$ and dynamic derivatives $\partial f^*/\partial z$ and $\partial f^*/\partial v, k\in[N_0,N_f-1]$ (See Section 8.3).

Step 2 — Compute the parameters $\alpha^i(k)$, $\gamma^i(k)$ to satisfy (5,17). This is called the *parameter estimation step*. This may be done simultaneously with step 1

Step 4 — Compute the multipliers $\lambda^i(k)$ and $\beta^i(k)$ from (5,16). This may also be carried out simultaneously with step 1.

Step 5 — With specified $\alpha(k)$, $\gamma(k)$, $\lambda(k)$ and $\beta(k)$ solve the discrete-time modified model-based optimal control problem MMOP5 to obtain $u^{i+1}(k)$, $x^{i+1}(k)$ and $p^{i+1}(k)$. This is called the *system optimisation step*.

Step 6 — This step tests convergence and updates the estimate for the optimal solution of ROP5. In order to provide a mechanism for regulating convergence, a simple relaxation method is employed to satisfy (5,18). This is:

$$v^{i+1}(k) = v^{i}(k) + k_v(u^{i+1}(k) - v^{i}(k))$$
$$\hat{p}^{i+1}(k) = \hat{p}^{i}(k) + k_p(p^{i+1}(k) - \hat{p}^{i}(k)) \qquad \textbf{(8,1)}$$

where k_v, and $k_p \in (0,1]$ are scalar gains. If $v^{i+1}(k) = v^{i}(k)$, $k \in [N_0, N_f - 1]$ within a given tolerance stop, else set $i=i+1$ and continue from step 1.

It is important to remark that Algorithm 8.2 requires precise knowledge of the values of the state variables of the process during the transformation phase and, furthermore, the values of the derivative matrices $\partial f^*/\partial z$ and $\partial f^*/\partial v$ are also required during that period. An approach for obtaining reliable values of these variables is given below.

Notice that the algorithm is adaptive, since changes in the dynamics of the system or external disturbances may be detected in the estimation of the state variables and hence by the algorithm itself, which would lead the system to the correct optimum.

Recall that the initial state conditions and transformation time are assumed fixed. An optimisation algorithm located at an upper level (relative to DISOPE) might be used in order to optimise the process with respect to such parameters.

8.3 THE SHADOW MODEL CONCEPT

It is proposed to use the shadow model concept, introduced by Griffiths (1993), for dealing with the problem of accurate measurement of state information from the process, required by the DISOPE algorithm.

The increase in relatively cheap computer power with high performance, together with advances in software tools, have originated significant developments in the area of process simulation. The real time *shadow model* is a highly rigorous mathematical model of the process based on first-principle physical laws which is run in parallel with the process. The use of very reliable data is very important for

the implementation of such an accurate model and hence data validation and reconciliation schemes need to be used in order to increase the validity and accuracy of the sometimes uncertain and noisy process measurements. This model provides control systems and operators with variables and parameters not measured in the real plant. This innovative concept has already been implemented in industrial scale problems as reported by Griffiths (1993). Depending on the complexity of the process the shadow model might consists on several hundreds of differential and algebraic equations, while the real process order might be considered very large indeed. For the practical application of DISOPE in the optimisation of batch processes it is then necessary to *choose* a number of relevant state variables to be measured from the shadow model. These relevant states, chosen by an expert in the process, should be a good representation of the dynamics of the process.

8.4 ESTIMATION OF DERIVATIVES

The accurate estimation of derivatives is also very important for the success of the application of DISOPE in batch process optimisation. A model of identical structure and parameters as the shadow model, but which is operated a few times faster than real time, might be used to obtain the derivatives by predicting the process response into the next sampling time based on perturbations over current state and control data. Then finite differences may be used to obtain the derivatives.

8.5 CHOICE OF THE DYNAMIC MODEL FOR THE OPTIMISATION STEP

It is also necessary to choose the dynamic model to be used to define the model based optimal control problem. The order of this model should be equal to the number of relevant states measured from the shadow model. It might be a nonlinear but simplified model based on physical laws. Alternatively, a linear dynamic model can be constructed from the derivative estimates. The number of DISOPE iterations, or in other words, the number of batches required to achieve the dynamic optimum will depend on the choice of this model.

8.6 CHOICE OF THE ALGORITHM FOR SOLVING THE MODEL-BASED PROBLEM

An algorithm has to be used to solve the modified model-based problem at every iteration. If a simplified nonlinear model is available, then the algorithm has to be able so solve a nonlinear optimal control problem. On the other hand, if no other model is available then a linear model can be constructed from the derivative estimates as follows:

$$x(k+1) = \hat{A}x(k) + \hat{B}u(k) + \alpha(k) \qquad \textbf{(8,2)}$$

where $\hat{A}$ and $\hat{B}$ are the estimated derivatives at, say, the initial batch at time N_0. If, in addition to having linear model-based dynamics, the model-based performance index L is chosen as a quadratic approximation of L^* then, as we have seen in the previous chapters, standard LQ methods, such as Procedure 5.3.1, may be used for solving the modified model-based problem. If so, control magnitude constraints may be handled in a straightforward manner by saturating the controls at the bounds when they are applied to the plant (see Section 3.3). Notice that if control magnitude constraints are handled at the model-based level by the optimisation algorithm, an improvement in the convergence behaviour of DISOPE is possible (See Chapter 3).

If a simplified nonlinear dynamic model of the process is available, its use might be an advantage from the point of view of the number of iterations necessary for convergence, since they might be lower than by using a linear model, provided the nonlinear model is a better approximation to reality (the batch process) than the simple linear dynamic model. In this case a nonlinear optimal control algorithm has to be used for solving the modified model-based problem. For instance, Algorithm 5.3.3 (DISOPE with LQ model based problem), applied to the (nonlinear) model-based problem playing the role of "reality", is a possible candidate.

8.7 SIMULATION EXAMPLE

Example 8.7.1: Batch chemical reactor

This example, taken from (Ray, 1981), consists of the dynamic optimisation of a batch chemical reactor in which it is assumed that the reaction temperature can be controlled exactly. We wish to carry out the following reaction in the reactor:

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

The objective is to find the optimal temperature profile that, for a fixed batch time of 1 hour, will maximize the production of the intermediate B. As pointed out by Ray (1981) the scheme given by the reaction considered here is of practical importance in a number of chemical processing operations, including the oxidation of hydrocarbons or the chlorination of aromatics. Notice that we wish to maximize the production of an intermediate product and hence it is necessary to prevent the reaction from going to completion.

The purpose of this example is to illustrate the use of DISOPE for the optimisation of batch processes. To pursue this objective we will simulate a very simple nonlinear "shadow model" of the process, which we will assume to be in ideal correspondence with the real process and from which we will measure the state and derivative information which DISOPE requires for the iterations. Recall that in the scheme proposed in this chapter, every iteration of DISOPE corresponds with a batch cycle.

The dynamics of the shadow model are given by:

$$\begin{aligned}
\dot{x}_1 &= -k_1(u(t))x_1(t)^2 \\
\dot{x}_2 &= k_1(u(t))x_1(t)^2 - k_2(u(t))x_2(t) \\
x(t=0) &= [1.0 \quad 0]^T
\end{aligned}$$

where:

$x_1 =$ concentration of A

$x_2 =$ concentration of B

$u =$ reaction temperature (K)

$k_1 =$ $A_1 e^{-\frac{E_1}{Ru}}$

$k_2 =$ $A_2 e^{-\frac{E_2}{Ru}}$

$A_1 =$ 4000.0 L/(mol)(s)

$A_2 =$ 620000 1/(s)

$E_1 =$ 5000 cal/(g)(mol)

$E_2 =$ 10000 cal/(g)(mol)

The control variable u is bounded between upper and lower levels $298\,\mathrm{K} \le u \le 398\,\mathrm{K}$. The control signal is constant between sampling times and the length of the sampling interval is $T_s = 3$ min. States and derivatives are measured with the same sampling rate as the control signal is discretized. The shadow model was integrated between sampling times by using an adaptive 5th order Runge-Kutta integrator (Press *et al*, 1992). The derivatives required by DISOPE were computed by using a central difference formula. The tolerance specified for convergence was $\varepsilon_v = 2.0\,\mathrm{K}$ and the relaxation gains used in the iterations were $k_v = k_p = 1$. Notice that the initial conditions are fixed at every batch.

The real performance index to be minimized reflects our desire to maximize the production of B and is given by:

$$J^* = -x_2(N_f)$$

where $N_f = 20$ corresponds with the transformation time of one hour, taking $N_0 = 0$ as the initial time index of every batch or iteration. The model-based problem was taken as linear dynamics and a quadratic approximation of the performance index as follows:

MOP

$$\min_{u(k)} \quad J_m = -x_2(N_f) + \sum_{k=0}^{N_f-1} \quad 0.00025u(k)^2 + \alpha(k)$$

s.t.

$$x(k+1) = \begin{bmatrix} 0.919217 & 0 \\ 0.080756 & 0.998564 \end{bmatrix} x(k) + \begin{bmatrix} -0.000518 \\ 0.000551 \end{bmatrix} u(k) + \alpha(k)$$

The modified model-based problem was solved by using Procedure 5.3.1, where the recursion of h was started from the terminal condition $h(N_f) = [0 \quad -1]^T$, so as to satisfy the boundary condition (5,7). The initial guesses of control and costate trajectories to start the iterations were $v^0(k) = 298\,\mathrm{K}$, $k \in [0,19]$, $\hat{p}^0(k) = [0 \quad 0]^T$, $k \in [0,20]$, respectively. Table 8.7.1 shows the algorithm's performance in terms of number of iterations for convergence, the concentrations of product B for the initial control guess $x_2(1\,\mathrm{h})^0$, and for the final batch $x_2(1\,\mathrm{h})^{20}$. Figures 8.7.1.1 and 8.7.1.2 shows the final concentration and temperature profiles. Figure 8.7.1.3 shows the evolution of the control variation norm between iterates. Figure 8.7.1.4 shows the evolution of the terminal concentration of product B from the initial batch to the end of the iterations of DISOPE. Notice that $x_2(1\,\mathrm{h})$ grows monotonically with the iterations.

No. of iterations	$x_2(1\,\mathrm{h})^0$	$x_2(1\,\mathrm{h})^{20}$
20	0.46	0.61

Table 8.7.1: Performances for example 8.7.1

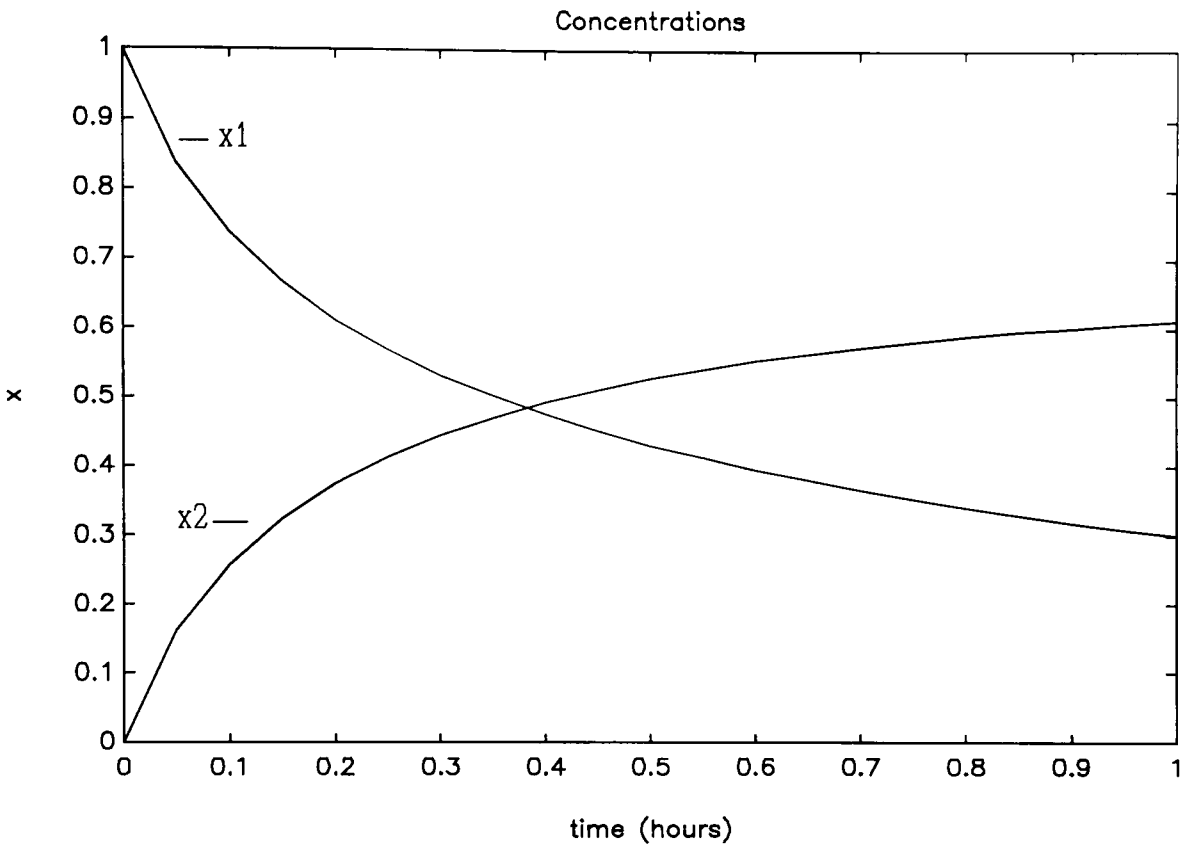

Figure 8.7.1.1: Example 8.7.1, final concentration responses

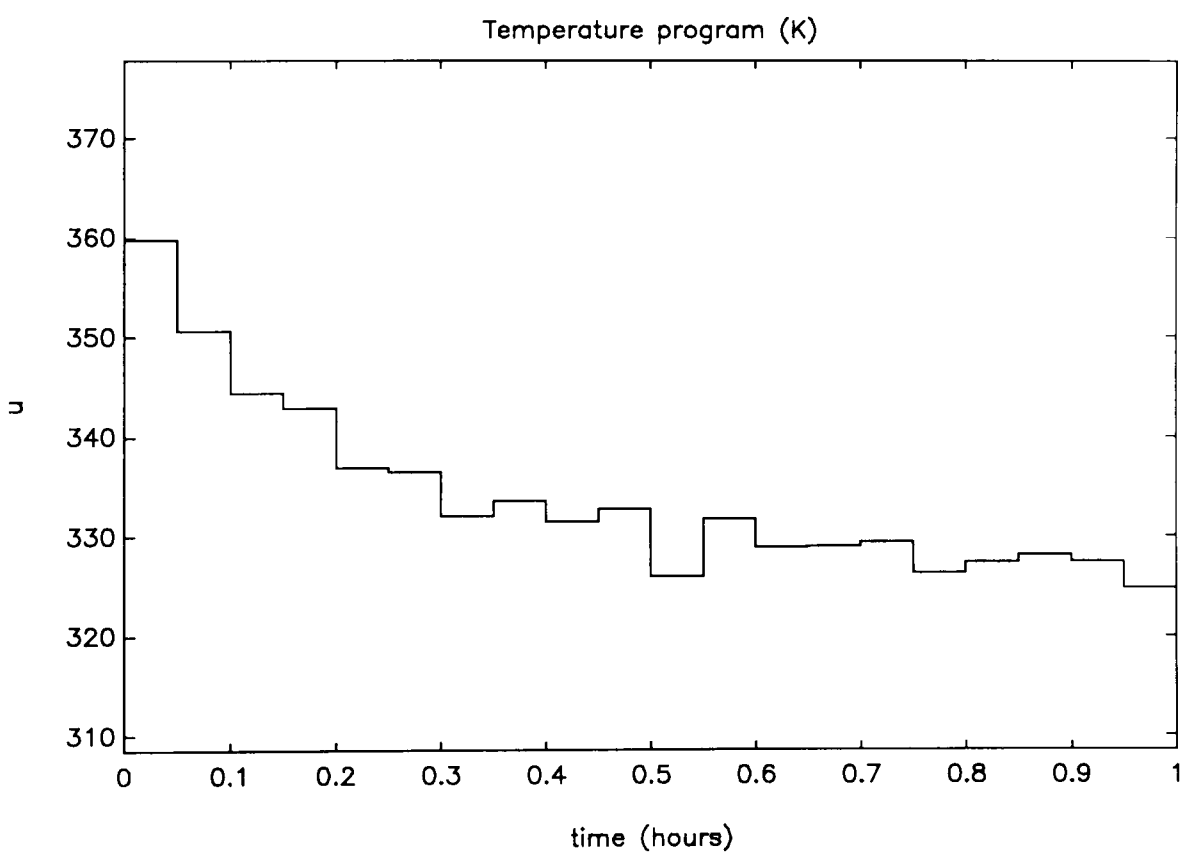

Figure 8.7.1.2: Example 8.7.1, final temperature profile

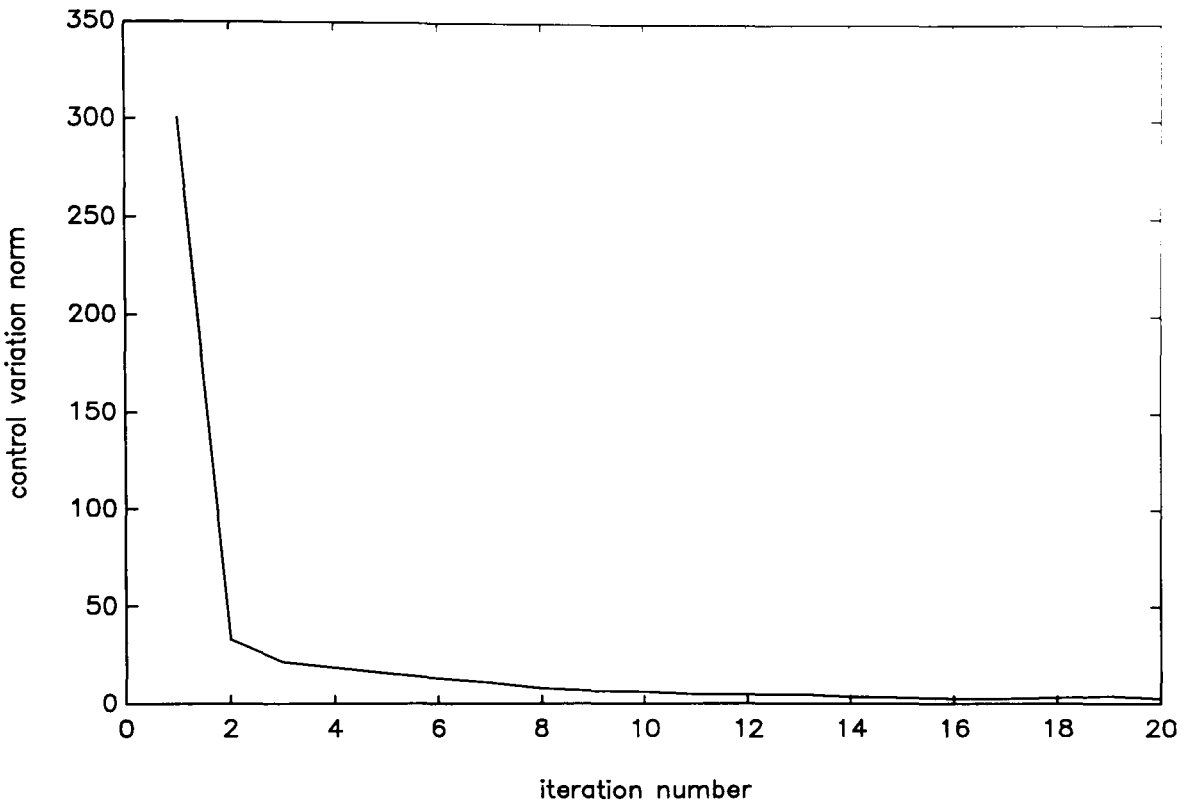

Figure 8.7.1.3: Example 8.7.1, control variation norm vs. iterations

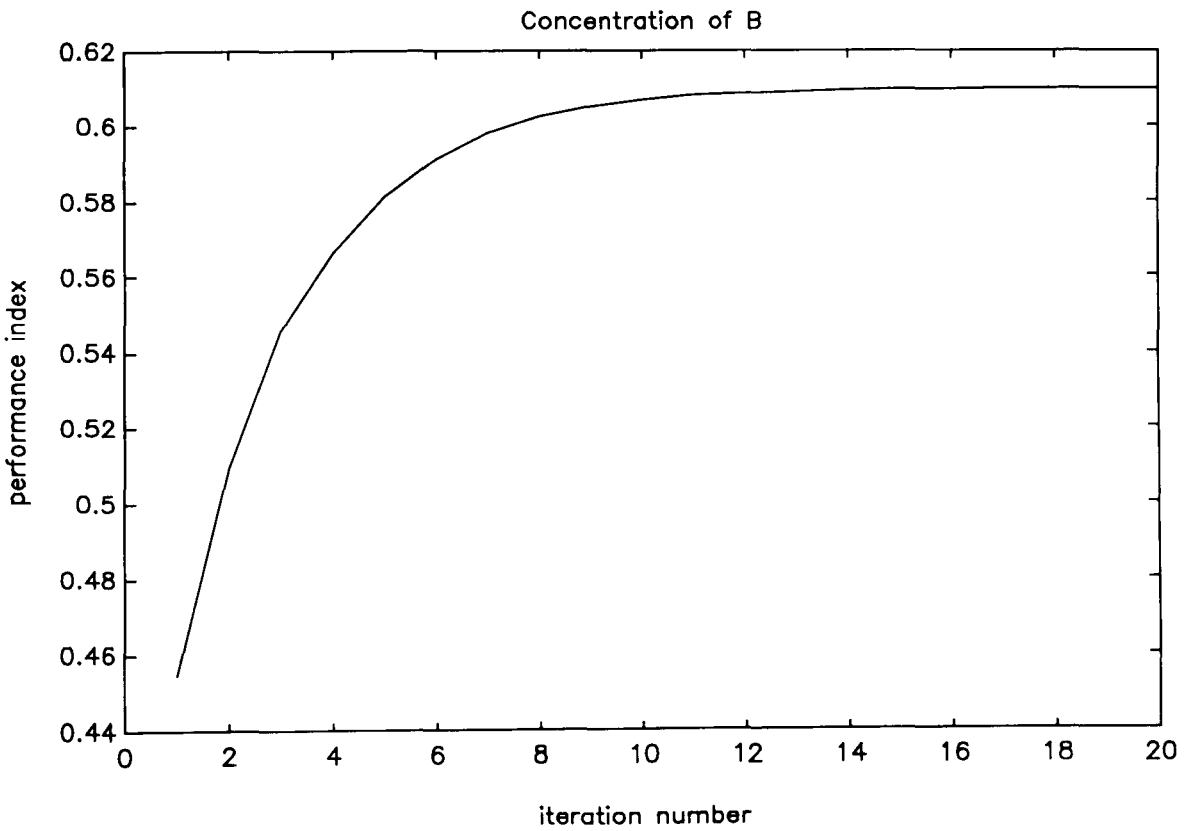

Figure 8.7.1.4: Example 8.7.1 performance index vs. iterations

8.8 SUMMARY

The application of the DISOPE algorithm has been proposed for the dynamic optimisation of batch processes. A modification of the discrete-time DISOPE algorithm has been introduced in order to integrate the iterations of DISOPE with the batchwise operation of the process. The DISOPE algorithm for batch processes achieves the real dynamic optimum of the batch process in spite of deficiencies in the mathematical model used for the computations. The shadow model concept has been used in order to deal with the problem of accurate state and dynamic derivative estimation, required by the DISOPE algorithm. The approach has been illustrated with one simulation example consisting of the dynamic optimisation of a batch chemical reactor.

CHAPTER 9

APPLICATION OF DISOPE IN NONLINEAR PREDICTIVE CONTROL

In this chapter, a nonlinear predictive controller based on state space models of the controlled plant has been developed and implemented in software. The receding horizon long range prediction and dynamic optimization is carried out at every sampling time by using the DISOPE algorithm. States and parameters are estimated from the possibly noisy output measurements by using an Extended Kalman Filter. The technique has been tested with simulation examples and its performance has been evaluated. The controller is flexible and is able to handle input bound and state dependent constraints. The research work presented in this chapter is also described in (Becerra, 1993e).

9.1 PREDICTIVE CONTROL

Industrial processes have been typically operated using linear controllers. However, most processes are inherently nonlinear and linear controllers yield satisfactory action only if the process operating point does not change significantly.

Process dynamic characteristic may change dramatically if large disturbances occur or after a significant set-point change. Additionally, batch processes work over very different operating ranges, which makes batch process controllers difficult to tune.

Predictive Control techniques, based on linear models of the plant, have been developed and widely applied in process plants in the last two decades. Predictive control belongs to the class of *model-based* controller design concepts, because a model of the plant is used to compute the control action. The reasons for their acceptation are many, but the main ones are: they are easy to tune; they may handle systematically process constraints, multivariable processes and time delays;

knowledge of future set-point changes can be included; their computational requirements are modest (Soeterboek, 1992; Garcia *et al*, 1989). However, linear predictive controllers are only locally valid. Even if the linear input-output model is updated through an estimation scheme, its long-range prediction capabilities may be poor in the presence of nonlinearities if such predictions lie out of the (local) linear region in the state space.

There have been growing interest in the last few years on extending predictive control concepts to take into account process nonlinearities. Some of those schemes have been presented in the literature (Sistu and Bequette, 1992; Gattu and Zafiriou, 1992; Balchen *et al*, 1992). A common characteristic is their increased computational load. Most of them are based on nonlinear state space models of the process and some of them include state and parameter estimation to give robustness, adaptability, and stability to the controller. Therefore, it is assumed in this chapter that a nonlinear model of the process is available for prediction purposes. If the dynamic optimisation scheme allows for general performance index specifications, economic criteria can be included (Balchen *et al*, 1992). If the control objectives pursued are to regulate the plant about the current set-points in spite of external disturbances and to track changes in such set-points in an appropriate way, then a nonlinear predictive controller has to solve a tracking optimal control problem in a receding-horizon fashion.

In Chapter 6 the discrete time formulation of DISOPE is used to solve nonlinear optimal set-point tracking problems. This algorithm is appropriate for use in a nonlinear predictive control scheme. Experimental results indicate that if the model-based problem is chosen as a linearization of the nonlinear system about the initial state condition, then convergence may be achieved faster than if other linearization points are chosen. This suggest that if DISOPE is used in a nonlinear predictive control scheme, then the current state estimate, which is the initial condition for the receding horizon optimization, may be used as a linearization point to obtain the linear model based dynamics.

In this chapter, a nonlinear predictive controller is developed and implemented in software. The receding horizon dynamic optimisation is carried out at every sampling time by the DISOPE algorithm. State estimation from possibly noisy measurements is carried out by using an Extended Kalman Filter (EKF). The controller may have adaptive features because the EKF can be used to continuously

estimate relevant uncertain parameters. The controller is able to handle input, output and internal state magnitude constraints. Simulation examples illustrating the capabilities of the controller are presented.

Predictive controllers are usually formulated in discrete time. The way a predictive controller operates is illustrated by Figure 9.2.1. Based on present and past output measurements and control action, the controller predicts the future output response (using an *internal model* of the plant) and chooses the best future control sequence by minimizing a given performance index. The plant may be required to follow a desired output trajectory. Then, only the first element in the computed control sequence is applied to the plant. Such calculations are performed at every sampling interval in a *receding-horizon* fashion. The *control horizon* is the number of future control steps which are computed. The *prediction horizon* is the number of output samples which are predicted.

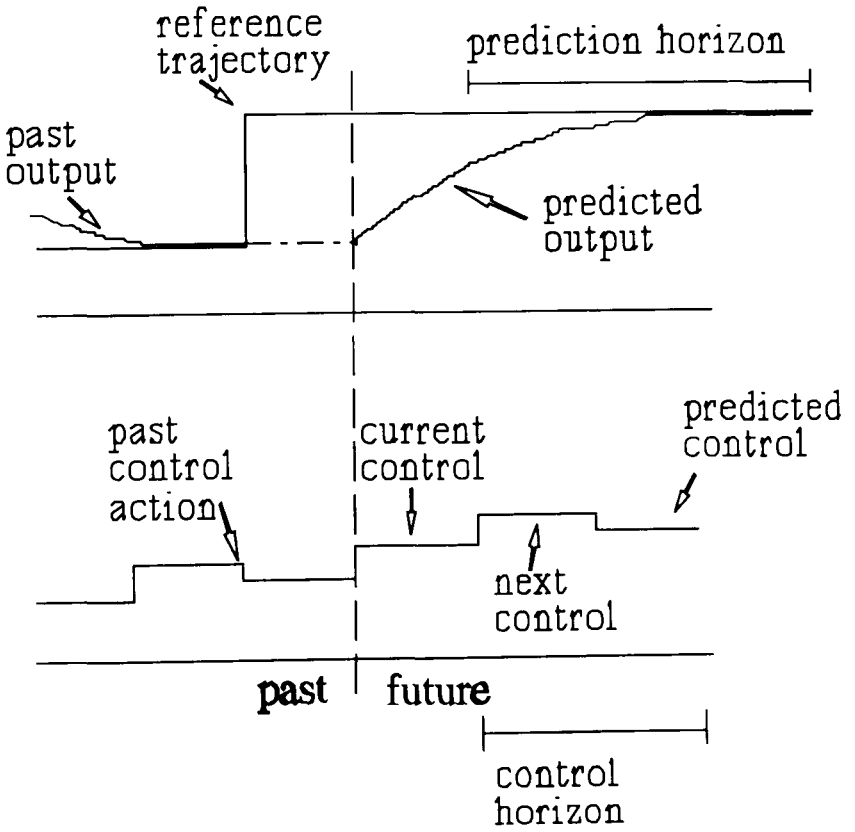

Figure 9.2.1: Predictive control approach

9.2 NONLINEAR PREDICTIVE CONTROL

9.2.1 Formulation

When the plant is nonlinear, appropriate models can be defined in the state space. Thus, the receding horizon optimisation to be solved at every sampling time can be formulated as follows:

Plant dynamic model:

$$\frac{dx}{dt'} = F^*(x(t'),u(t'),\hat{\theta},t') \qquad \textbf{(9,1)}$$

$$y(t') = \zeta(x(t'))$$

Receding Horizon Optimisation

$$\min_{\substack{u(j) \\ k\in[k+1,k+N_c]}} J_r(u,t) = \varphi(x(t+1+N_c)) + \sum_{k=t+1}^{t+N_c} L^*(x(k),u(k),\Delta u(k),y_c(k),r_0(k),k)$$

subject to

$$x(k+1) = f^*(x(k),u(k),\hat{\theta},k) = \int_{t_k}^{t_{k+1}} F^*(x(\tau),u(k),\hat{\theta},\tau)\,d\tau$$

$$x(t+1) = \hat{x}(t+1\,|\,t)$$

$$y(k) = \zeta(x(k),k)$$

$$u_{\min} \le u \le u_{\max}$$

$$\Psi(x(k)) \le 0$$

where

u	=	m-dimensional control vector
Δu	=	m-dimensional control increment vector ($=u(k)-u(k-1)$)
x	=	n-dimensional state vector
$\hat{\theta}$	=	n_θ-dimensional vector of estimated parameters

y	=	n_0-dimensional model-based output prediction vector
y_m	=	n_0-dimensional vector of measured outputs
y_c	=	n_0-dimensional vector of corrected output predictions
r_0	=	n_0-dimensional reference trajectory vector
$u_{\min}, u_{\max}$	=	m-dimensional vectors of control bounds
t	=	current discrete time index
N_c	=	control horizon
F^*	=	$\Re^n \times \Re^m \times \Re \rightarrow \Re$, mapping of continuous state equations
f^*	=	$\Re^n \times \Re^m \times \Re \rightarrow \Re$, mapping of discrete state equations
ζ	=	$\Re^n \rightarrow \Re^{n_0}$ output mapping
$J_r(u,t)$	=	receding horizon performance index
L^*	=	$\Re^n \times \Re^m \times \Re^m \times \Re^{n_0} \times \Re^{n_0} \times \Re \rightarrow \Re$ performance function
φ	=	$\Re^n \rightarrow \Re$ scalar terminal weighting function
Ψ	=	$\Re^n \rightarrow \Re^{n_c}$ system of state dependent constraints
$\hat{x}(t+1 \mid t)$	=	n-dimensional vector of state predictions at $t+1$ based on estimation at t

The performance index may weight deviations from desired set-points or it may reflect economic objectives. It is expressed in discrete time. The plant dynamics are discretized in an exact way, by integrating the continuous state equation between sampling times (*exact discretization scheme*, see Section 5.4). The relationship between control horizon N_c and prediction horizon N_p is fixed: $N_p = N_c$. Knowledge of future set-point changes can be included. This may be important in batch processing or in scheduled operations.

The output prediction is corrected by adding, along the prediction horizon, the current prediction error to the model based prediction, in the same way as future disturbances are characterized in the formulation of Dynamic Matrix Control (Garcia

and Morshedi, 1986). This is equivalent to the estimation of output disturbances. The current prediction error d_y is given by:

$$d_y = y_m(t) - \zeta(x(t)) \tag{9,2}$$

where $y_m(t)$ is the measured output at discrete time t.

The initial state condition $x(t+1)$ is the current state estimate $\hat{x}(t)$ (which is based on the latest output measurement at discrete time t) projected one step ahead in the future by integrating a plant model from t to $t+1$. This prediction is based on the current control input being applied to the plant (Sistu and Bequette, 1992).

If the structure of the model is incorrect, then after the uncertain parameters are estimated, there will still be a difference between output predictions and measurements. This remaining error is estimated as an output disturbance by using the output correction outlined above. This combines both parameter estimation and disturbance estimation in the feedback path (Eaton and Rawlings, 1990).

The controller has a time delay of one sampling period to account for the controller calculations. Thus, it is assumed that all the controller calculations can be done in one sampling interval. With the increasing computer power becoming available at decreasing prices, this will not be a limitation.

9.2.2 Nonlinear predictive control algorithm (NLP-DISOPE)

The predictive controller can be described in an algorithmic way as follows:

Algorithm 9.2.2: Nonlinear predictive control using DISOPE (NLP-DISOPE)

Step 0 Initialization

Step 1 Measure plant output $y(t)$

Step 2 Apply current control $u(t)$ to the plant.

Step 3 Simultaneously with step 2, do the following calculations

3.1 Estimate the current state $\hat{x}(t)$ and parameter vector $\hat{\theta}$ by using the Extended Kalman Filter.

3.2 Predict the state one step ahead in the future to obtain $\hat{x}(t+1|t)$ by integrating a nonlinear model of the plant from the current estimate based on the control signal being applied to the plant.

3.4 Solve the nonlinear dynamic optimization problem over the specified prediction horizon using DISOPE, with $\hat{x}(t+1|t)$ as initial state condition to obtain the control sequence $\{ u(t+1), \ldots, u(t+N_c) \}$.

Step 4 Wait until the next sampling time, then set $t = t+1$ and go to step 1.

Figure 9.2.2 shows an schematic diagram of the nonlinear predictive controller.

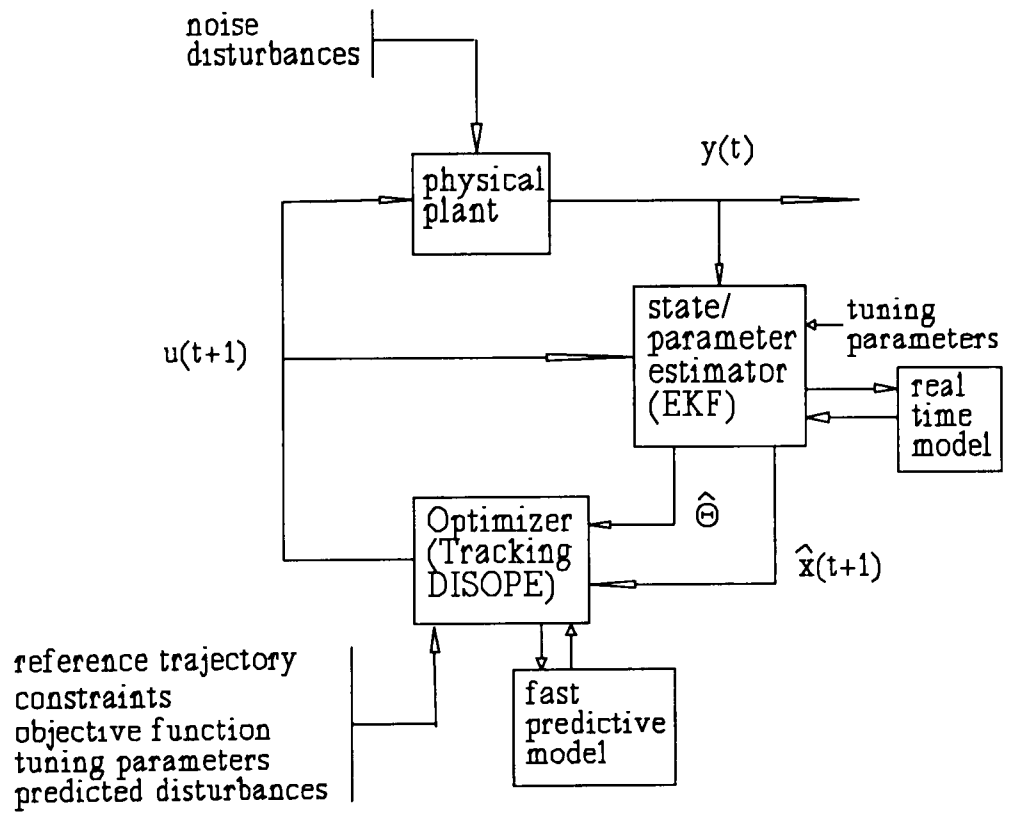

Figure 9.2.2: Schematic diagram of the controller

9.3 DYNAMIC OPTIMISATION ALGORITHM

In this work, the receding horizon optimisation formulated in Section 9.2.1 is solved at every sampling interval by using the set-point tracking DISOPE algorithm (Algorithm 6.2.1). The reader is referred to Chapter 6 for the development of Algorithm 6.2.1.

9.3.1 Choice of the LQ model-based problem in set-point tracking DISOPE.

An advantageous choice (see Example 6.4.1) within the framework of predictive control for the model-based dynamic matrices A ,B, C is a linearization about the initial state condition (which is the current state estimate projected one step ahead based on the current control). Therefore we have:

$$A = \frac{\partial f^*}{\partial x}\Big|_{\hat{x}(t+1|t),\,u(t),\,t+1}$$

$$B = \frac{\partial f^*}{\partial u}\Big|_{\hat{x}(t+1|t),\,u(t),\,t+1} \qquad \textbf{(9,3)}$$

$$C = \frac{\partial \zeta}{\partial x}\Big|_{\hat{x}(t+1|t),\,t+1}$$

The choice of the model-based weighting matrices Q, R should be done in such a way that the model based performance function L represents an approximation of the not necessarily quadratic L^*.

9.3.2 Handling of control magnitude constraints

Control magnitude constraints may be handled by using the variable transformation technique described in Section 3.3.

9.3.3 Handling of state dependent constraints

State dependent constraints may be handled by using the penalty relaxation approach described in Chapter 7

9.3.4 DISOPE algorithm initialization (nominal solution)

The nominal solution u^0, x^0, p^0 for the DISOPE algorithm may be taken as $x^0(k) = x_0$, $p^0(k) = 0$, $k \in [t+1, t+N_c+1]$, and $u^0(k) = u(t)$, $k \in [t+1, t+N_c]$, where $x_0 = \hat{x}(t+1 \mid t)$ is the predicted state at $t+1$ given information up to t, and $u(t)$ is the latest applied control.

9.4 STATE/PARAMETER ESTIMATION ALGORITHM

A very important aspect for the performance of the nonlinear predictive controller is the correct initialization of the state vector at each receding horizon optimization. Normally, such states are not directly available for measurements. Even if the full state vector is available from plant measurements, the performance of the controller can be very poor in the presence of model-plant mismatch and/or measurement noise (Bequette, 1991). Thus it is very important to estimate uncertain parameters and also to estimate the state vector from possibly noisy and incomplete output measurements.

When using state-space models developed from first principles, the number of uncertain parameters is usually small. According to Balchen *et. al.* (1992):

> *" One of the most important features of a state-space model developed by first principles is that most of the model parameters are either given or reasonable values can be found from physical considerations. The number of unknown parameters is then reduced to a minimum"*

9.4.1 Extended Kalman Filter

The nonlinear state estimator used in this work is the well known Extended Kalman Filter (EKF) (Maybeck, 1982; Lewis, 1986b). In order to estimate simultaneously states and uncertain parameters the state vector can be augmented with such parameters.

Assume the nonlinear stochastic continuous-time system with discrete-time measurements, as described as follows

$$\frac{dx}{d\tau} = F^*(x,u,\tau) + G_f(\tau)w(\tau)$$
$$y_m(t) = \zeta(x(t)) + v_m(t) \qquad \textbf{(9,4)}$$

where τ is a continuous time variable and t is a discrete time variable, $F^* : \Re^n \times \Re^m \times \Re \rightarrow \Re^n$ is a dynamic model, $\zeta : \Re^n \rightarrow \Re^{n_o}$ is an output mapping, $x \in \Re^n$ is the state vector, $u \in \Re^m$ is the control vector, $y_m \in \Re^{n_o}$ is the measurement vector, $w(\tau) \in \Re^{n_s}$ and $v_m(t) \in \Re^{n_o}$ are zero mean, white noise processes uncorrelated with each other and with the expected value of the initial state $x(0)$, $Ex(0) = \bar{x}_0$; Q_f, R_f and P_0 are covariance matrices of the appropriate dimensions corresponding to $w(\tau)$, $v_m(t)$ and $x(0)$, respectively; G_f is the process noise distribution matrix.

Define

$$A_f(x,\tau) = \frac{\partial F^*(x,u,\tau)}{\partial x}$$
$$C_f(x) = \frac{\partial \zeta(x)}{\partial x}$$

A description of the continuous-discrete EKF is given in Algorithm 9.4.1.

Algorithm 9.4.1: Extended Kalman Filter

Data — Q_f, R_f, G_f, P_0 and access to F^* and ζ.

Step 0 — Filter initialization t=0, k=0, $P(0) = P_0$, $x(0) = \bar{x}_0$, where P is the error covariance matrix.

Step 1 — *Time update.* Integrate from τ_{t-1} to τ_t the following differential equations to obtain $\hat{x}^-(\tau_t)$ and $P^-(\tau_t)$:

estimate

$$\dot{x} = F(x(\tau),u(\tau),\tau)$$

error covariance

$$\dot{P} = A_f(x,\tau)P(\tau) + P(\tau)A_f(x,\tau)^T + G_f Q_f G_f^T$$

Step 2 *Measurement update.* Wait until next sampling time, measure the current output $y_m(t)$ and then compute:

Kalman gain

$$K(t) = P^-(\tau_t)C_f^T(\hat{x}^-(\tau_t))[C_f(\hat{x}^-(\tau_t))P^-(\tau_t)C_f^T(\hat{x}^-(\tau_t)) + R_f]^{-1}$$

error covariance

$$P(\tau_t) = [I_{n_f} - K(k)C_f(\hat{x}^-(\tau_t))]P^-(\tau_t)[I_{n_f} - K(t)C_f(\hat{x}^-(\tau_t))]^T + K(t)R_f K^T(t)$$

corrected estimate

$$\hat{x}(\tau_t) = \hat{x}^-(\tau_t) + K(t)[y_m(t) - \zeta(\hat{x}^-(\tau_t), t)]$$

then set $t = t+1$ and go to step 1.

9.4.2 Kalman Filter tuning

The parameters to be initially specified when implementing a Kalman filter are the initial error covariance matrix P_0, the expected value of the initial state $\bar{x}_0$, the measurement noise covariance matrix R_f and the process noise covariance matrix Q_f. These values should be chosen from physical considerations whenever possible, taking into account that the value of the error covariance matrix is a measure of the uncertainty in the value of the initial state.

In the presence of model structure uncertainty, unmodelled dynamics, parametric uncertainty and poor information on process noise statistics, there are strategies to prevent the *divergence* of the estimates and give robustness to the filter (Lewis, 1986b; Maybeck, 1982). Such approaches prevent the covariance error, and hence the Kalman gain from going to zero with time. This keeps information from plant measurements entering into the filter calculations, so avoiding too much

confidence in the filter predictions. One of those approaches, called *fictitious process noise injection*, consists in increasing the values of the process noise covariance matrix Q_f, or using a finite positive value if believed to be zero.

The Extended Kalman filter can be used to estimate poorly known parameters within the state-space model of the plant. This is achieved by modelling the unknown parameters as additional states. Then the original state vector is augmented with the following state vector corresponding to the parameter vector $\hat{\theta} \in \Re^{n_\theta}$ to be estimated:

$$\dot{x}_\theta = \eta(\tau) \qquad \textbf{(9,5)}$$

where $\eta \in \Re^{n_\theta}$ is a zero mean white noise process with covariance Q_θ. Thus, it is necessary to specify initial estimates of the parameter vector $\hat{\theta}_0$ and its covariance matrix $P_\theta(0)$, and also the covariance matrix Q_θ of the pseudo-noise η. These are considered as tuning parameters. The process covariance matrices are then augmented correspondingly.

9.5 SIMULATION EXAMPLES

Simulation software implementing the above described nonlinear predictive controller was developed in the C++ programming language using object oriented programming and modular programming techniques, making use of existing code of an implementation of set-point tracking DISOPE. The simulations were run on a SUN SPARC-station ELC. The results of examples 9.5.1, 9.5.2 and 9.5.3 may be compared with those presented by (Sistu and Bequette, 1992).

Example 9.5.1: Exothermal CSTR

This example consists of a first order irreversible chemical reaction carried out under exothermal conditions in a continuous stirred tank reactor (CSTR). Here x_1 represents the dimensionless concentration, x_2 represents the dimensionless temperature (controlled variable), the control variable u is the dimensionless cooling

jacket temperature. The control variable is bounded between upper and lower levels. This reactor has a complex dynamic behaviour (parametric sensitivity, multiplicity of steady states) and represents a challenging control problem (Sistu and Bequette, 1992). Here t' is a dimensionless continuous time variable and t is the corresponding discrete time variable.

Plant dynamics:

$$\frac{dx_1}{dt'} = -\phi x_1 \exp\left(\frac{x_2}{1+x_2/\gamma}\right) + q(x_{1f} - x_1)$$

$$\frac{dx_2}{dt'} = \beta\phi x_1 \exp\left(\frac{x_2}{1+x_2/\gamma}\right) - (q+\delta)x_2 + \delta u + q x_{2f}$$

$$y(t) = x_2(t)$$

$$x(0) = [0.8560 \quad 0.8859]^T$$

$$u(0) = 0$$

$$-1 \le u(t) \le 2$$

where

$$[\ \phi,\ \gamma,\ \delta,\ q,\ x_{1f},\ \beta,\ x_{2f}\ ] = [\ 0.072,\ 20,\ 0.3,\ 1,\ 1,\ 8,\ 0\ ]$$

Performance index:

$$J_r(u,t) = \frac{1}{2}\sum_{k=t+1}^{t+N_c} [\ (y^c(k) - r_0(k))^2 + 0.05\,\Delta u(k)^2\ ]$$

Model-based performance index:

$$J_m(u,t) = \frac{1}{2}\sum_{k=t+1}^{t+N_c} [\ (y^c(k) - r_0(k))^2 + 0.05\,u(k)^2 + 2\gamma(k)\ ]$$

There is a set-point change from the stable lower steady state y_{sp}=0.8859 to the set-point y_{sp}=2 at t=0 (Sistu and Bequette, 1992). The sampling interval of the controller was T_s = 1.0. Ten steps of fixed step-size Runge-Kutta integration per

sample interval were used to discretize the continuous dynamics. An ideal model was assumed. The control horizon used was $N_c = 10$. Table 9.5.1.1 shows the controller performance in terms of maximum and average number of iterations per sample interval, and also maximum and average CPU time used for the controller computations per sample interval. Figure 9.5.1.1 shows the output response of the reactor. Figure 9.5.1.2 shows the corresponding control signal. Figure 9.5.1.3 shows the number of DISOPE iterations performed per sample interval. Notice that the number of iterations per sample interval increases in the transient periods and it is only one at steady state.

Maximum No. DISOPE iter.	Average No. DISOPE iter.	Maximum CPU time (s)	Average CPU time (s)
23	2.9	126	16.3

Table 9.5.1.1: Example 9.5.1, controller performance

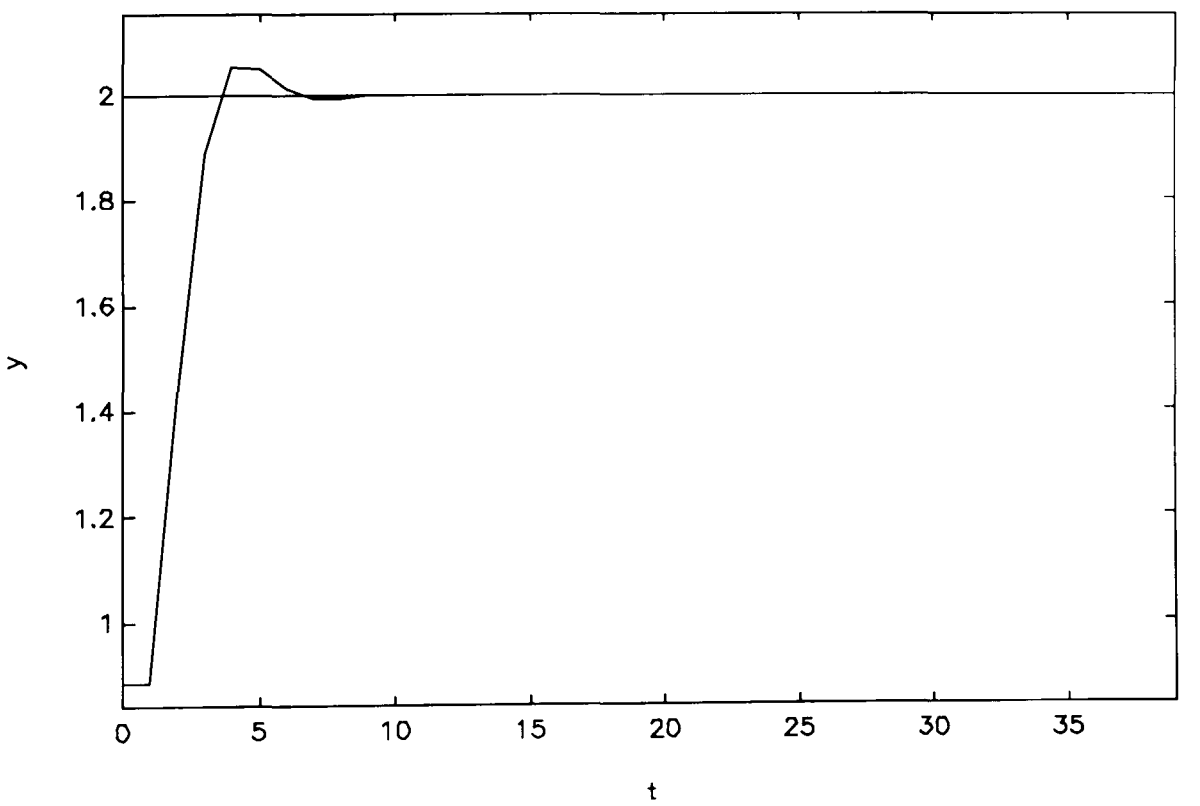

Figure 9.5.1.1: Example 9.5.1, dimensionless temperature and set-point

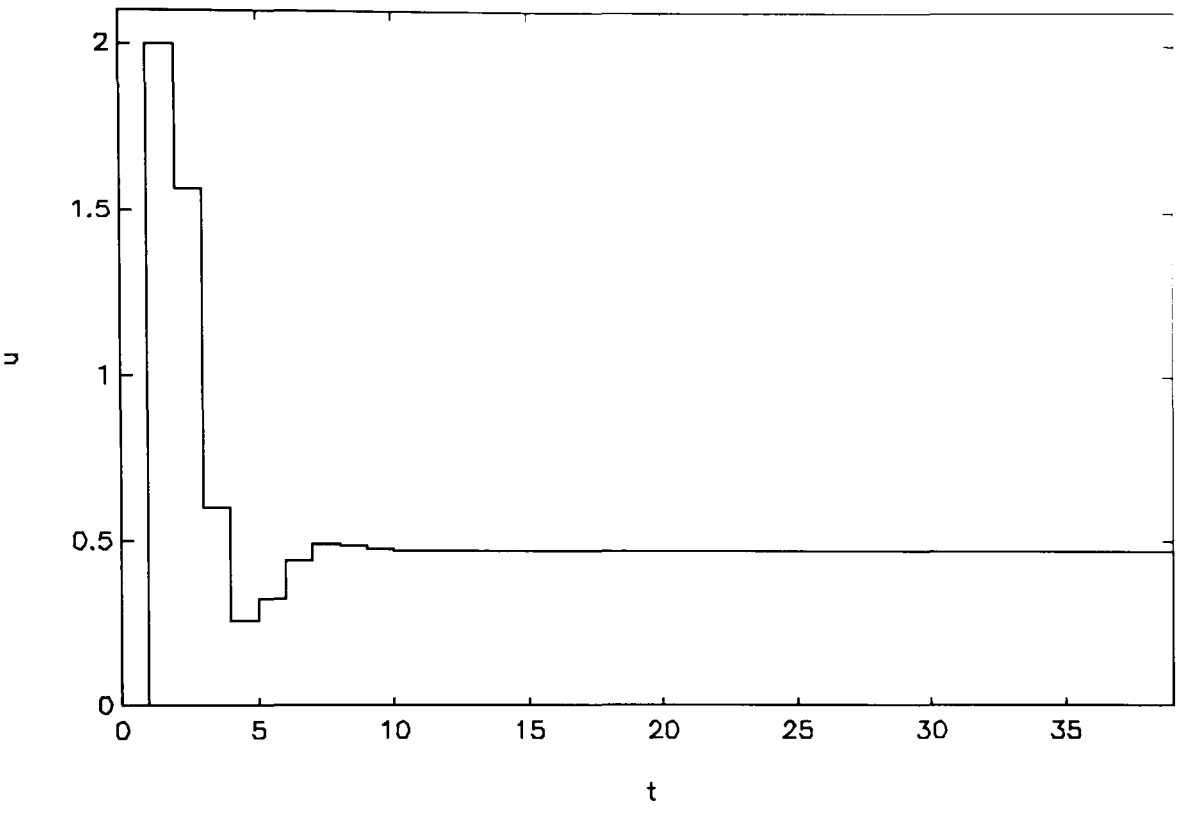

Figure 9.5.1.2: Example 9.5.1, control signal

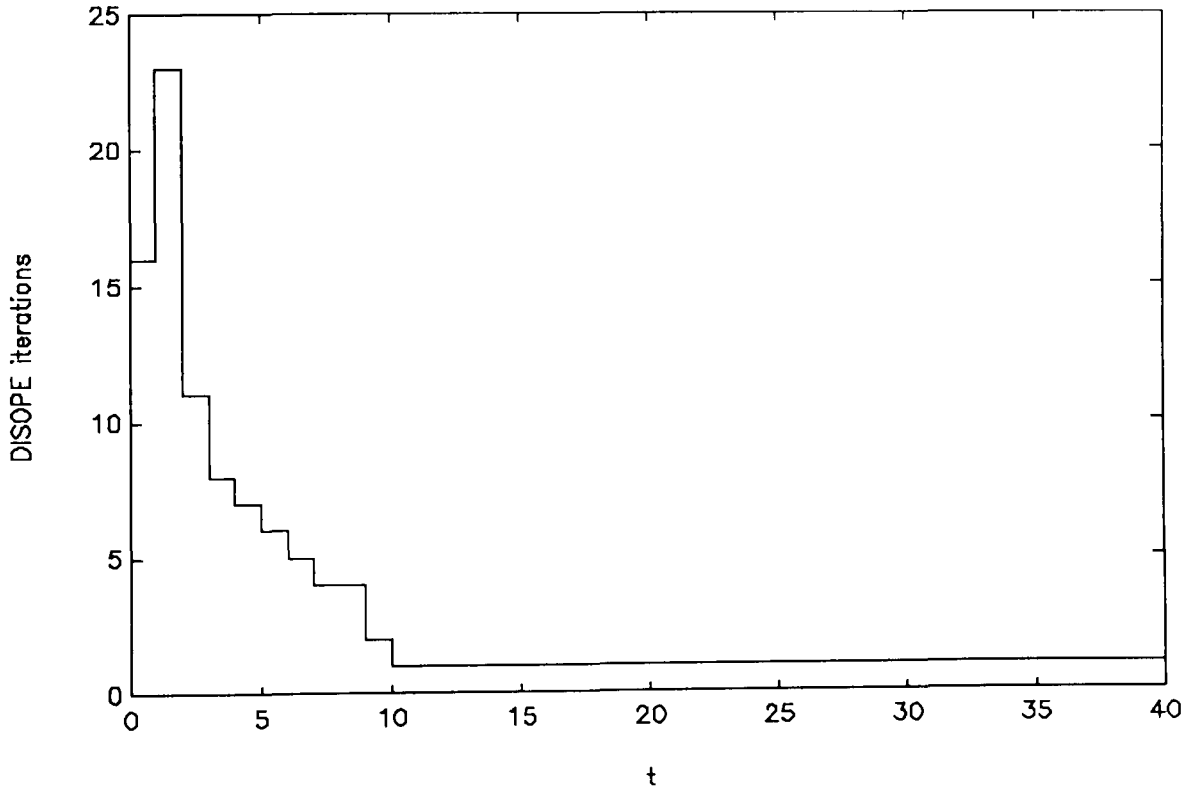

Figure 9.5.1.3: Example 9.5.1, number of DISOPE iterations per sample interval

Example 9.5.2: Exothermal CSTR with parametric uncertainty

This example consists of the same reactor system as in Example 9.5.1, but here we want to illustrate the performance of the controller in the face of model uncertainties. Parametric uncertainty was simulated assuming that the model has a heat transfer coefficient δ=0.2, while the true plant value was δ=0.3. Here we make use of the parameter adaption capabilities of the EKF to give robustness to the estimates. The effects of uncertain parameter estimation on the performance of the controller are clearly shown.

Using the same performance index as in Example 9.5.1, the responses for the EKF tuning cases (a) (without uncertain parameter estimation, see Table 9.5.2.1), and (b) (with estimation of parameter δ from the initially incorrect value δ=0.2) are shown in Figures 9.5.2.1 and 9.5.2.2, respectively. The parameter estimate and control signal for case (b) are shown in Figures 9.5.2.3 and 9.5.2.4, respectively. It is observed that in (a) there is a steady state off-set, while the steady state off-set is removed by estimating the uncertain parameter in (b).

Case	Q_f	P_0	$\bar{x}_0$	R_f
a	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$	$\begin{bmatrix} 0.8560 \\ 0.8859 \end{bmatrix}$	0.001
b	$\begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.8560 \\ 0.8859 \\ 0.2 \end{bmatrix}$	0.001

Table 9.5.2.1: Extended Kalman Filter tuning parameters for example 9.5.2

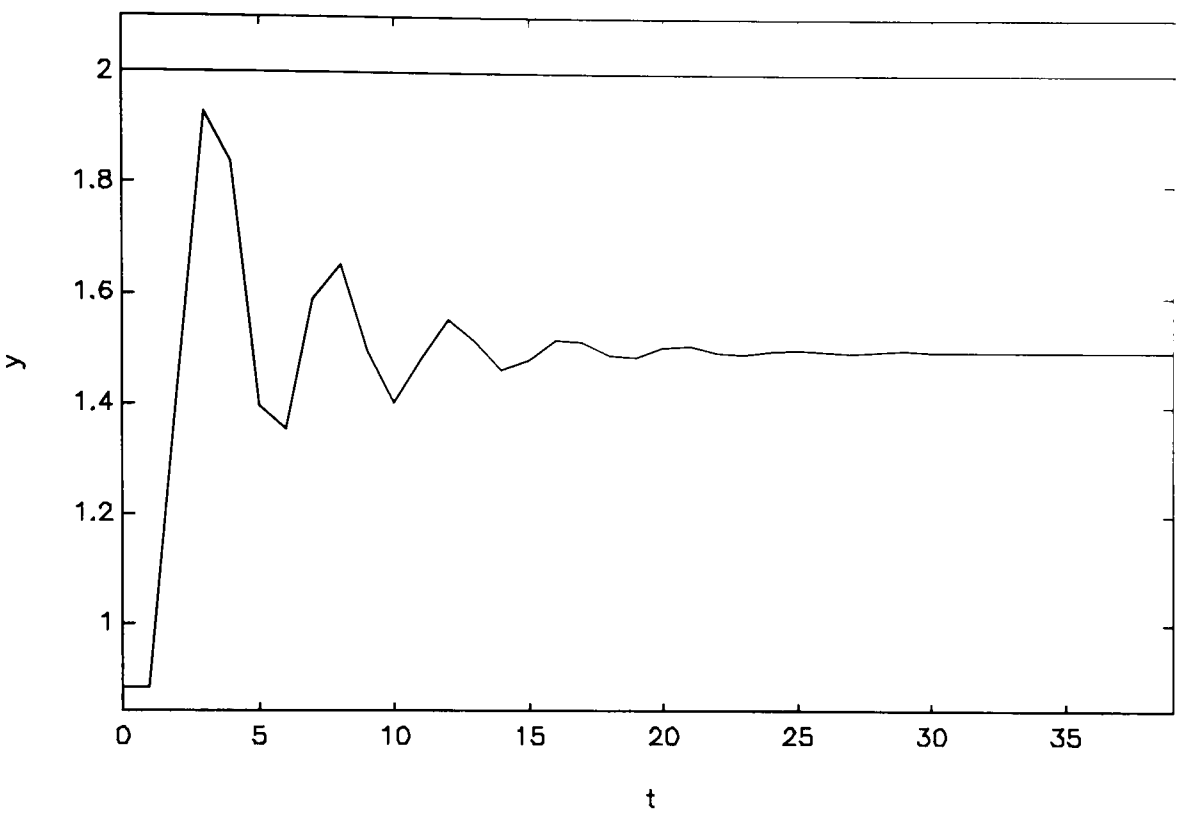

Figure 9.5.2.1: Example 9.5.2, dimensionless temperature and set-point, case (a).

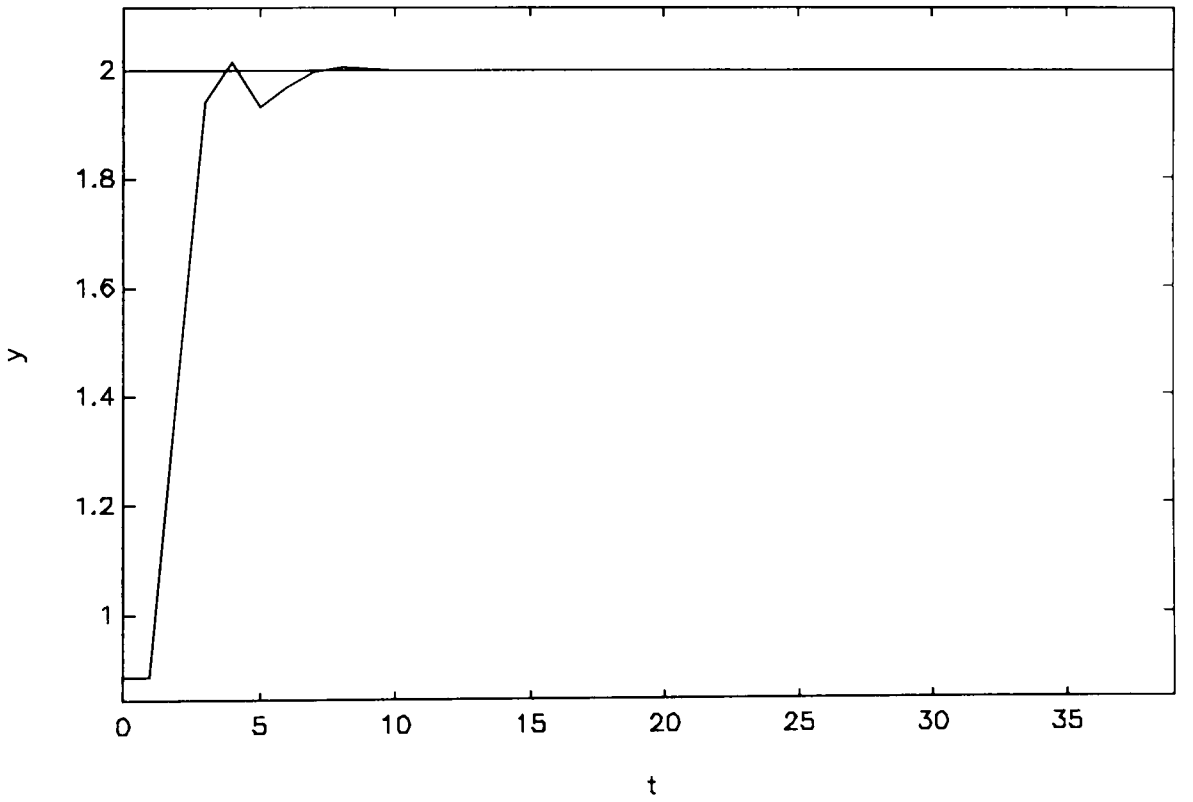

Figure 9.5.2.2: Example 9.5.2, dimensionless temperature and set-point, case (b).

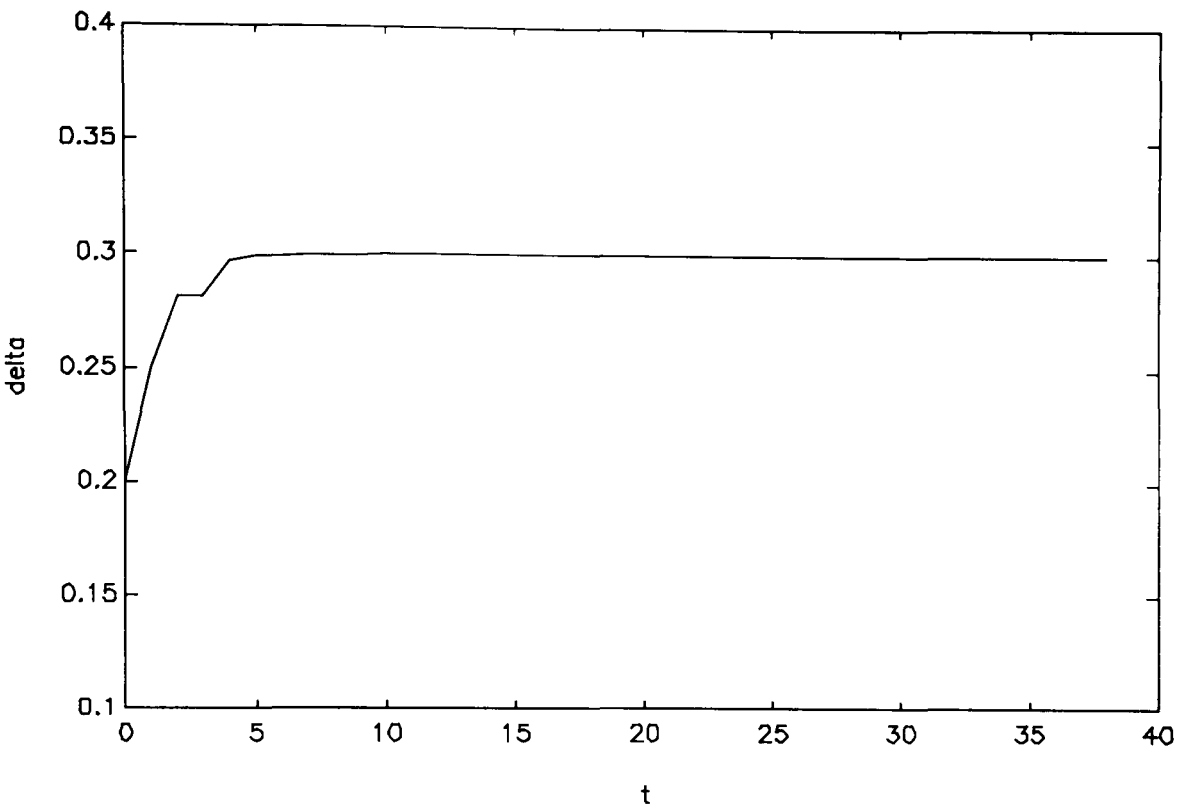

Figure 9.5.2.3: Example 9.5.2, parameter estimate, case (b)

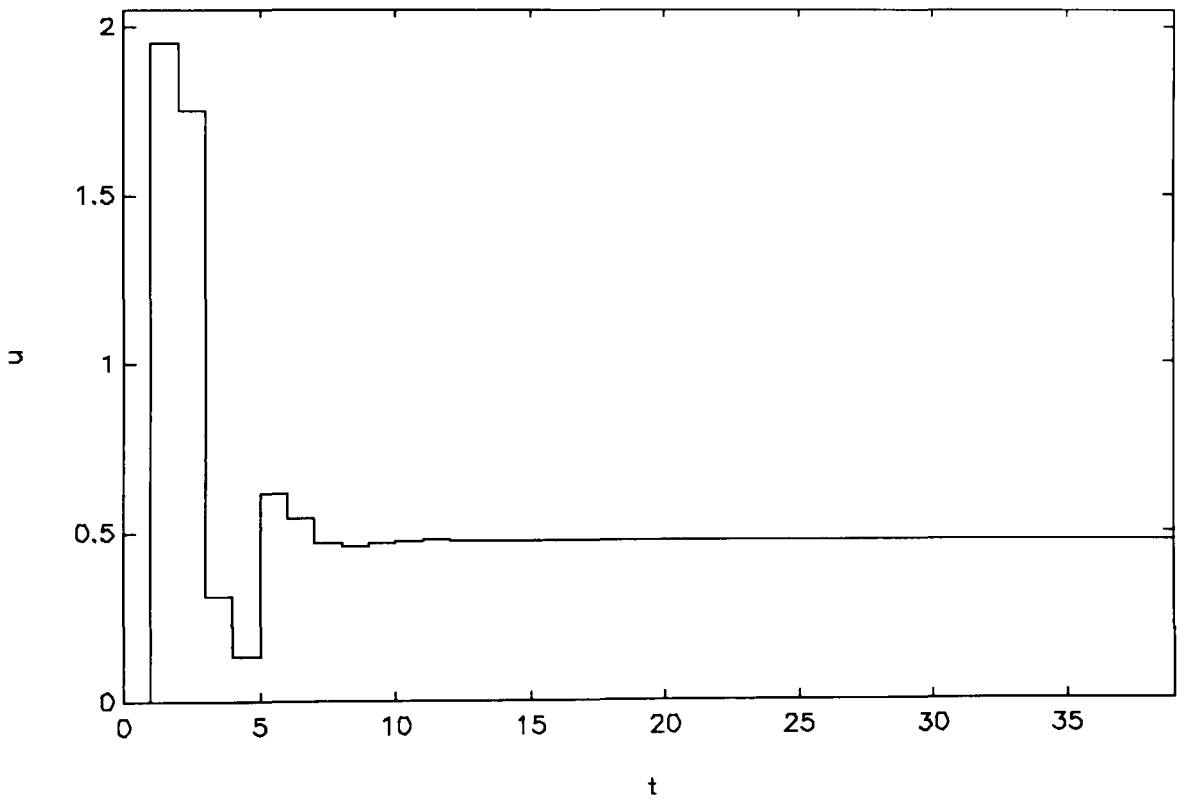

Figure 9.5.2.4: Example 9.5.2, control signal case (b)

Example 9.5.3: Exothermal CSTR under load disturbance

This example consists of the same reactor of Example 9.5.1, but here we want to illustrate the disturbance rejection of the system. This was tested with a simulated feed concentration (x_{1f}) increase of 20% at t=5. The system was assumed to be operating at its lower steady state before the disturbance occurred. The parameter x_{1f} was initialized with its correct nominal value $x_{1f}=1$. The performance index specifications used were the same as in Example 7.5.1.

Here we will distinguish two cases according to the tuning of the Kalman filter (see Table 9.5.3.1):

(a) Fictitious process noise injection;

(b) Continuous estimation of the faulty parameter x_{1f}.

Case	Q_f	R_f	$\bar{x}_0$	P_0
a	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$	0.001	$\begin{bmatrix} 0.8560 \\ 0.8859 \end{bmatrix}$	$\begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$
b	$\begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$	0.001	$\begin{bmatrix} 0.8560 \\ 0.8859 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.05 \end{bmatrix}$

Table 9.5.3.1: Extended Kalman Filter tuning parameters for example 9.5.3

Figure 9.5.3.1 shows the output response for cases (a) and (b). Notice that in case (a) there is a steady state off-set and that such an off-set is removed by estimating the faulty parameter in case (b). Figure 9.5.3.2 shows the estimate of x_{1f} where the parameter adaptation can be appreciated. Figure 9.5.3.3 shows the control

signal for cases (a) and (b), noticing that the estimation of x_{1f} results in feed-forward action since the controller in case (b) responds faster than in case (a).

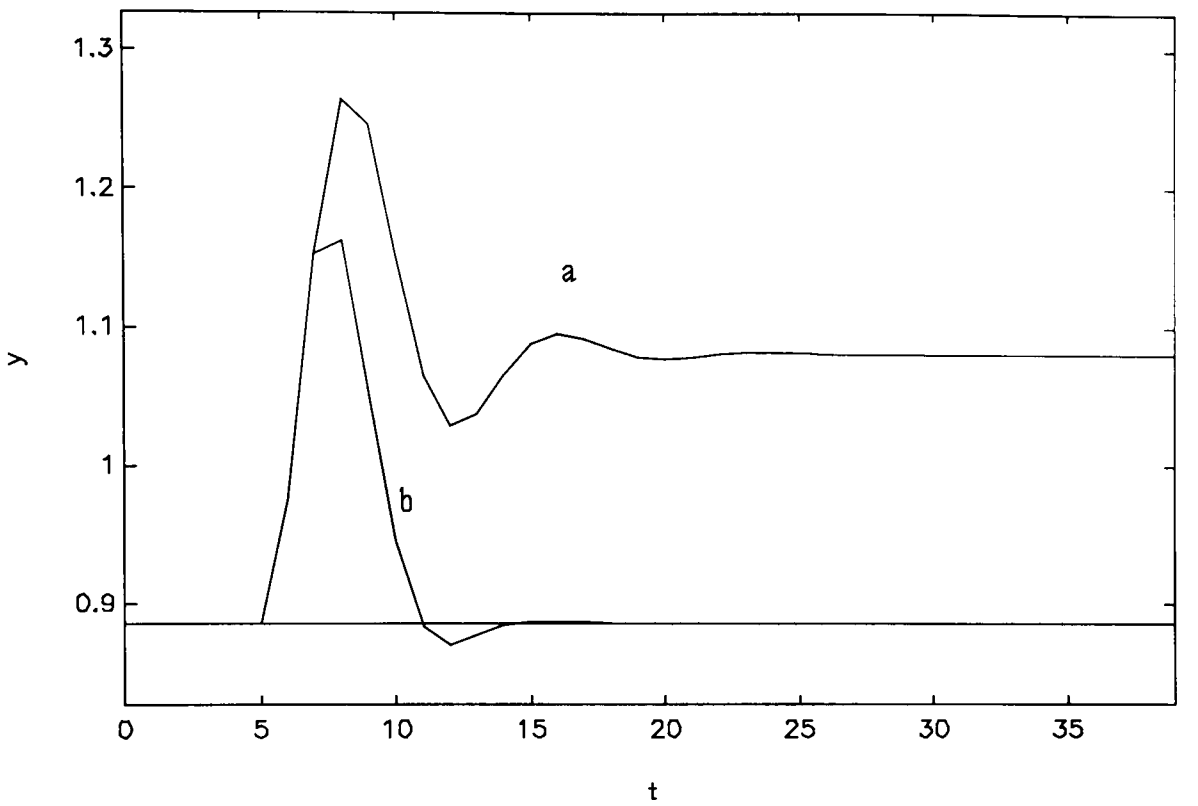

Figure 9.5.3.1: Example 9.5.3, dimensionless temperature and set-point, cases (a), (b).

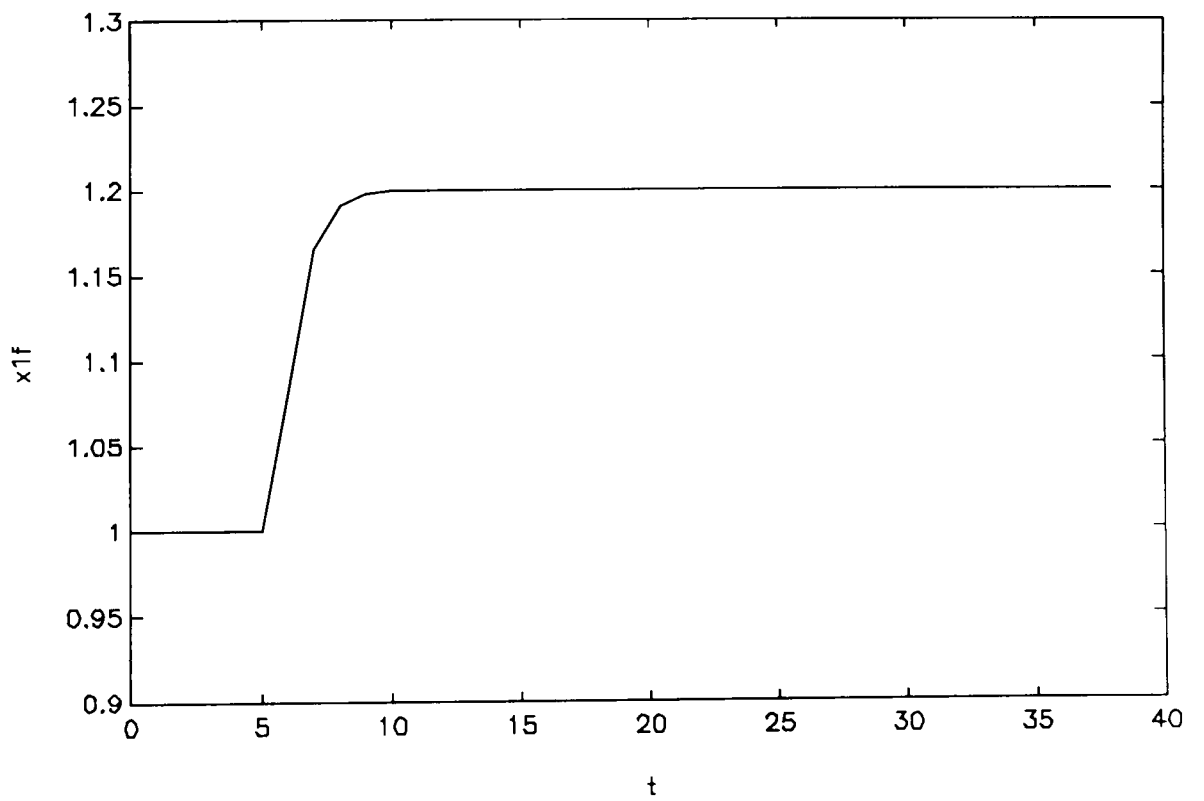

Figure 9.5.3.2: Example 9.5.3, parameter estimate, case (b)

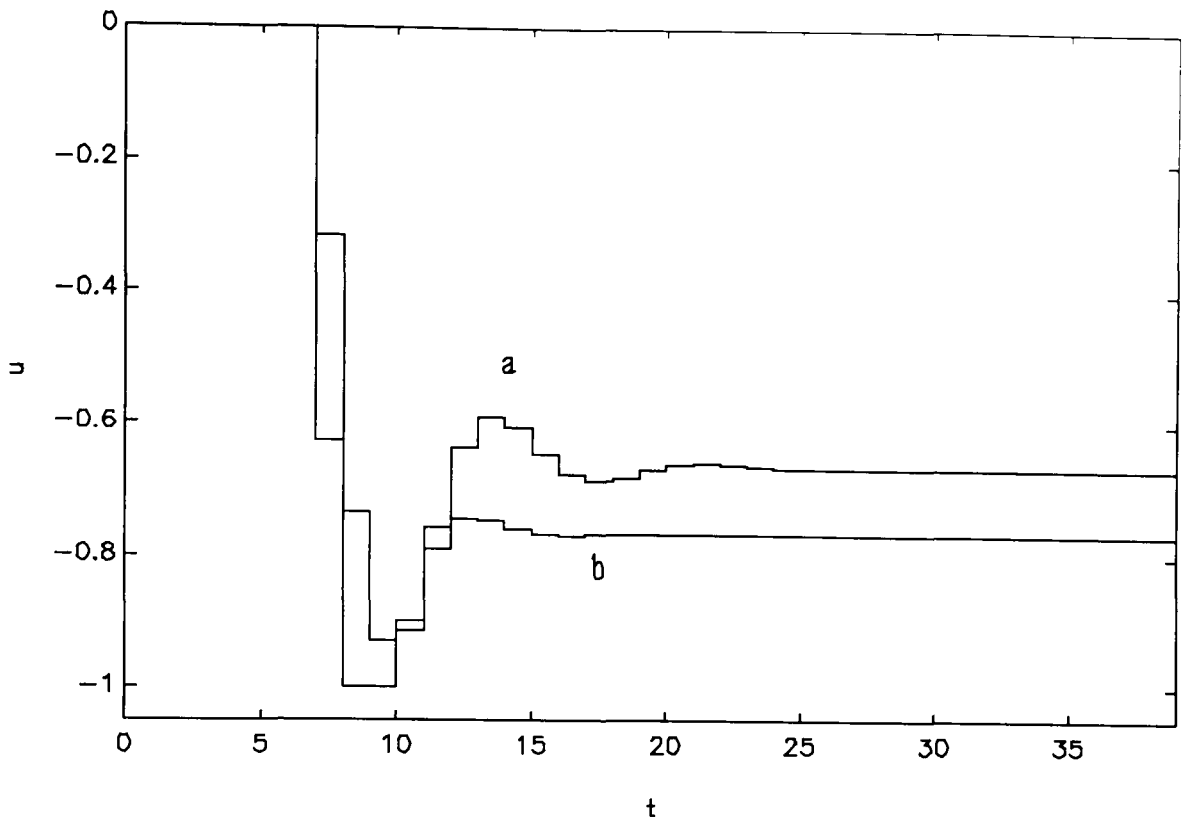

Figure 9.5.3.3: Example 9.5.3, control signal for cases (a) and (b)

Example 9.5.4: Optimisation of two CSTR in series

This example, originally presented by Garcia and Morari (1981) and later used by Jang *et al* (1987), consists of two CSTR's in series in which an exothermic autocatalytic reaction is taking place. The units interact in both directions due to the recycle of a 50% fraction of the product stream into the first reactor. Regulatory controllers are used to control the temperature in both reactors. The dynamics associated with these controllers are neglected. Here t' (min) is a continuous time variable and t is the corresponding discrete time variable. The performance index optimised by the nonlinear predictive controller reflects economic objectives associated with the achievement of maximum production of substance B. The reaction is:

$$A + B \underset{k_-}{\overset{k_+}{\rightleftharpoons}} 2B$$

The dynamics of the reactor are given by:

$$\frac{dCa_1}{dt'} = \frac{0.5}{\tau_1}(Ca_0+Ca_2) - \frac{Ca_1}{\tau_1} - (k_{1+}Ca_1 Cb_1 - k_{1-}Cb_1^2)$$

$$\frac{dCb_1}{dt'} = \frac{0.5}{\tau_1}Cb_2 - \frac{Cb_1}{\tau_1} + (k_{1+}Ca_1 Cb_1 - k_{1-}Cb_1^2)$$

$$\frac{dCa_2}{dt'} = \frac{Ca_1}{\tau_2} - \frac{Ca_2}{\tau_2} - (k_{2+}Ca_2 Cb_2 - k_{1-}Cb_2^2)$$

$$\frac{dCb_2}{dt'} = \frac{Cb_1}{\tau_2} - \frac{Cb_2}{\tau_2} + (k_{2+}Ca_2 Cb_2 - k_{2-}Cb_2^2)$$

where

Ca_1= Concentration of A in reactor 1

Cb_1= Concentration of B in reactor 1

Ca_2= Concentration of A in reactor 2

Cb_2= Concentration of B in reactor 2

τ_1= Reactor 1 residence time, 30 min.

τ_2= Reactor 2 residence time, 25 min.

$k_{i\pm}$= $A_{\pm}\exp(-E\pm/RT_i)$

E_+/R= 17786 K

E_-/R= 23523 K

A_+= 9.73×10^{22} m^3/kmols

A_-= 3.1×10^{30} m^3/kmols

Ca_0= Feed concentration of A, 0.1

T_1= Temperature of reactor 1 (manipulated variable)

T_2= Temperature of reactor 2 (manipulated variable)

The manipulated variables are bounded between upper and lower levels: $300 \le T_1 \le 312$ and $300 \le T_2 \le 312$.

The measured variables are Cb_1 and Cb_2, while Ca_1 and Ca_2 are estimated by using the Extended Kalman Filter (see Table 9.5.4.1). The sampling interval used was T_s = 5 min. An ideal model was assumed for the model-based open loop optimal control calculations. The continuous dynamics were integrated between sampling times by using an adaptive step-size Runge-Kutta integrator (Press *et al.* 1992).

The real performance index reflects our objective of maximising the concentration of B in the second reactor and, in addition, we use quadratic weighting on the increments of the manipulated variables:

$$J_r = \sum_{k=t+1}^{t+N_c} (-Cb_2(k) + 0.002\Delta T_1(k)^2 + 0.002\Delta T_2(k)^2)$$

with a control horizon $N_c = 5$.

The model-based performance index used for the iterations of DISOPE was:

$$J_m = \sum_{j=t+1}^{t+N_c} (0.002\, T_1(k)^2 + 0.002\, T_2(k)^2 + \gamma(k))$$

Notice that in this example, since the performance index is based on economic objectives, no reference trajectory is specified and the predictive controller decides where to take the system (the steady-state optimum) and how it takes the system to that operating point.

The reactor is started at the suboptimal steady-state point specified by the set-points T_1 = 307 K and T_2 = 302 K. Figure 9.5.4.1 shows the response of the concentration of B in the second reactor. Figure 9.5.4.2 shows the evolution of the control signals. The steady-state optimum set-points obtained were T_1 = 312 K, T_2 = 309.39 K, while the optimal concentration of B was Cb_2 = 0.07251.

Q_f	P_0	$\bar{x}_0$	R_f
$\begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 0.048351 \\ 0.051649 \\ 0.041362 \\ 0.058638 \end{bmatrix}$	$\begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix}$

Table 9.5.4.1: Extended Kalman Filter tuning parameters for example 9.5.4

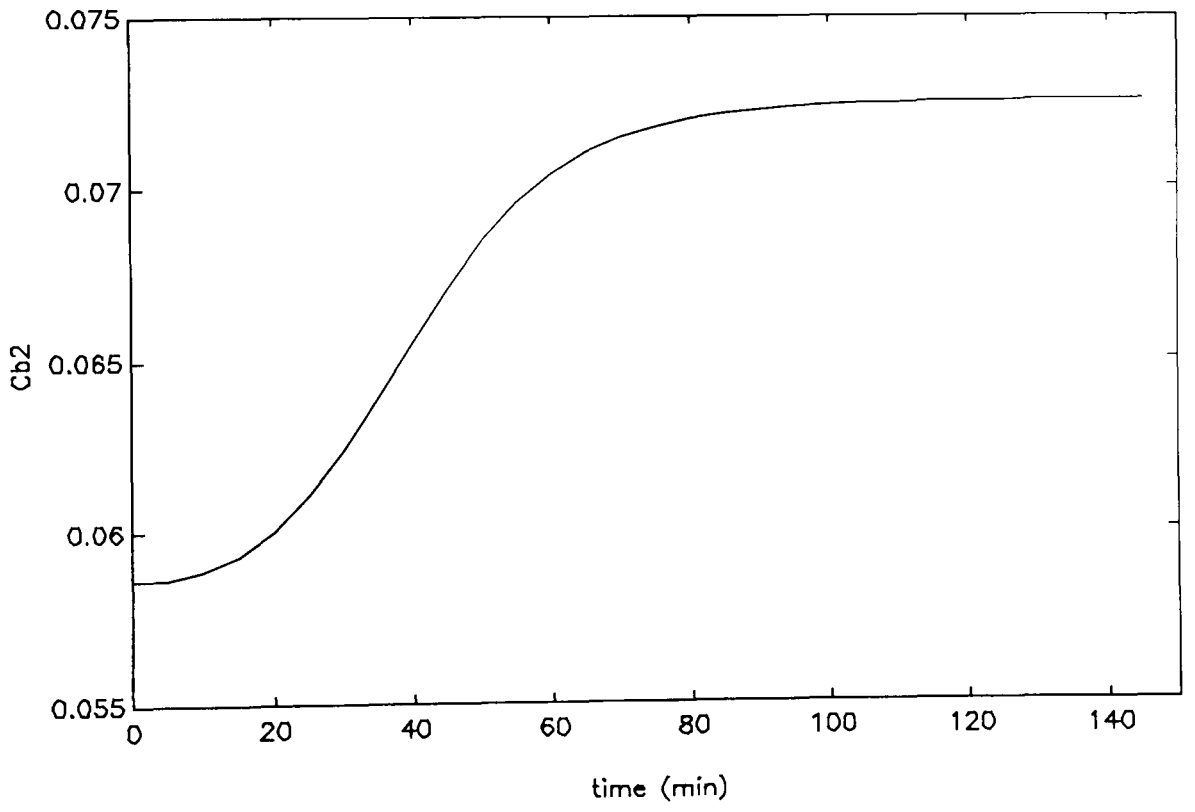

Figure 9.5.4.1: Example 9.5.4, trajectory of concentration of B in the second reactor

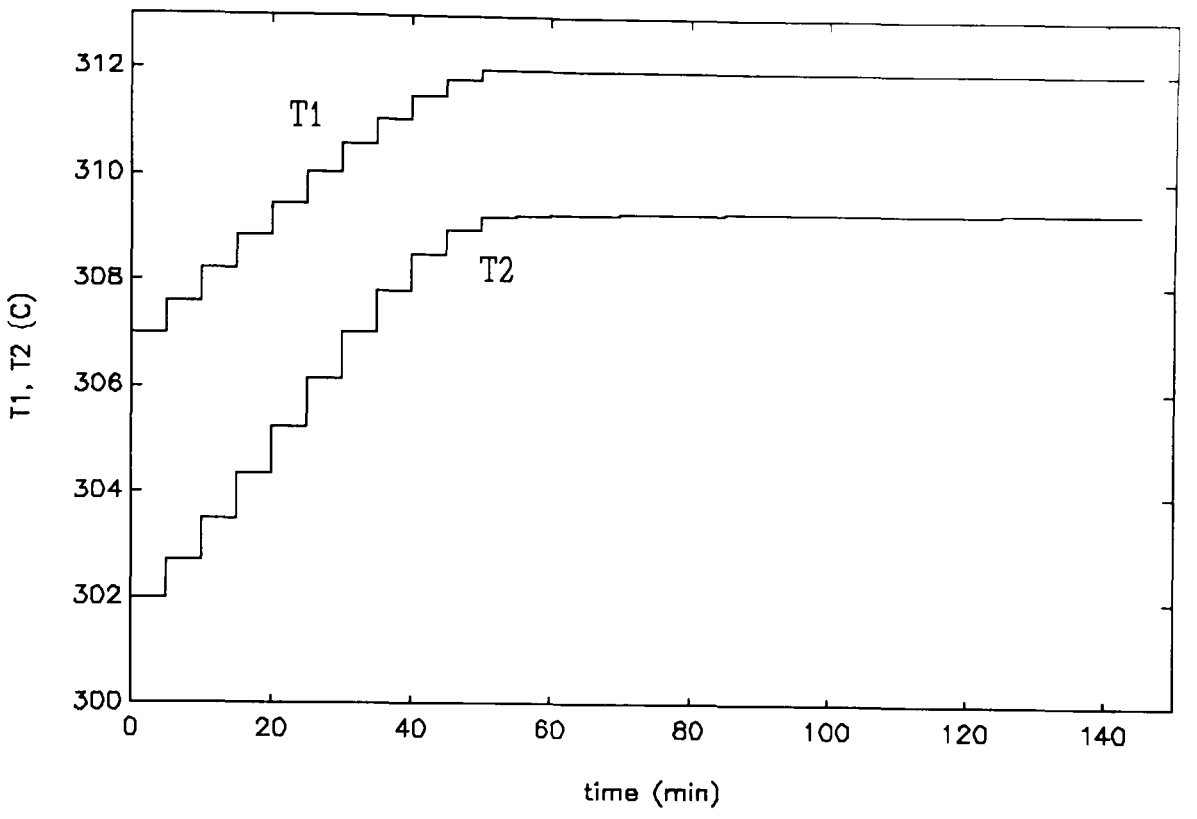

Figure 9.5.4.2: Example 9.5.4, control signals, temperatures in reactors 1 and 2

9.6 SUMMARY

A nonlinear predictive controller has been developed and implemented in software. The controller uses a Extended Kalman Filter (EKF) for state/parameter estimation and the DISOPE algorithm for the solution of the receding horizon dynamic optimisation. The controller has been tested with simulation examples, where its capabilities for handling nonlinear systems, model-plant mismatch and parameter adaptation, disturbance rejection and inequality constraints have been evaluated. Both set-point tracking and economic performance indexes have been handled by the same algorithm.

The set-point tracking DISOPE algorithm presented in Chapter 6 and used for the dynamic optimisation in this chapter, appears to be very appropriate for its use in predictive control. The choice of the performance index is flexible, allowing the inclusion of different control strategies, including economic objectives.

CHAPTER 10

NOVEL DEVELOPMENTS IN STEADY-STATE PROCESS OPTIMISATION USING DYNAMIC INFORMATION

In this chapter, an optimising controller that is able to drive a plant from a suboptimal operational condition to its steady-state optimum in a continuous way based on dynamic information, is designed. Using standard results from optimisation theory and discrete optimal control, the solution of a steady-state optimisation problem is achieved by solving a receding-horizon optimal control problem which uses derivative and state information from the plant via a shadow model and a state-space identifier. The new algorithm is developed from the basis that a nonlinear model of the process is not available for predictions and, hence, some of the ideas in developed in Chapter 9 are not applicable. The optimality of the procedure is analyzed and algorithms with and without control rate constraints are developed. A way of overcoming the lack of a nonlinear model for prediction purposes is developed. This enables the use of DISOPE in a predictive control framework by using a linearized model of the process for the predictions. Both algorithms are tested with simulation examples, including realistic simulations of an industrial distillation column using a rigorous process simulator.

10.1 LINEAR STATE-SPACE MODEL IDENTIFICATION SCHEMES

Two alternatives for the identification of a linear state space model of the process are treated in this work.

In the first alternative, a discrete state-space model in observable canonical form and in which the output variables are equal to the states, is identified and updated with a certain period using input-output data with a fixed length (data window) obtained from the *shadow model* of the plant (Griffiths, 1993; see Chapter

8). This identification scheme for estimating dynamic derivatives has been devised and implemented by Lin (1993). The identified state-space model has the following structure:

$$\hat{x}(k+1) = \hat{A}\hat{x}(k) + \hat{B}\hat{u}(k) \qquad \textbf{(10,1)}$$

where:

$$\begin{aligned} \hat{x} &= x - x_r(t) \\ \hat{u} &= u - u_r(t) \end{aligned} \qquad \textbf{(10,2)}$$

are deviations from state and control reference trajectories, respectively, $x \in \Re^n$ and $u \in \Re^m$. The value of the state vector $x(t)$ at present time t is also available.

The values of the estimated matrices $\hat{A}$, $\hat{B}$ are updated every N_u samples using data from the previous N_d samples. The states of the identified model correspond with a number of relevant states measured from the shadow model, which, chosen by an expert in the plant, are a good representation of the dynamics of the process.

In the second alternative, the derivative estimates are computed by using a recursive extended least squares estimator (Ljung, 1987) based on the multivariable ARMAX model:

$$A(q^{-1})\bar{y}(t) = B(q^{-1})\bar{u}(t) + C(q^{-1})e(t) \qquad \textbf{(10,3)}$$

where the overbar denotes data differencing, which is done in order to avoid estimating offsets, $e(t)$ is a zero mean uncorrelated random variable, $A(q^{-1})$, $B(q^{-1})$, and $C(q^{-1})$ are first order matrix polynomials:

$$\begin{aligned} A(q^{-1}) &= I_{ny} + A_1 q^{-1} \\ B(q^{-1}) &= B_1 q^{-1} \\ C(q^{-1}) &= I_{ny} + C_1 q^{-1} \end{aligned}$$

The equivalent state space model has the following structure, in innovations form:

$$\bar{x}(t+1) = \hat{A}(\theta)\bar{x}(t) + \hat{B}(\theta)\bar{u}(t) + K(\theta)e(t)$$
$$y(t) = Cx(t)+e(t) \qquad \textbf{(10,5)}$$

where θ is a parameter vector related with the polynomial coefficients in (10,4). In the particular case when the output matrix is equal to the identity matrix ($C = I_n$), and the order of the matrix polynomials (10,4) is equal to one, it is easy to find that there is a direct relationship between the identified matrix coefficients in (10,4) and the state space matrices as follows:

$$\hat{A}(\theta) = -A_1$$
$$\hat{B}(\theta) = B_1 \qquad \textbf{(10,6)}$$
$$K(\theta) = C_1 - A_1$$

The general approach treated in this chapter is illustrated in Figure 10.1.

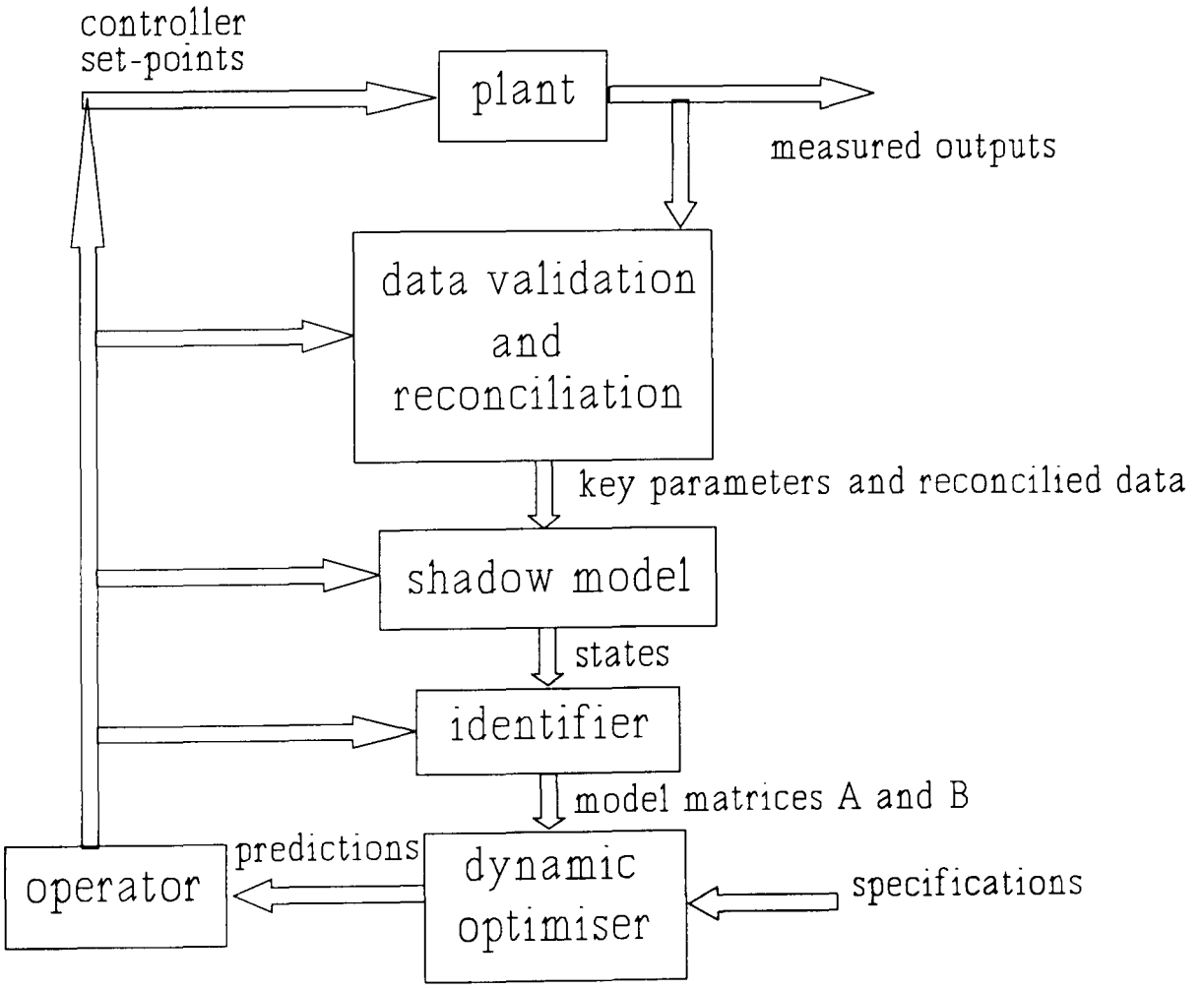

Figure 10.1: Simplified diagram of shadow model approach for plant dynamic optimisation

10.2 PRELIMINARY ASSUMPTIONS

A10.2.1 For any admissible set of set-points, the corresponding steady-state of the process is asymptotically stable.

A10.2.2 The rate of update of the identified model is fast compared to the dynamics of the process.

A10.2.3 The derivatives of $f^*(x,u)$ with respect to its arguments exist and are Lipschitz continuous.

10.3 INITIAL FORMULATION

Assume that the process dynamics may be described by the following nonlinear and unknown difference equation:

$$x(t+1) = f^*(x(t),u(t)) \qquad \textbf{(10,7)}$$

where $f^* : \Re^n \times \Re^m \rightarrow \Re^n$ is an unknown set of nonlinear discrete state equations.

Approximation 10.3

Under assumptions A10.2.2 and A10.2.3 the following approximation holds at a given instantaneous operating point $x_o(t)$, $u_0(t)$

$$\begin{aligned} \frac{\partial f^*(x,u)}{\partial x} \Big|_{x_0(t),u_0(t)} &\approx \hat{A} \\ \frac{\partial f^*(x,u)}{\partial u} \Big|_{x_0(t),u_0(t)} &\approx \hat{B} \end{aligned} \qquad \textbf{(10,8)}$$

Proof

Let us examine the increment in $x(t+1)$ given small increments in $x(t)$ and $u(t)$:

$$x(t+1) + \Delta x(t+1) = f^*(x(t)+\Delta x(t), u(t)+\Delta u(t)) \qquad \textbf{(10,9)}$$

Expanding by Taylor's series the right hand side:

$$x(t+1) + \Delta x(t+1) = f^*(x(t),u(t)) + \frac{\partial f^*}{\partial x(t)}\Delta x(t) + \frac{\partial f^*}{\partial u(t)}\Delta u(t) + \ldots \qquad \textbf{(10,10)}$$

Subtracting equation (10,7) from (10,10) and neglecting higher order terms we have:

$$\Delta x(t+1) = \frac{\partial f^*}{\partial x(t)}\Delta x(t) + \frac{\partial f^*}{\partial u(t)}\Delta u(t) \qquad \textbf{(10,11)}$$

Comparison with equation (10,1) yields the approximation.

Q.E.D.

We want to drive the plant to its steady-state optimum, given a steady-state objective function $N^*(x,u)$. In mathematical terms we want to solve the following steady-state optimisation problem (SSOP):

SSOP

$$\min_{u \in \Re^n} \; N^*(x,u)$$

subject to

$$g^*(x,u) = 0$$
$$C(u) \leq 0$$

where x and u are steady state values of state and control vectors, $g^* : \Re^n \times \Re^m \rightarrow \Re^n$ is the steady state mapping of the plant, and $C : \Re^m \rightarrow \Re^p$ represents a set of inequality constraints dependent on the control variables.

Notice that in the steady state equation (10,7) becomes:

$$x = f^*(x,u) \qquad \textbf{(10,12)}$$

which is equivalent to:

$$g^*(x,u) = f^*(x,u) - x \qquad \textbf{(10,13)}$$

Instead of solving SSOP directly by a two-step or modified two-step method (Roberts, 1979), which involves long settling times after every set-point change, we consider the following receding horizon dynamic optimal control problem (RHDOP).

RHDOP

$$\min_{\substack{u(k) \\ k\in[t,t+N-1]}} \quad \varphi(x(t+N)) + \sum_{k=t}^{t+N-1} L^*(x(k),u(k))$$

subject to

$$x(k+1) = f^*(x(k),u(k))$$
$$C(u(k)) \leq 0$$
$$x(t) \quad \text{given}$$
$$t\in[0,\infty]$$

where $L^* : \Re^n\times\Re^m\rightarrow\Re$ is a discrete performance function and $\varphi : \Re^n\rightarrow\Re$ is a terminal weighting function.

In other words, we want to reach the optimum solution (u_s,x_s) of SSOP by solving RHDOP in real time.

10.3.1 First order necessary optimality conditions of SSOP

The Lagrangian function of SSOP, using equation (10,13), is:

$$\mathcal{L}_s(x,u,\mu,\lambda) = N^*(x,u) + \mu^T(f^*(x,u)-x) + \lambda^T C(u)$$

where $\mu\in\Re^n$ is a Lagrange multiplier and $\lambda\in\Re^p$ is a Kuhn-Tucker multiplier. The following necessary optimality conditions can be stated (Fletcher, 1981; Lewis, 1986a):

$$\nabla_x\mathcal{L}_s = \nabla_x N^* + \left[\frac{\partial f^*}{\partial x} - I_n\right]^T\mu = 0 \qquad \textbf{(10,15)}$$

$$\nabla_u \mathcal{L}_s = \nabla_u N^* + \frac{\partial f^{*\top}}{\partial u}\mu + \frac{\partial C^\top}{\partial u}\lambda = 0 \qquad \textbf{(10,16)}$$

$$\nabla_\mu \mathcal{L}_s = f^*(x,u) - x = 0 \qquad \textbf{(10,17)}$$

$$\lambda \begin{cases} =0 & C(u)<0 \\ \geq 0 & C(u)=0 \end{cases} \qquad \textbf{(10,18)}$$

10.3.2 First order necessary optimality conditions of RHDOP

Define the Hamiltonian function as:

$$H(x,u,p,k) = L^*(x,u) + p(k+1)^\top f^*(x,u) + \Theta(k)^\top C(u) \qquad \textbf{(10,19)}$$

where $k \in [t, t+N-1]$, $p \in \Re^n$ (the costate) is a Lagrange multiplier and $\Theta \in \Re^p$ is a Kuhn-Tucker multiplier. The following necessary optimality conditions can be stated (Bryson and Ho, 1975):

$$\begin{aligned} &\nabla_{x(k)} H - p(k) = 0 \\ &\Rightarrow \nabla_{x(k)} L^* + \frac{\partial f^{*\top}}{\partial x(k)} p(k+1) - p(k) = 0 \end{aligned} \qquad \textbf{(10,20)}$$

$$\begin{aligned} &\nabla_{u(k)} H = 0 \\ &\Rightarrow \nabla_{u(k)} L^* + \frac{\partial f^{*\top}}{\partial u(k)} p(k+1) + \frac{\partial C^\top}{\partial u(k)} \Theta(k) = 0 \end{aligned} \qquad \textbf{(10,21)}$$

$$\nabla_{p(k+1)} H = x(k+1) = f^*(x,u) \qquad \textbf{(10,22)}$$

$$p(t+N) = \nabla_x \varphi(x)\big|_{t+N} \tag{10,23}$$

$$\Theta(k) \begin{cases} =0 & C(u(k))<0 \\ \geq 0 & C(u(k))=0 \end{cases} \tag{10,24}$$

10.4 FIRST APPROACH: DSSO, A STEADY-STATE OPTIMISER BASED ON DYNAMIC INFORMATION

As we have no access to the function $f^*(x,u)$, we cannot solve RHDOP in general. However, in the particular case when $N = 1$ and the terminal weighting function has the linear form:

$$\varphi(x) = \phi^\top x$$

where $\phi \in \Re^n$ is a given vector, it is not necessary to predict the state into the future in order to solve RHDOP, since the terminal condition (10,23) on the costate then becomes:

$$p(t+1) = \phi \tag{10,26}$$

10.4.1 Steady-state optimality of a RHDOP controller with N=1

Assumptions

A10.4.1 Assume that by applying to the plant (10,7) a control sequence $\{u(0), u(1) \ldots u(t_s)\}$ computed from the solution of RHDOP with $N = 1$ and $\varphi = \phi^\top x(t+1)$, the system achieves a steady-state (x_d, u_d) (in practical terms) for some finite time t_s.

A10.4.2 Assume that, in the steady state, the following holds:

$$N^*(x(t),u(t)) = \lim_{t\to\infty} L^*(x(t),u(t)) \qquad \textbf{(10,27)}$$

A10.4.3 Assume that the solution of SSOP (x_s,u_s) exists and is unique.

A10.4.4 Assume that the estimation of the derivatives of $f^*(x,u)$ with respect to x and u is perfect in the steady state.

A10.4.5 Assume that derivatives of $L^*(x,u)$ and $N^*(x,u)$ with respect to their arguments exist and are Lipschitz continuous.

The achievement of the solution of SSOP via RHDOP with *N=1* is given by the following theorem.

Theorem 10.4.1 - Steady state optimality of RHDOP solution with N=1

Under assumptions A10.2.1, A10.2.3, A10.4.1-A10.4.5, if the linear terminal weighting vector ϕ is given by:

$$\phi = \left[I_n - \frac{\partial f^*(x,u)^T}{\partial x_d} \right]^{-1} \nabla_x L^*(x,u)\big|_{(x_d,u_d)} \qquad \textbf{(10,28)}$$

then the steady state solution (x_d,u_d) of RHDOP with *N=1* satisfies also the necessary optimality conditions of SSOP.

Proof

Since the system is in steady-state then ϕ as given by (10,28) is constant. This implies that the costate p is also constant and equal to ϕ. This can be proved by noticing that the steady state solution (x_d,u_d) satisfies the optimality conditions of RHDOP. Therefore, from (10,23):

$$p(t+1) = \phi \qquad \textbf{(10,29)}$$

Furthermore, from (10,20) we have:

$$\nabla_{x_d} L^* + \frac{\partial f^{*\top}}{\partial x_d}\phi - p(t) = 0 \qquad \textbf{(10,30)}$$

If we assume that $p(t) = \phi$ from (10,30), we obtain

$$\phi = \left[I_n - \frac{\partial f^*(x_d, u_d)^\top}{\partial x_d} \right]^{-1} \nabla_{x_d} L^*(x_d, u_d) \qquad \textbf{(10,31)}$$

which agrees with (10,28).

From A10.4.2 we have $\nabla_{x_d} N^*(x_d, u_d) = \nabla_{x_d} L^*(x_d, u_d)$ which may replaced in (10,30) together with $p(t) = \phi$:

$$\nabla_{x_d} N^*(x_d, u_d) + \left[\frac{\partial f^*(x_d, u_d)^\top}{\partial x_d} - I_n \right]\phi = 0 \qquad \textbf{(10,32)}$$

Notice that ϕ as given by (10,28) will exist only if

$$\det \left[I_n - \frac{\partial f^*(x_d, u_d)^\top}{\partial x_d} \right] \neq 0 \qquad \textbf{(10,33)}$$

which is guaranteed by assumption A10.2.1 (A10.2.1 says that the system is supposed to be asymptotically stable at (x_d, u_d), hence the magnitude of each eigenvalue of $\partial f^*/\partial x$ is lower than one. Notice that the only way the determinant (10,33) is equal to zero is that $\partial f^*/\partial x$ has at least one eigenvalue $\lambda_i = 1$).

From (10,21) we have, using (10,29) and A10.4.2:

$$\nabla_{u_d} L^*(x_d, u_d) + \frac{\partial f^*(x_d, u_d)^\top}{\partial u_d}\phi + \frac{\partial C(u_d)^\top}{\partial u_d}\Theta = 0 \qquad \textbf{(10,34)}$$

From (10,22) we have, in the steady state,

$$f^*(x_d,u_d) - x_d = 0 \tag{10,35}$$

Finally, from (10,24)

$$\Theta \begin{cases} =0 & C(u_d)<0 \\ \geq 0 & C(u_d)=0 \end{cases} \tag{10,36}$$

We note that (10,32), (10,34), (10,35) and (10,36) are consistent with the necessary optimality conditions of SSOP (10,15), (10,16), (10,17) and (10,18). Moreover, from A10.4.3 we have:

$$\begin{aligned} (x_d,u_d) &= (x_s,u_s) \\ \phi &= \mu \\ \Theta &= \lambda \end{aligned} \tag{10,37}$$

Q.E.D.

10.4.2 Development of a steady-state optimiser based on the above analysis

Assume the following structure for the dynamic objective function L^*

$$L^*(x,u) = N^*(x,u) + \tfrac{1}{2}\Delta u(t)^T R \Delta u(t) \tag{10,38}$$

which satisfies A10.4.2, where $R \geq 0$ is a symmetric matrix of the appropriate dimensions and $\Delta u(t) = u(t) - u(t-1)$.

For notational simplicity, define:

$$\begin{aligned} A^* &= \frac{\partial f^*(x,u)}{\partial x}\Big|_{x(t),u(t)} \\ B^* &= \frac{\partial f^*(x,u)}{\partial u}\Big|_{x(t),u(t)} \end{aligned} \tag{10,39}$$

using $N = 1$, $p(t+1) = \phi$ and assuming in order to ease the analysis that no constraints are saturated we have from (10,21):

$$R\Delta u(t) + \nabla_u N^*(x,u) + B^{*\top}\phi \mid_{x(t),u(t)} = 0 \qquad \textbf{(10,40)}$$

but from (10,28):

$$\phi(t) = \left[I_n - \frac{\partial f^{*\top}}{\partial x}\right]^{-1} \nabla_x L^* = \left[I_n - A^{*\top}\right]^{-1} \nabla_x N^* \mid_{x(t),u(t)} \qquad \textbf{(10,41)}$$

Then from (10,40):

$$\Delta u(t) = -R^{-1}\left[B^{*\top}\left[I_n - A^{*\top}\right]^{-1} \nabla_x N^* + \nabla_u N^* \right]_{x(t),u(t)} \qquad \textbf{(10,42)}$$

Thus

$$u(t) = -R^{-1}\left[B^{*\top}\left[I_n - A^{*\top}\right]^{-1} \nabla_x N^* + \nabla_u N^* \right]_{x(t),u(t)} + u(t-1) \qquad \textbf{(10,43)}$$

However, in order to make the above control law realizable, we need the right hand side of (10,43) to be in terms of known quantities at time t. Now, we approximate $\nabla_{u(t)} N^*(x(t),u(t)) \approx \nabla_{u(t-1)} N^*(x(t),u(t-1))$ (this is exact in the steady state), and furthermore, we use the estimates of the derivatives $\hat{A} \approx A^*$ and $\hat{B} \approx B^*$ rather than their exact values. Therefore we have:

$$u(t) = -R^{-1}\left[\hat{B}^{\top}\left[I_n - \hat{A}^{\top}\right]^{-1} \nabla_x N^*(.) + \nabla_u N^*(.) \right]_{x(t),u(t-1)} + u(t-1) \qquad \textbf{(10,44)}$$

This is a recursive control law. Notice that at time t we have available the current state estimate $x(t)$, estimates of the derivatives A^* and B^*, and also the control applied to the plant in the last sample interval $u(t-1)$. Also notice that in the steady state $u(t) = u(t-1)$ and from Theorem 10.4.1 the plant under control law (10,44) will reach the steady-state optimum (x_s, u_s) solution of SSOP.

Assume now that there are only bound constraints on the control signal

$$u_{\min} \leq u(t) \leq u_{\max} \tag{10,45}$$

Define the following saturation function:

$$sat(s, s_{\min}, s_{\max}) = \begin{cases} s_{\max} & s \geq s_{\max} \\ s & s_{\min} < s < s_{\max} \\ s_{\min} & s \leq s_{\min} \end{cases} \tag{10,46}$$

where s is a vector with bounds $s_{\max}$, $s_{\min}$, and where every element s of is saturated independently.

The above analysis gives rise to the following control algorithm, noticing that the set-point update period of the controller, N_e, may be greater than the sampling rate of the identifier.

Algorithm 10.4.2: Steady-state optimiser based on dynamic information (DSSO)

Data: N^*, R, $u_{\max}$, $u_{\min}$, k_v, N_e and estimates of A^*, B^*, and $x(t)$.

Step 0 Initialize controller, set t=0.

Step 1 Obtain values of $x(t)$, $\hat{A}$, $\hat{B}$. Then compute the next control candidate $\hat{u}(t)$ from:

$$\hat{u}(t) = sat\left(u(t), u_{\min}, u_{\max}\right) \tag{10,47}$$

where $u(t)$ is given by (10,44).

Step 2 Filter the next control candidate with:

$$u(t) = u(t-1) + k_v(\hat{u}(t) - u(t-1))$$

where $k_v \in (0,1]$ is a scalar relaxation gain, to obtain the next control $u(t)$ and apply it to the plant. Wait until next updating time, then set $t = t+N_e$ and go to Step 1.

10.4.3 Analysis of the converged Algorithm 10.4.2 when the control is saturated

Assume a single control for simplicity and only an upper bound. In this case we have:

$$C(u) = u - u_{\max} \leq 0$$
$$\frac{\partial C}{\partial u} = 1 \qquad \textbf{(10,49)}$$

Assume that the system under control Algorithm 10.4.2 has achieved steady state and that the constraint (10,49) is active so that:

$$u_d = sat\left(-R^{-1}\left[\ B^{*\top}\left[I_n - A^{*\top}\right]^{-1}\nabla_x N^*(.) + \nabla_u N^*(.)\ \right]\Big|_{(x_d,u_d))} + u_d\ \right) = u_{\max} \qquad \textbf{(10,50)}$$

then we have:

$$-R^{-1}\left[\ B^{*\top}\left[I_n - A^{*\top}\right]^{-1}\nabla_{x_d} N^*(x_d,u_d) + \nabla_{u_d} N^*(x_d,u_d)\ \right] \geq 0 \qquad \textbf{(10,51)}$$

which implies

$$B^{*\top}\left[I_n - A^{*\top}\right]^{-1}\nabla_{x_d} N^*(x_d,u_d) + \nabla_{u_d} N^*(x_d,u_d) \leq 0 \qquad \textbf{(10,52)}$$

But from (l0,37)

$$\phi = \left[I_n - \frac{\partial f^*(x_d, u_d)^\tau}{\partial x_d} \right]^{-1} \nabla_{x_d} N^*(x_d, u_d) \qquad \textbf{(l0,53)}$$

Then we have from (l0,52)

$$\frac{\partial f^*(x_d, u_d)^\tau}{\partial u_d} \phi + \nabla_{u_d} N^*(x_d, u_d) \leq 0 \qquad \textbf{(l0,54)}$$

But from (l0,21) we have that at the steady state, the dynamic optimum satisfies

$$\frac{\partial f^*(x_d, u_d)^\tau}{\partial u_d} \phi + \nabla_{u_d} N^*(x_d, u_d) + \frac{\partial C(u_d)^\tau}{\partial u_d} \Theta = 0 \qquad \textbf{(l0,55)}$$

given (l0,50), (l0,54) and (l0,55) we conclude that $\Theta \geq 0$, which satisfies the Kuhn-Tucker condition (l0,24). Given the equivalence ensured by Theorem 10.4.1, we conclude that the simple saturation scheme of Algorithm 10.4.2 yields the steady state optimum solution of SSOP (with control bound constraints) when Algorithm 10.4.2 converges. The analysis for the multivariable case is similar.

10.4.4 DSSO Algorithm with rate constraints

Assume that in addition to the bound constraints (l0,45) the rate of variation of the manipulated variables is also constrained by:

$$-\Delta u_{\max} \leq \Delta u(t) \leq \Delta u_{\max}$$

where $\Delta u(t) = u(t) - u(t-1)$.

In this case the application of the relaxation filter in Algorithm 1 becomes unnecessary and we can formulate the following DSSO algorithm with rate constraints:

Algorithm 10.4.4: DSSO with rate constraints

Data: N^*, R, $u_{\max}$, $u_{\min}$, $\Delta u_{\max}$, N_e and estimates of A^*, B^*, and $x(t)$.

Step 0 Initialize controller, set t=0.

Step 1 Obtain values of $x(t)$, $\hat{A}$, $\hat{B}$. Then compute the next control change $\Delta u(t)$ from:

$$\Delta u(t) = sat\left(\Delta v(t), -\Delta u_{max}, \Delta u_{max}\right) \qquad \textbf{(10,57)}$$

where

$$\Delta v(t) = -R^{-1}\left[\hat{B}^{T}\left[I_n - \hat{A}^{T}\right]^{-1}\nabla_x N^*(x,u) + \nabla_u N^*(x,u)\right]_{x(t),u(t-1)} \qquad \textbf{(10,58)}$$

Step 2 Compute

$$u(t) = sat(\, u(t-1) + \Delta u(t),\, u_{min},\, u_{max}\,)$$

to obtain the next control $u(t)$ and apply it to the plant. Wait until next sampling time, then set $t=t+N_e$ and go to Step 1.

10.4.5 Comments on the structure of the control law

The structure of the control law (10,44) may be decomposed in the following terms:

$$u(t) = u(t-1) - R^{-1}\,\nabla_u \mathcal{L}|_{x(t),u(t-1)} \qquad \textbf{(10,60)}$$

where the gradient is given by

$$\nabla_u \mathcal{L} = \hat{B}^{T}\phi + \nabla_u N^*\,|_{x(t),u(t-1)} \qquad \textbf{(10,61)}$$

and ϕ is given by:

$$\phi = \left[I_n - \hat{A}^{T}\right]^{-1}\nabla_x N^*|_{x(t),u(t-1)} \qquad \textbf{(10,62)}$$

Notice that, neglecting any constraints, gradient (10,61) is an approximation to the steady-state gradient (10,16), with an estimation of μ given by (10,62), using the dynamic derivative estimates ($\hat{A}$, $\hat{B}$), and evaluated at the instantaneous operating

point $(x(t), u(t-1))$. This instantaneous operating point is not necessarily the solution of the steady state mapping $g^*(x,u) = 0$, but corresponds to the dynamic response of the system governed by $x(k+1) = f^*(x(k), u(k))$. Notice, also, that under the assumptions of Theorem 10.4.1, gradient (l0,61) is equivalent to gradient (l0,16) in the steady state. Then DSSO can be regarded as a gradient-descent optimising algorithm which leads the plant to the steady-state optimum through an unfeasible path from the steady-state point of view (but feasible from a dynamic perspective). It is this infeasibility that gives DSSO the potential to achieve the steady-state optimum faster than other on-line optimising methods in which steady-state feasibility is required (and hence long settling times).

Notice that the structure of the control law in DSSO is thus similar to the gradient-type control law of the adaptive on-line optimisation algorithm originally proposed by Bamberger and Isermann (1978) and later modified and used by Garcia and Morari (1981) and Rolf and Henry (1984). The main differences between DSSO and the techniques described in the above references are:

* DSSO is derived from a state-space perspective using constrained optimisation theory and discrete optimal control.
* The states used in DSSO are derived from the shadow model of the plant and not merely from output measurements.
* DSSO explicitly takes into account input dependent constraints and rate constraints.

10.5 SECOND APPROACH: LONG RANGE PREDICTION WITH A LINEARIZED STATE-SPACE MODEL OF THE PROCESS (DISOPE AS A STEADY-STATE OPTIMISER)

We have discussed in Chapter 9 the application of DISOPE as a dynamic optimiser in nonlinear predictive control. A nonlinear model of the plant was then used for the long range predictions. However, as we discussed in Section 10.1, it may be the case that a nonlinear model of the process is not available for prediction purposes. In this is case, instead of a nonlinear model, a linearized state-space model

of the process may be identified as shown in Section 10.1. In this section, we will explore the possibility of using a linearized model for performing the long range predictions and dynamic optimisation. The fact that this linear model is locally valid and that its parameters are adapted may be exploited in order to gradually achieve the steady state optimum of the plant. The necessary information is readily available from the performance index specifications, the values of state variables and the adaptive matrix coefficients of the linear model. We propose here to force the state and control variables to remain reasonably close (along the prediction horizon) to the current operating point from which the predictions are started, so that the predictions performed with the linear model remain valid. A clear advantage of the long range predictive control approach is that the saturation of (state and control dependent) constraints may be anticipated and appropriate action may be taken by the controller in advance. Predictions may also be displayed to plant operators for different purposes. The economic objectives are included in the dynamic performance index, so that no separate optimising algorithm is required.

10.5.1 Formulation

Based on the information available (see Section 10.1) a locally valid dynamic model of the process may be written as follows, in terms of incremental variables:

$$\Delta x(t+j) = \hat{A}\Delta x(t+j-1) + \hat{B}\Delta u(t+j-1) \qquad \textbf{(10,63)}$$

where $\hat{A}$, $\hat{B}$ are the estimated matrices as explained in Section 10.1, $x(t)$ is the current state of the process, and the incremental state and control variables are given by:

$$\begin{aligned} \Delta x(k) &= x(k) - x(k-1) \\ \Delta u(k) &= u(k) - u(k-1) \end{aligned} \qquad \textbf{(10,64)}$$

Suppose now that the steady state objective function is $N^*(x,u) : \Re^n \times \Re^m \rightarrow \Re$, that the control vector is bounded between upper and lower levels $u_{\min} \leq u(t) \leq u_{\max}$, and that $\Psi(x) \geq 0$, $\Psi: \Re^n \rightarrow \Re^c$ is a given set of state dependent inequality constraints. Thus we define the real optimal control problem to be solved by DISOPE in a receding horizon fashion as:

ROP10

$$\min_{\substack{\Delta u(k) \\ k \in [t, t+N-1]}} \frac{1}{2}\Delta x(t+N)^T \Phi \Delta x(t+N) + \sum_{k=t}^{t+N-1} [N^*(x(k),u(k)) + \frac{1}{2}\Delta x(k)^T Q \Delta x(k) + \frac{1}{2}\Delta u(k)^T R \Delta u(k) + P(x(k),u(k))]$$

subject to

$$\Delta x(k+1) = \hat{A}\Delta x(k) + \hat{B}\Delta u(k)$$
$$\Delta x(t) = x(t) - x(t-1)$$
$$-\Delta u_{\max} \leq \Delta u(k) \leq \Delta u_{\max}$$
$$k \in [t, t+N-1]$$

where

$$x(k) = x(t-1) + \sum_{j=t}^{k} \Delta x(j)$$
$$u(k) = u(t-1) + \sum_{j=t}^{k} \Delta u(j) \qquad \textbf{(10,65)}$$

$x(t)$ and $u(t-1)$, the current state and control variables, are known, $x(t-1)$ is also known, Φ, Q and R are weighting matrices of the appropriate dimensions, N is the prediction horizon, $P(x(k),u(k))$ is a penalty term dependent on the saturation of control magnitude constraints $u_{\min} \leq u(k) \leq u_{\max}$ and on the activation of state dependent constraints $\Psi(x) \geq 0$ (See Chapter 7). Notice that the decision variables are control *increments* rather than absolute values and, hence, rate constraints on the controls may be handled by a saturation function as explained in Chapter 3.

The model-based problem is chosen so as to use convenient linear-quadratic methods in the model-based calculations.

MOP10

$$\min_{\substack{\Delta u(k) \\ k\in[t,t+N-1]}} \quad J = \tfrac{1}{2}\Delta x(t+N)^{\top}\,\Phi\,\Delta x(t+N) + \sum_{k=t}^{t+N-1} [\ \tfrac{1}{2}\Delta x(k)^{\top} Q \Delta x(k) + \tfrac{1}{2}\Delta u(k)^{\top}\ R\ \Delta u(k) + \gamma(k)\ ]$$

subject to

$$\Delta x(k+1) = A_m \Delta x(k) + B_m \Delta u(k) + \alpha(k)$$
$$\Delta x(t) = x(t) - x(t-1)$$
$$k\in[t,t+N-1]$$

where A_m and B_m are model-based dynamic matrices of the appropriate dimensions. Then Algorithm 5.3.3 in conjunction with Procedure 5.3.1 may be used for solving ROP10. Notice that the values of the model-based matrices A_m and B_m may be different from those of the reality matrices $\hat{A}$ and $\hat{B}$. In the case of using batch identification with a moving data window, it may be convenient to have two sets of identifiers operating in parallel but with different update periods. The identifier with the shorter update period would provide the matrices $\hat{A}$ and $\hat{B}$ and that with the longer update period would provide the matrices A_m and B_m. This may provide savings in terms of computational time since the Riccati matrices $G(k)$ and $S(k)$ would only need updating when the model-based matrices A_m and B_m are updated (See Procedure 5.3.1). The use of two identifiers would be justified in the case when the additional burden of the second identifier is lower than the savings of computation time in the optimisation calculations.

Based on the above discussion, the following predictive control algorithm is proposed. It is called LP-DISOPE (Linear-Predictive DISOPE). Notice that the set-point update period of the controller, N_e, may be greater than the sampling rate of the identifier. This may be convenient since the computational burden of the optimiser is reduced by increasing N_e without affecting the optimality of the steady-state solution.

Algorithm 10.5.1: Predictive optimising algorithm based on DISOPE and an adaptive linear model of the process (LP-DISOPE)

Data: N^*, R, $u_{\max}$, $u_{\min}$, N, N_e, the latest applied control $u(t-1)$ and estimates of A^*, B^*, and $x(t)$, $x(t-1)$.

Step 0 Initialize controller, set t=0.

Step 1 Obtain values of $x(t)$, $\hat{A}$, $\hat{B}$, A_m, B_m. Solve ROP10 by using Algorithm 5.3.3 to obtain the predicted control sequence $u(t)...u(t+N-1)$, and state predictions $x(t)...x(t+N)$. Apply the first control $u(t)$ to the plant.

Step 2 Wait until next set-point update time, then set $t = t+N_e$ and repeat from step 1.

10.5.2 Optimality

Now the optimality of the steady-state solution of Algorithm 10.5.1 will be analyzed. For the sake of simplicity, we will neglect state and control dependent constraints in this analysis. Thus we intend to solve a steady state optimisation problem like SSOP (see Section 10.4) without any inequality constraints. The following assumptions will be made:

A10.5.2.1 Assume that by applying to the plant (10,7) a control sequence $\{u(0), u(1) .... u(t_s)\}$ computed from the solution of Algorithm 10.5.1, the system achieves a steady-state (x_d, u_d) (in practical terms) for some finite time t_s.

A10.5.2.2 Assume that the solution of SSOP (x_s, u_s) exists and is unique.

A10.5.2.3 Assume that the estimation of the derivatives of $f^*(x,u)$ with respect to x and u is perfect in the steady state.

A10.5.2.4 Assume that derivatives of $f^*(x,u)$ and $N^*(x,u)$ with respect to their arguments exist and are Lipschitz continuous.

A10.5.2.5 Assume that $[\hat{A},\hat{B}]$ is stabilizable (i.e. the unstable poles of $\hat{A}$ may be manipulated via linear state feedback). Suppose also that $[\hat{A},\sqrt{Q}\,]$ is observable (i.e. all the modes of $\hat{A}$ may be observed through the fictitious output matrix $\sqrt{Q}\,$) and that $R>0$, $Q\geq 0$.

Preliminarily, we will state the following result (see Lewis, (1986a) for the proof):

Theorem 10.5.2.1 (Lewis, 1986a) - Assume $R>0$, $Q\geq 0$ and suppose that $[\hat{A},\sqrt{Q}\,]$ is observable. Consider the Riccati Difference Equation

$$\begin{aligned} S(k) &= Q + A^{\top}S(k+1)(A-BG(k)), \quad S(N) = \Phi \\ G(k) &= [R+B^{\top}S(k+1)B]^{-1}B^{\top}S(k+1)A \end{aligned} \tag{10,66}$$

associated with an LQ optimal control problem. Then $[A,B]$ is stabilizable if and only if :

a. There is an unique positive definite limiting solution S_∞ to the Riccati difference equation. Furthermore S_∞ is the unique positive definite solution to the Algebraic Riccati equation.

$$S = A^{\top}(S-SB(B^{\top}SB+R)^{-1}B^{\top}S)A+Q \tag{10,67}$$

b. The closed loop plant:

$$x(k+1) = (A-BG_\infty)x(k) \tag{10,68}$$

is asymptotically stable, where G_∞ is the limiting solution to $G(k)$.

The achievement of the solution of SSOP via Algorithm 10.5.1 is given by the following theorem.

Theorem 10.5.2.2 - Optimality of the steady-state solution of Algorithm 10.5.1

Under assumptions A10.5.2.1-A10.5.2.5, if the prediction horizon $N \to \infty$, then the steady state solution (x_d, u_d) of Algorithm 10.5.1 satisfies also the necessary optimality conditions of SSOP.

Proof

Neglecting inequality constraints, ROP10 may be re-written as follows

ROP10b

$$\min_{\substack{\bar{u}(k) \\ k\in[t,t+N-1]}} \frac{1}{2}\bar{x}(t+N)^{\top}\Phi\bar{x}(t+N) + \sum_{k=t}^{t+N-1} [\, N^{*}(\bar{x}(k)+x(k-1),\bar{u}(k)+u(k-1)) + \frac{1}{2}\bar{x}(k)^{\top}Q\bar{x}(k) + \frac{1}{2}\bar{u}(k)^{\top}R\bar{u}(k) \,]$$

subject to

$$\bar{x}(k+1) = \hat{A}\bar{x}(k) + \hat{B}\bar{u}(k), \quad \bar{x}(t) = x(t)-x(t-1)$$

where $\bar{x}(k) = x(k) - x(k-1)$ and $\bar{u}(k) = u(k) - u(k-1)$.

The Hamiltonian of ROP10b may be written as follows

$$H^{*} = N^{*}(\bar{x}(k)+x(k-1),\bar{u}(k)+u(k-1)) + \frac{1}{2}\bar{x}(k)^{\top}Q\bar{x}(k) + \frac{1}{2}\bar{u}(k)^{\top}R\bar{u}(k) + p(k+1)^{\top}(\hat{A}\bar{x}(k)+\hat{B}\bar{u}(k)) \quad \textbf{(10,69)}$$

Now, the necessary optimality conditions of ROP10b are:

From $\nabla_{\bar{u}(k)}H^{*} = 0$ we obtain:

$$\bar{u}(k) = -R^{-1}(\hat{B}^{\top}p(k+1) + \nabla_{\bar{u}(k)}N^{*}) \quad \textbf{(10,70)}$$

From $\nabla_{\bar{x}(k)} H^* = p(k)$ we obtain:

$$Q\bar{x}(k) + \nabla_{\bar{x}(k)} N^* + \hat{A}^T p(k+1) = p(k) \qquad \textbf{(10,71)}$$

From $\nabla_{\bar{p}(k)} H^* = \bar{x}(k+1)$ we obtain:

$$\bar{x}(k+1) = \hat{A}\bar{x}(k) + \hat{B}\bar{u}(k), \; \bar{x}(t) = 0 \qquad \textbf{(10,72)}$$

But the fact that the system has achieved a steady state implies that $\bar{u}(k) = 0$, $k = t, t+1...$ and as a consequence $\bar{x}(k) = 0$, $k = t, t+1....$ The solution to the TPBVP defined by (10,71) and (10,72) is given by:

$$p(k) = S(k)\bar{x}(k) + h(k) \qquad \textbf{(10,73)}$$

where $S(k)$ is the solution of the Riccati Difference Equation and $h(k)$ is the solution of the following difference equation (see, for instance, Procedure 5.3.1):

$$h(k) = (A - BG(k))^T h(k+1) + (\nabla_{\bar{x}(k)} N^* - G(k)^T \nabla_{\bar{u}(k)} N^*), \; h(N) = 0 \qquad \textbf{(10,74)}$$

Since $N \to \infty$, then from assumption A10.5.2.5 and Theorem 10.6.2.1 $S(k)$ has a limiting solution S_∞ and the matrix $(A - BG_\infty)$ has all its eigenvalues inside the unit circle, where G_∞ is the limiting solution of $G(k)$. Therefore, $h(k)$ also has a limiting solution h_∞ since the non-homogeneous term in (10,74) is constant and bounded. Furthermore, it follows that the costate is constant.

Given that $\bar{u}(k) = 0$, $\bar{x}(k) = 0$ and $p(k+1) = p(k) = \mu$ (a constant), using the chain rule to find that $\nabla_{\bar{x}(k)} N^* = \nabla_{x(k)} N^*$ and $\nabla_{\bar{u}(k)} N^* = \nabla_{u(k)} N^*$, and exploiting A10.5.2.3 to substitute $\hat{A} = \dfrac{\partial f^*}{\partial x_d}$ and $\hat{B} = \dfrac{\partial f^*}{\partial u_d}$, we have from (10.70) and (10,71):

$$\frac{\partial f^{*\top}}{\partial u_d}\mu + \nabla_{u_d} N^*(x_d, u_d) \qquad \textbf{(10,75)}$$

$$\nabla_{x_d} N^*(x_d, u_d) + (\frac{\partial f^{*\top}}{\partial x_d} - I_n)\mu = 0 \qquad \textbf{(10,76)}$$

which are in turn the necessary optimality conditions of SSOP (10,15) and (10,16) without inequality constraints, noticing that (10,17) is satisfied by assumption A10.5.2.4.

Q.E.D.

10.5.3 Practical considerations

It was convenient for the purposes of the above analysis to assume that the prediction horizon $N \to \infty$. However, in practice, a finite choice of N is mandatory. A recommended value of the prediction horizon is $N \approx (2...5)\tau_d / T_s$, where τ_d is the dominant time constant of the process and T_s is the sampling period of the identifier.

It is possible to define a control horizon $N_c < N$ after which the control signals are held constant, as is done is other predictive controllers (Soeterboek, 1992). This is achieved by using a large value of the control increment weighting matrix $R \to \infty$, for $k > N_c$.

Regarding the values of the quadratic weighting matrices $\{ \Phi, Q, R \}$, they must be chosen so as to obtain an adequate speed of response.

If the set-point update period N_u is greater than one, it is possible to apply to the plant the average of the first N_u predicted controls.

In order to avoid numerical ill-conditioning, the steady-state objective, state variables, control variables and constraints should be properly scaled (see, for instance, Fletcher (1981)).

The maximum of speed of response of the set-points should not only take account of any physical rate constraints associated with the process, but also such speed of response should be in agreement with the ability of the identifier to properly adapt the parameters in the face of changes in the operating point.

Notice that provided the steady-state objective function $N^*(x,u)$ is linear (which is common if the objective function is based on economic criteria) and if no inequality constraints are active, then the solution of ROP10 in Step 1 of LP-DISOPE (Algorithm 10.5.1) requires only one iteration of Algorithm 5.3.3. This is a computational advantage. When inequality constraints become active, then the necessary number of iterations increase.

In the formulation of Theorem 10.5.2.2 it was assumed that no inequality constraints were saturated at the optimum. However, the results obtained, as regards the optimality of the steady-state solution of Algorithm 10.5.1, are thought to be true even for the constrained case. Simulation experience confirms this belief.

If a simple procedure such as the clipping-off of unconstrained solutions was used for handling control magnitude constraints (as is done with DSSO), LP-DISOPE would still drive the process to its steady-state optimum (this can be proved by an analysis similar to that presented in Section 10.4.3).

10.5.4 A link between LP-DISOPE and DSSO

An interesting link between the dynamic predictive optimiser based on DISOPE and the dynamic optimiser DSSO is given in the following Theorem.

Theorem 10.5.4: Neglecting inequality constraints and assuming: (a) $N \rightarrow \infty$, $Q = \Phi = 0$, $\hat{A} = A_m$ and $\hat{B} = B_m$ in ROP10 and MOP10; (b) $\hat{A}$ has all its eigenvalues inside the unit circle; (c) the steady-state performance index is linear $N^*(x,u) = q^T x + r^T u$, then the control law provided by the solution of ROP10 is equivalent to the control law in DSSO, Equation (10,44).

Proof: The equivalence follows by substitution in Procedure 6.2.1. Taking into account that $Q = \Phi = 0$ implies $S(k) = 0$, $G(k) = 0$ and since $\hat{A}$ is assumed to be stable then $h(k)$ has a limiting solution $h_\infty = [I_n - \hat{A}^T]^{-1} q$. Furthermore, $\lambda(k) = -r$ and $\beta(k) = -q$. Finally, we obtain the control law $\Delta u(k) = g(k) = -R^{-1}[\hat{B}^T [I_m - \hat{A}^T]^{-1} q + r]$, which is equivalent to (10,44) with a linear objective.

Q.E.D.

10.6 SIMULATION STUDIES

LP-DISOPE and DSSO have been implemented in software, using the C and C++ programming languages, and their performance has been tested with the following simulation examples, including industrial-scale simulations of a multicomponent distillation column using the rigorous process simulator OTISS™ on a UNIX-based computer system at SAST Ltd (U.K.).

Example 10.6.1: Second order nonlinear system

This example consists of the optimisation of a nonlinear second order discrete-time dynamic system with respect to a linear performance index and two manipulated variables. In this case the derivative estimates were computed by using a recursive extended least squares estimator as explained in Section 10.1. In this example, the steady state performance index is given by

$$N^*(x,u) = q_1 x_1 - 2x_2 - 0.4u_1 - u_2 \qquad \textbf{(10,77)}$$

while the dynamics are given by:

$$\begin{aligned} x_1(k+1) &= 0.7x_1(k) + 0.1x_2(k) + 0.1u_1(k) + 0.1u_1(k)x_1(k) \\ x_2(k+1) &= 0.2x_1(k) + 0.1x_2(k) + 0.1u_2(k) - 0.1x_2^2(k) \end{aligned} \qquad \textbf{(10,78)}$$

with initial conditions:

$$x(0) = [0\ 0]^T$$
$$u(0) = [0\ 0]^T \qquad \textbf{(10,79)}$$

and the following constraints:

$$-1 \leq u_i(t) \leq 1$$
$$-1 \leq u_i(t) \leq 1 \qquad \textbf{(10,80)}$$
$$-0.001 \leq \Delta u(t) \leq 0.001$$

In order to illustrate the adaptive features of the DSSO controller, a change in the cost parameter q_1 from 1.00 to 1.05 at discrete time $t = 900$ is simulated.

The results were obtained using an implementation of Algorithm 2 with $R = diag(\ 20\ ,\ 20\ )$. The solution of the corresponding steady state optimisation problem has also been found using a constrained optimisation function of MATLAB's Optimisation Toolbox. The numerical results are shown in Table 10.6.1.1.

Notice that a pseudo-random binary signal of amplitude +/- 0.01 has been added to the inputs in order to enhance identifiability and to avoid the divergence of the estimates. In order to track parameter changes a forgetting factor $\lambda = 0.95$ was used in the estimation algorithm. The presence of the additional excitation noise is reflected in the state responses, which explains that the results obtained by using DSSO have less significant figures than those obtained by using MATLAB's Optimisation Toolbox (see Table 10.7.1.1). In order to provide the controller with good initial estimates, the recursive identifier was started at $t = 0$, while the controller was started at $t = 200$.

Control signals and state trajectories are presented in Figures 10.6.1.1 and 10.6.1.2. The evolution of the objective function is shown in Figure 10.6.1.3.

	DSSO q_1=1	MOTB q_1=1	DSSO q_1=1.05	MOTB q_1=1.05
u_1	0.75	0.7472	0.66	0.6619
u_2	1.00	1.0000	1.00	1.0000
x_1	0.42	0.4205	0.36	0.3635
x_2	0.20	0.2001	0.19	0.1880
N^*	-1.279	-1.279	-1.26	-1.259

Table 10.6.1.1: Optimal results obtained using DSSO and MATLAB's optimisation toolbox.

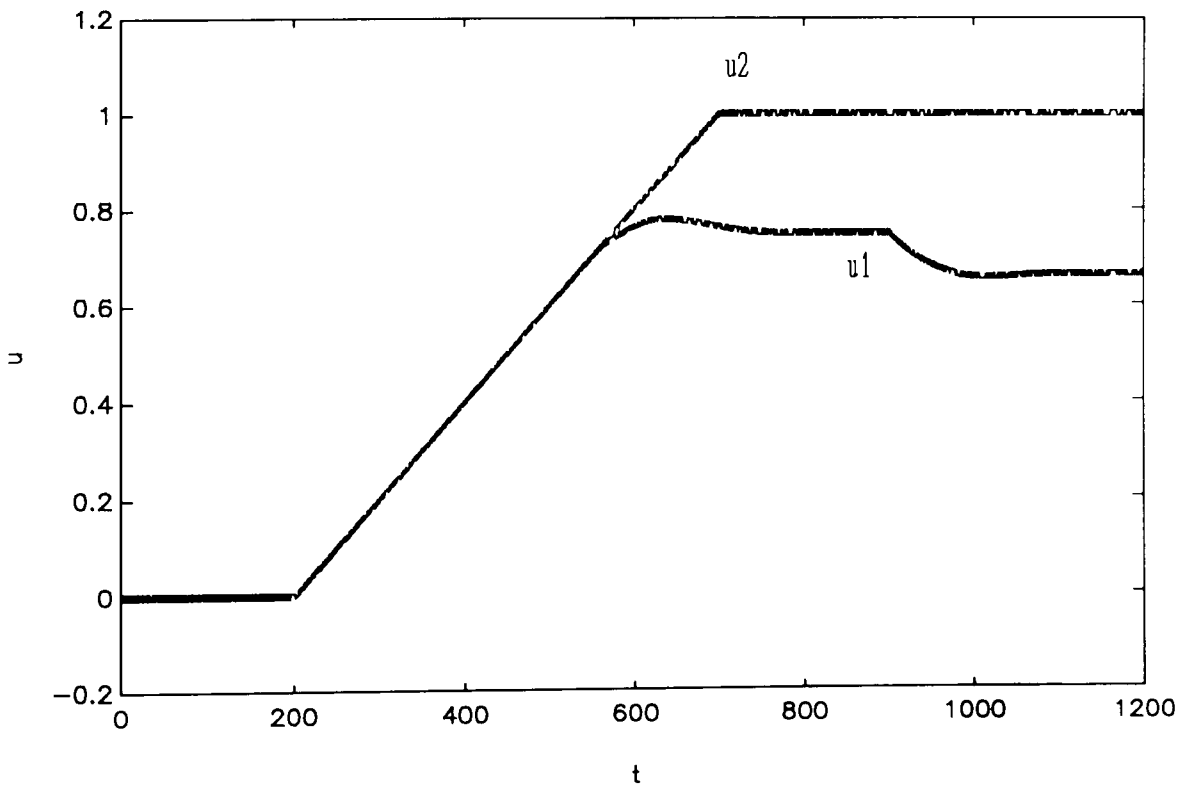

Figure 10.6.1.1: Example 10.6.1, control signals

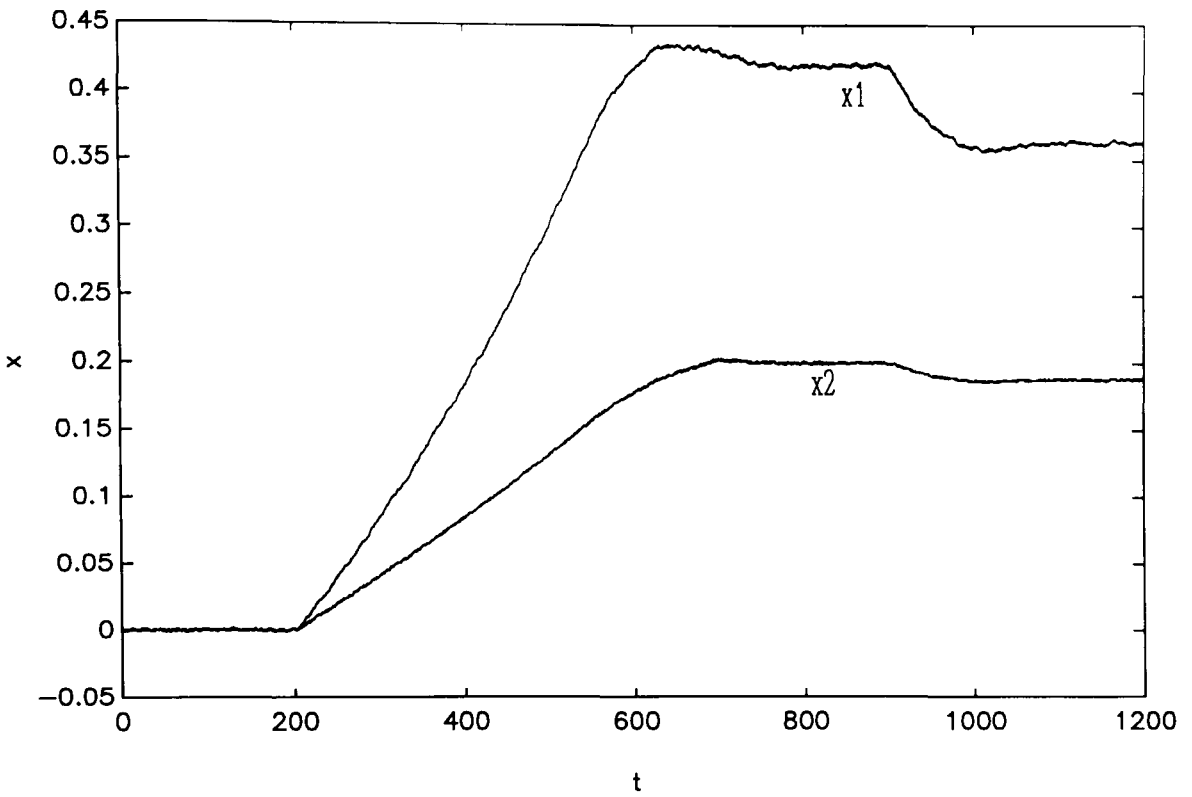

Figure 10.6.1.2: Example 10.6.1, state trajectory

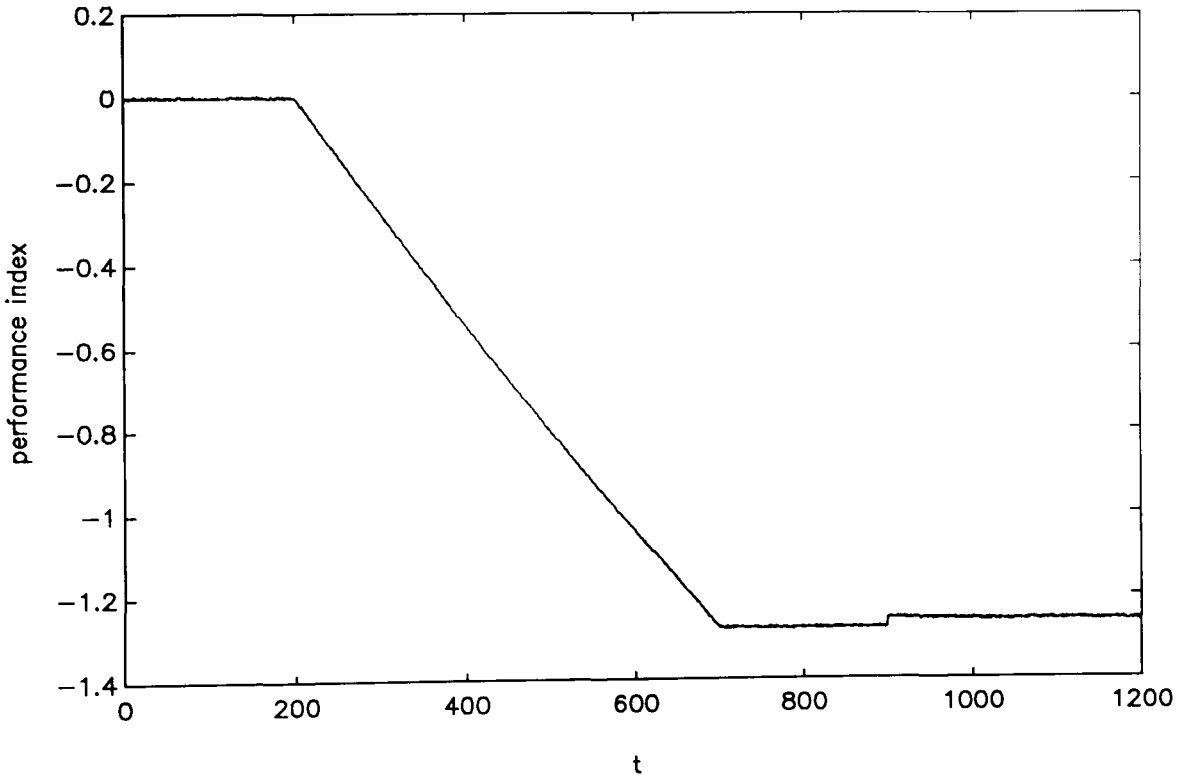

Figure 10.6.1.3: Example 10.6.1, evolution of the performance index

Example 10.6.2: Optimisation of a multicomponent distillation column

This simulation study consists of the optimisation of a multicomponent stabilizer distillation column, having 14 theoretical trays, and for which a rigorous mathematical model exists. The process was simulated using the high fidelity simulator OTISS™. This system provides a library of high fidelity, first principle dynamic models from which a chemical plant can be simulated. Figure 10.6.2.18 shows an schematic diagram of the process. The column normally receives a hydrocarbon feed of 153,000 kg/h at 177 °C.

The manipulated variables for optimisation are the set-points of the following controllers:

TIC4000: Condenser temperature, $u_1 \quad (^\circ C)$

TIC3000: Reboiler temperature, $u_2 \quad (^\circ C)$

PIC1000: Top pressure, $u_3 \quad (bar)$

The control variables are bounded between upper and lower levels:

$$\begin{aligned} 55 \;&\leq\; u_1 \;\leq\; 75 \quad (^\circ C) \\ 255 \;&\leq u_2 \leq 290 \quad (^\circ C) \\ 19 \;&\leq u_3 \leq 24 \quad (bar) \end{aligned} \tag{10,81}$$

In this example, the number of relevant states chosen was $n = 30$, while the actual process model consists of about one thousand of differential equations and several thousands of algebraic equations. The dominant time constant of this process, as computed from the eigenvalues of the identifed system matrix $\hat{A}$ at the initial steady state, is $\tau_d \approx 7.5$ min. The problem has been scaled for its solution

The linear steady state objective function is based on economic criteria and is given by:

$$N^*(x,u) = q_1 x_1 + r_1 u_1 + r_2 u_2 \tag{10,82}$$

where $q_1 = -0.03$ (£/kg), $r_1 = -2.5$ (£/h°C), $r_2 = 2.5$ (£/h°C), and x_1 (kg/h) is the top vapour flow rate.

This steady-state optimisation problem has been previously solved by using the Modified Two-step method (MTS) where the steady-state derivatives have been computed using the system identification method (Lin and Griffiths, 1992). In that study, the optimal solution was achieved after 3 MTS iterations.

Table 10.6.2.1 shows the tuning parameters of LP-DISOPE (case (a)). Table 10.6.2.2 shows the tuning parameters of DSSO (case b). Table 10.6.2.3 shows the tuning parameters of the identifier, noticing that in case (a) only one identifier has been used. A square wave of small magnitude has been added to the set-points so as to improve identifiability. In case (a), the control applied to the plant when the set-points were updated consisted of the average of the first N_e predicted samples.

Case	Φ	Q	R	Control Horizon $N_c T_s$ (min)	Prediction horizon NT_s (min)	Update period $N_e T_s$ (min)
a	0	$10I_{30}$	$3000I_3$	6	15	6

Table 10.6.2.1: LP-DISOPE tuning parameters

Case	R	$\Delta u_{max}(1)$ $°C/h$	$\Delta u_{max}(2)$ $°C/h$	$\Delta u_{max}(3)$ bar/h	Update period $N_e T_s$ (min)
b	$30I_3$	3	3	0.3	4

Table 10.6.2.2: DSSO tuning parameters

Case	Data length $N_d T_s$ (min)	Update period $N_u T_s$ (min)	Sampling period T_s (s)
a	30	18	0.5
b	40	12	0.5

Table 10.6.2.3 : Identifier tuning parameters

The results are presented in Table 10.6.2.4, together with those obtained by the Modified Two-step method and with the corresponding parameters of the design case. Figures 10.6.2.1 to 10.6.2.6 show the actual trajectories of different variables corresponding to case (a). Figure 10.6.2.7 to 10.6.2.11 show the predicted variables from $t=0$ as compared with the actual ones. Figures 10.6.2.12 to 10.6.2.17 show the trajectories of different variables corresponding to case (b). Notice that in every case there is an identification period equal to the data length of the identifier and which is not shown in the above mentioned figures. The initial steady state corresponds with the set-point values of the design case. Notice that in Table 10.6.2.4 the steady-state values for cases (a) and (b) have been averaged using the last ten samples in order to filter the noise present in some signals.

Parameter	Design case	Case a	Case b	MTS method
Condenser temperature u_1 $(°C)$	63	74.9	75	75
Reboiler temperature u_2 $(°C)$	272	284.0	282.1	285.5
Top pressure u_3 (bar)	20.7	19.0	19	19
Top vapour flow rate x_1 (kg/h)	23994	30620	30552	30686
Methane mole fraction of vapour x_2	0.2869	0.2467	0.2475	0.2465
Performance index N^* $(£/h)$	-197	-395.9	-398	-394

Table 10.6.2.4: Results obtained as compared with the design case and the MTS method

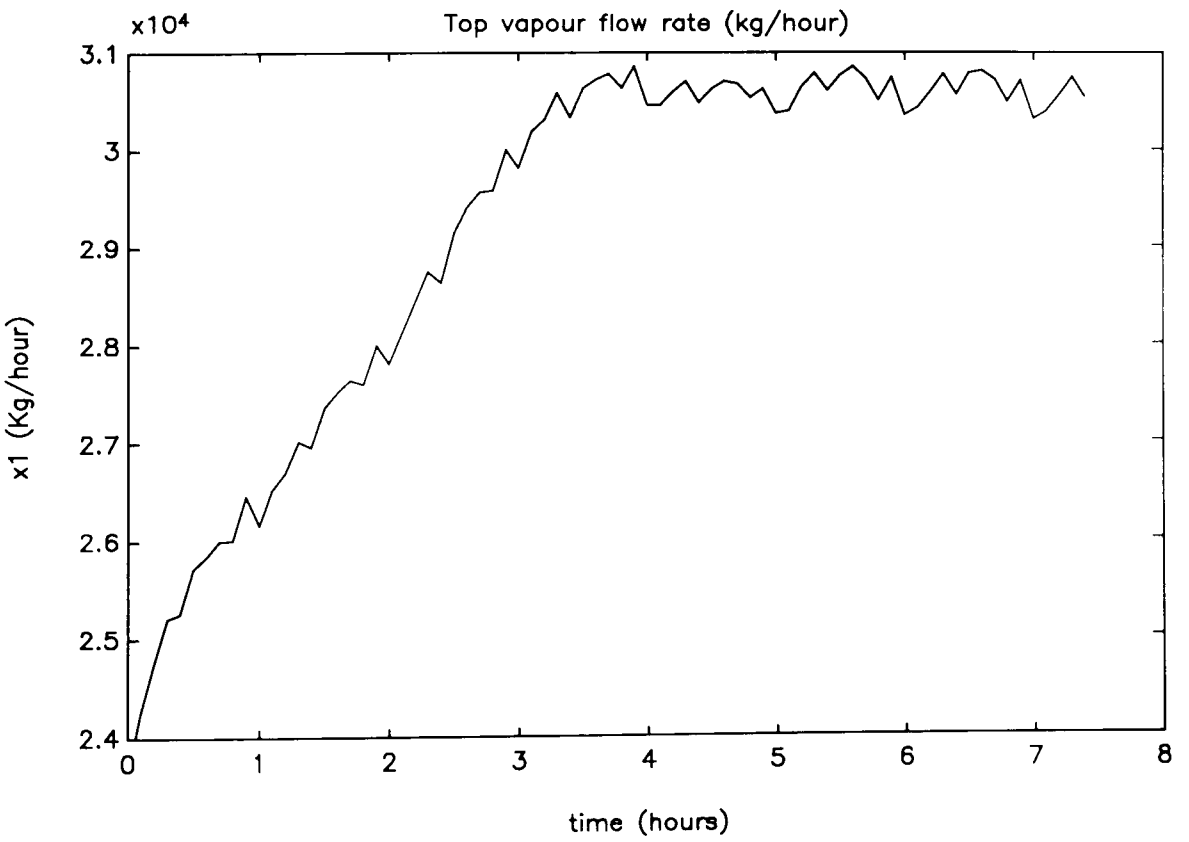

Figure 10.6.2.1: Example 10.6.2.a, Top vapour flow rate trajectory

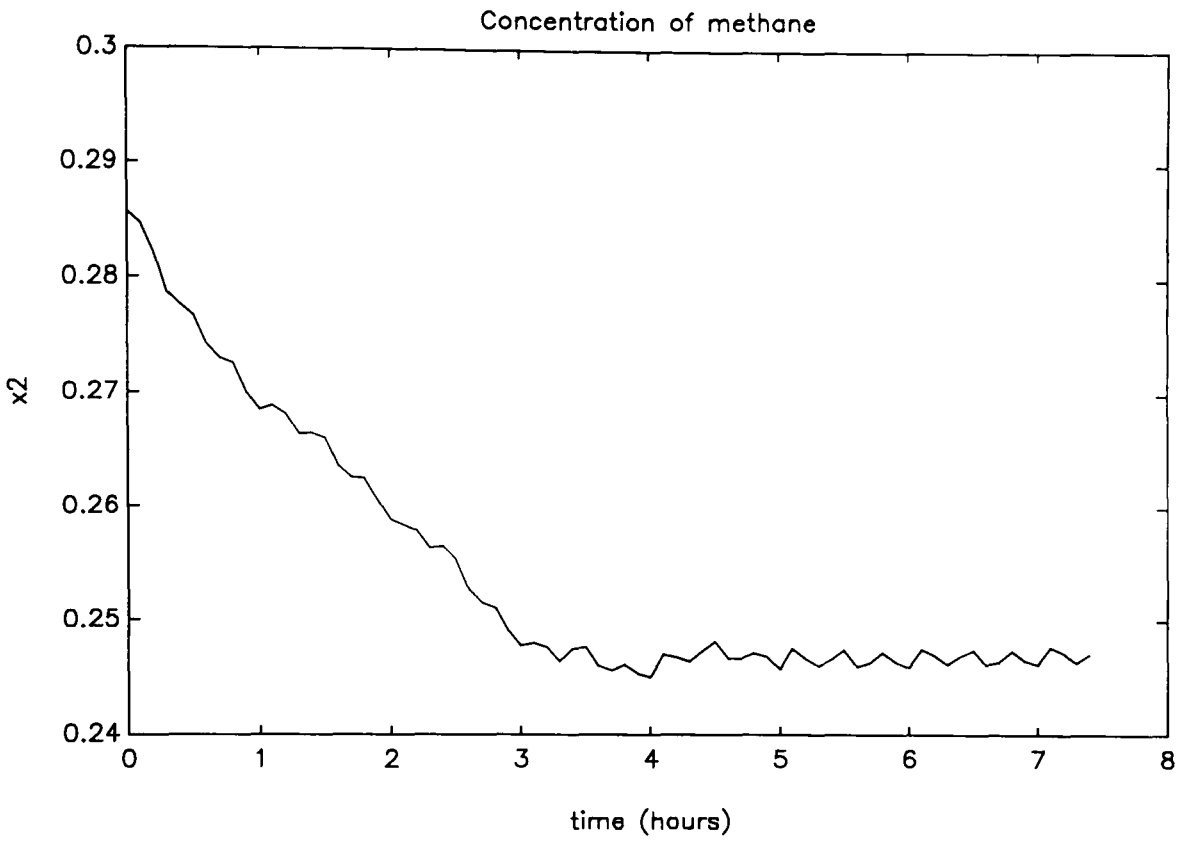

Figure 10.6.2.2: Example 10.6.2.a, concentration of methane trajectory

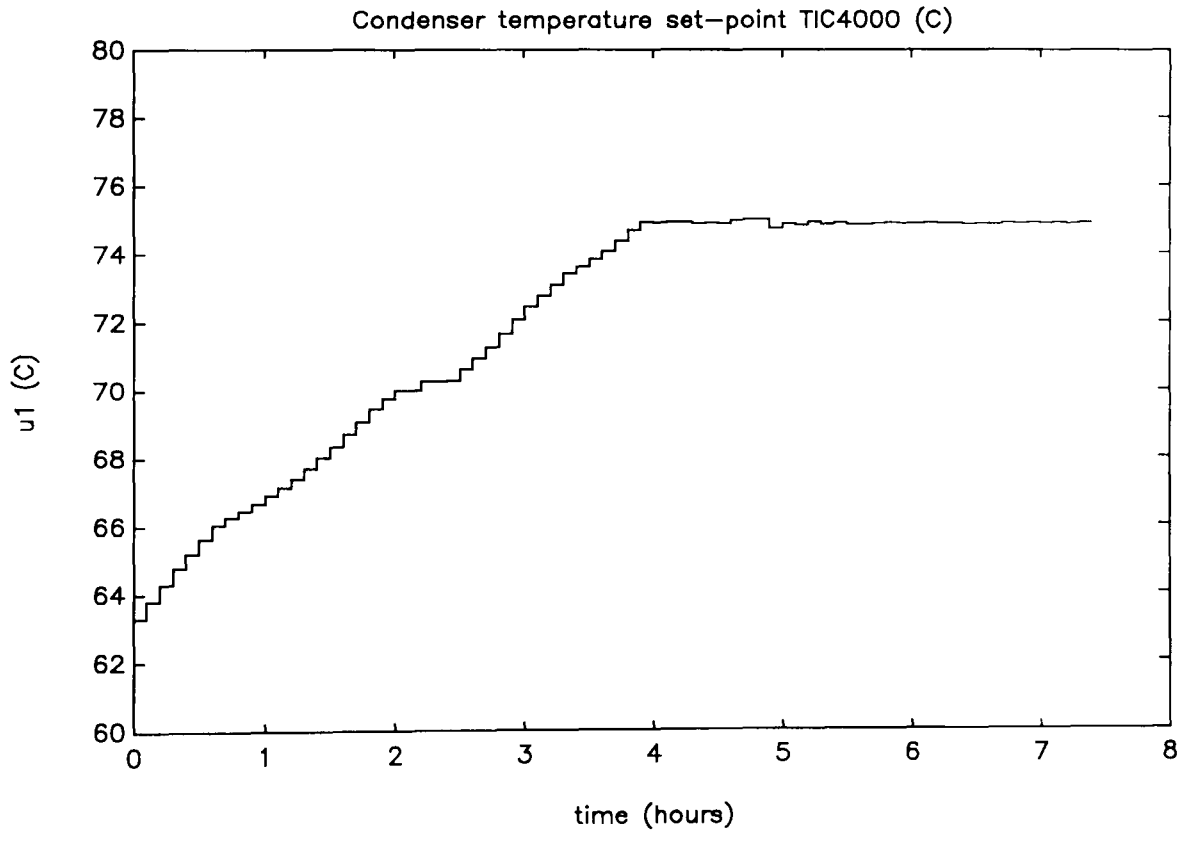

Figure 10.6.2.3: Example 10.6.2.a, condenser temperature controller set-point

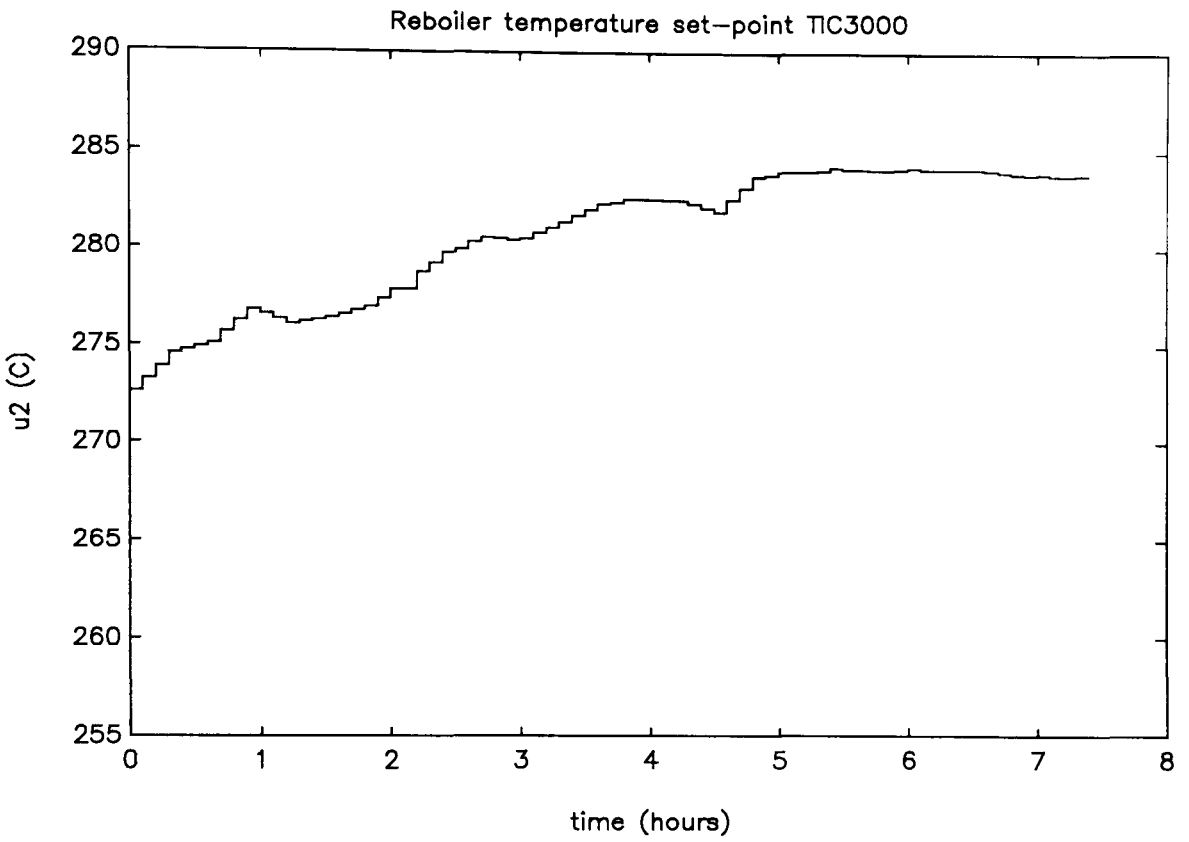

Figure 10.6.2.4: Example 10.6.2.a, reboiler temperature controller set-point

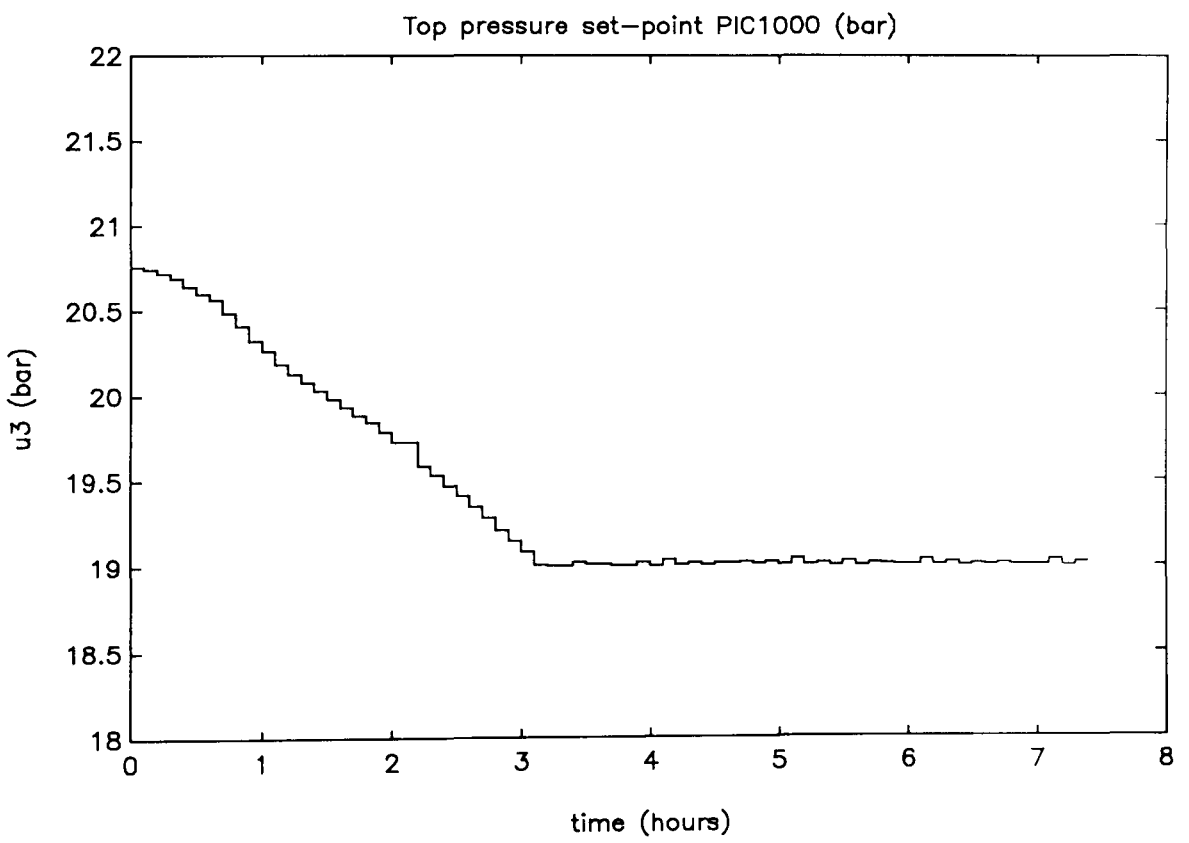

Figure 10.6.2.5: Example 10.6.2.a, top pressure controller set-point

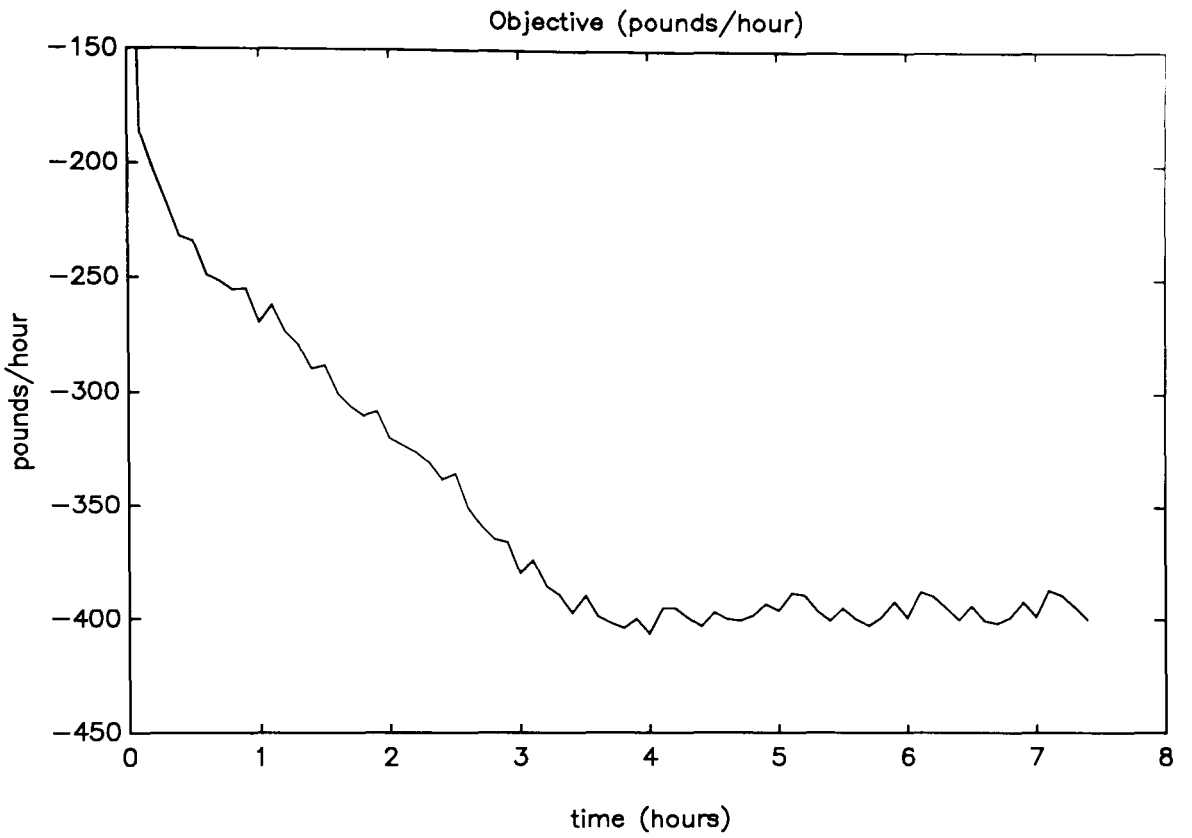

Figure 10.6.2.6: Example 10.6.2.a, evolution of the steady-state objective function

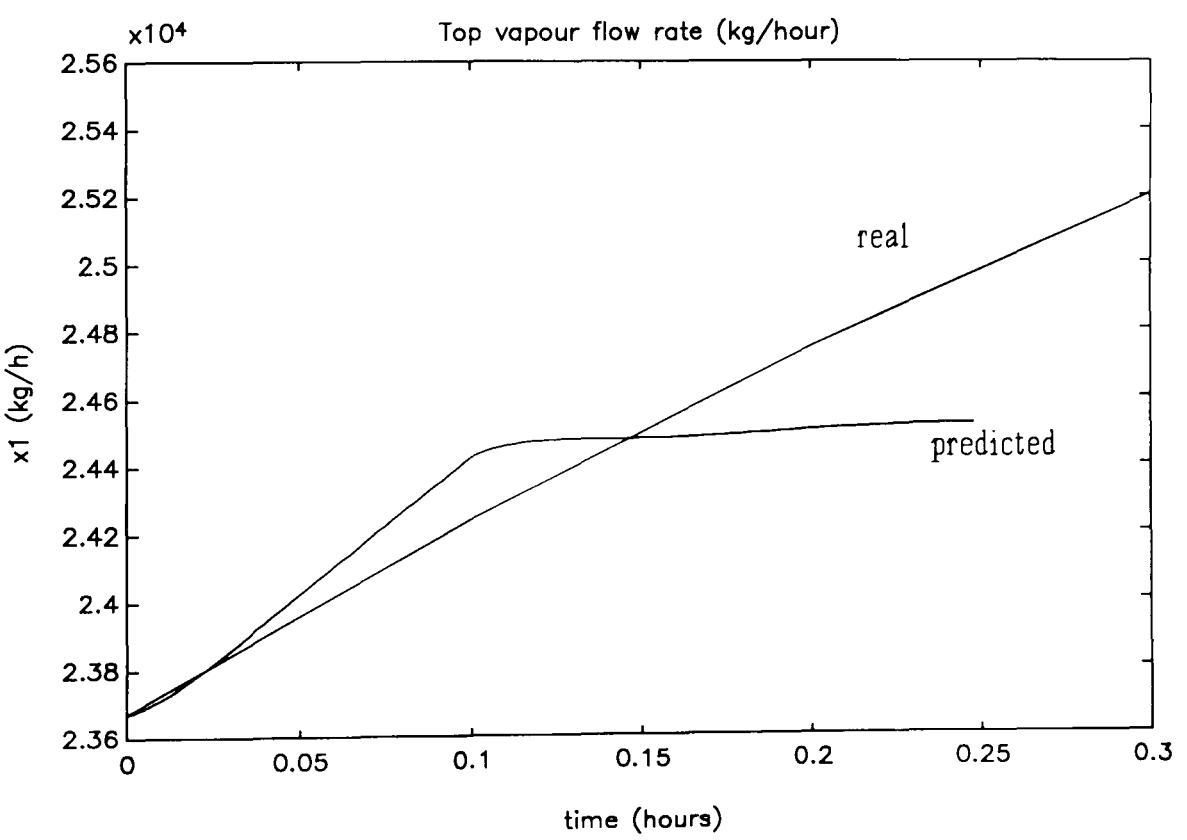

Figure 10.6.2.7: Example 10.6.2.a, Top vapour flow rate predicted and actual trajectories

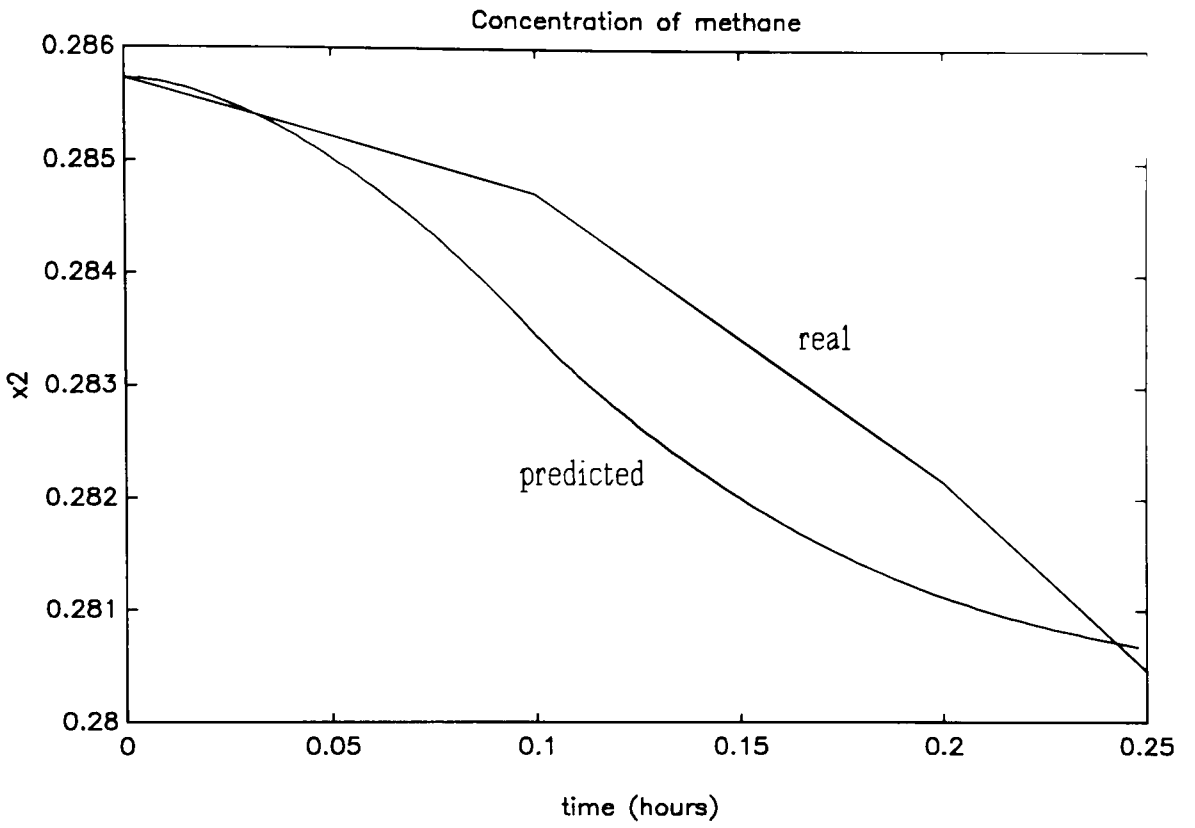

Figure 10.6.2.8: Example 10.6.2.a, concentration of methane predicted and actual trajectories

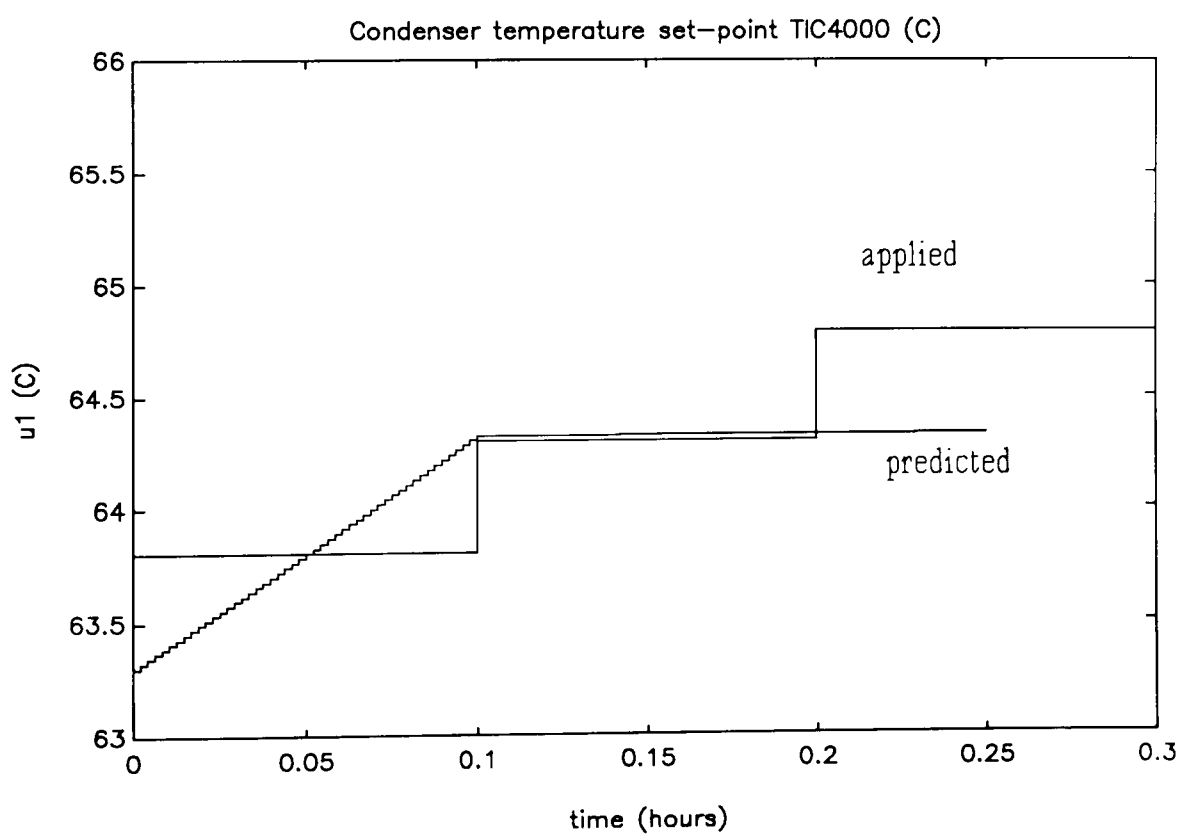

Figure 10.6.2.9: Example 10.6.2.a, predicted and applied condenser temperature controller set-point

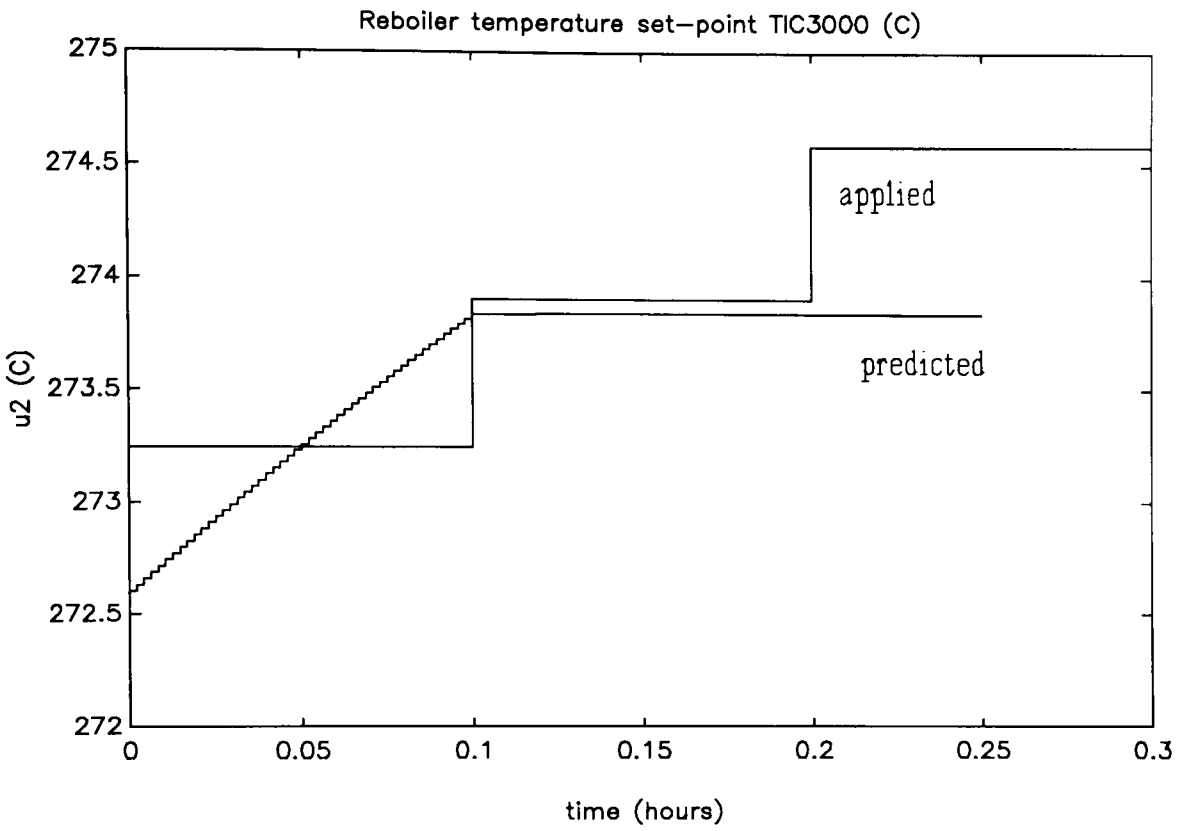

Figure 10.6.2.10: Example 10.6.2.a, predicted and applied reboiler temperature controller set-point

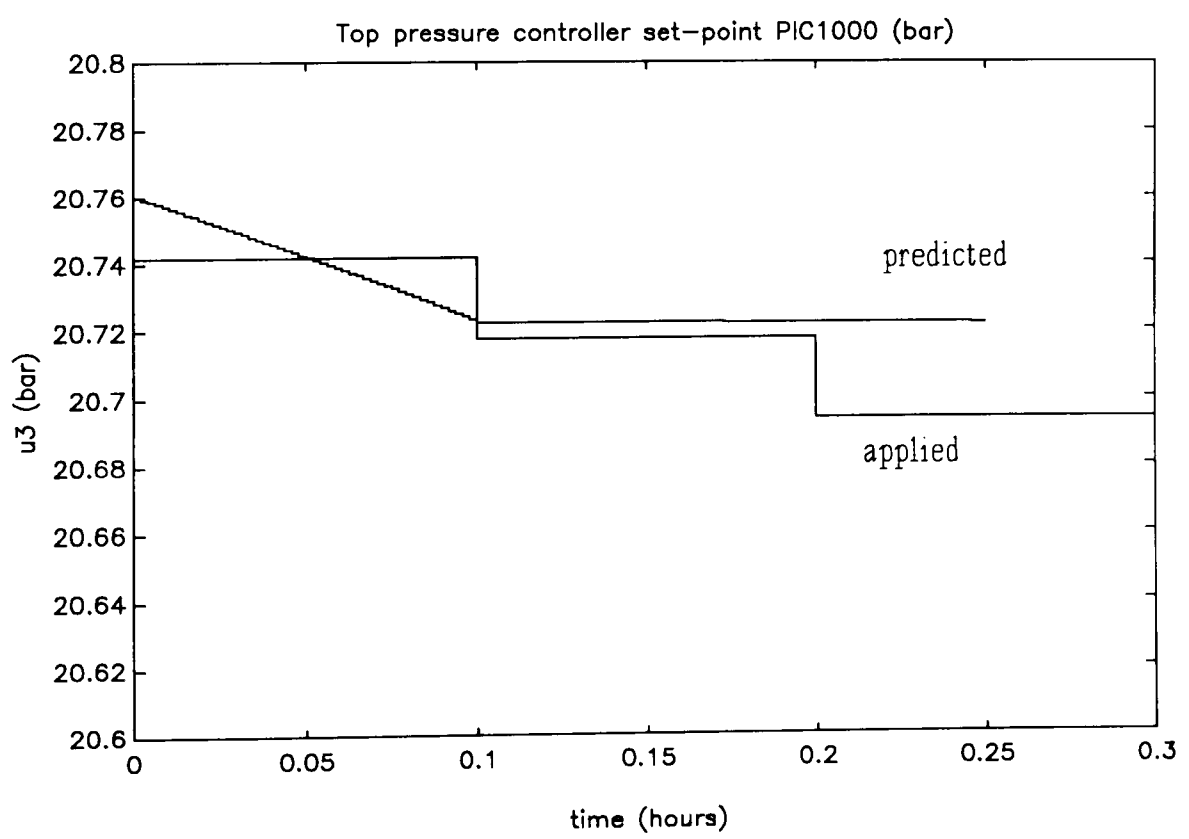

Figure 10.6.2.11: Example 10.6.2.a, predicted and applied top pressure controller set-point

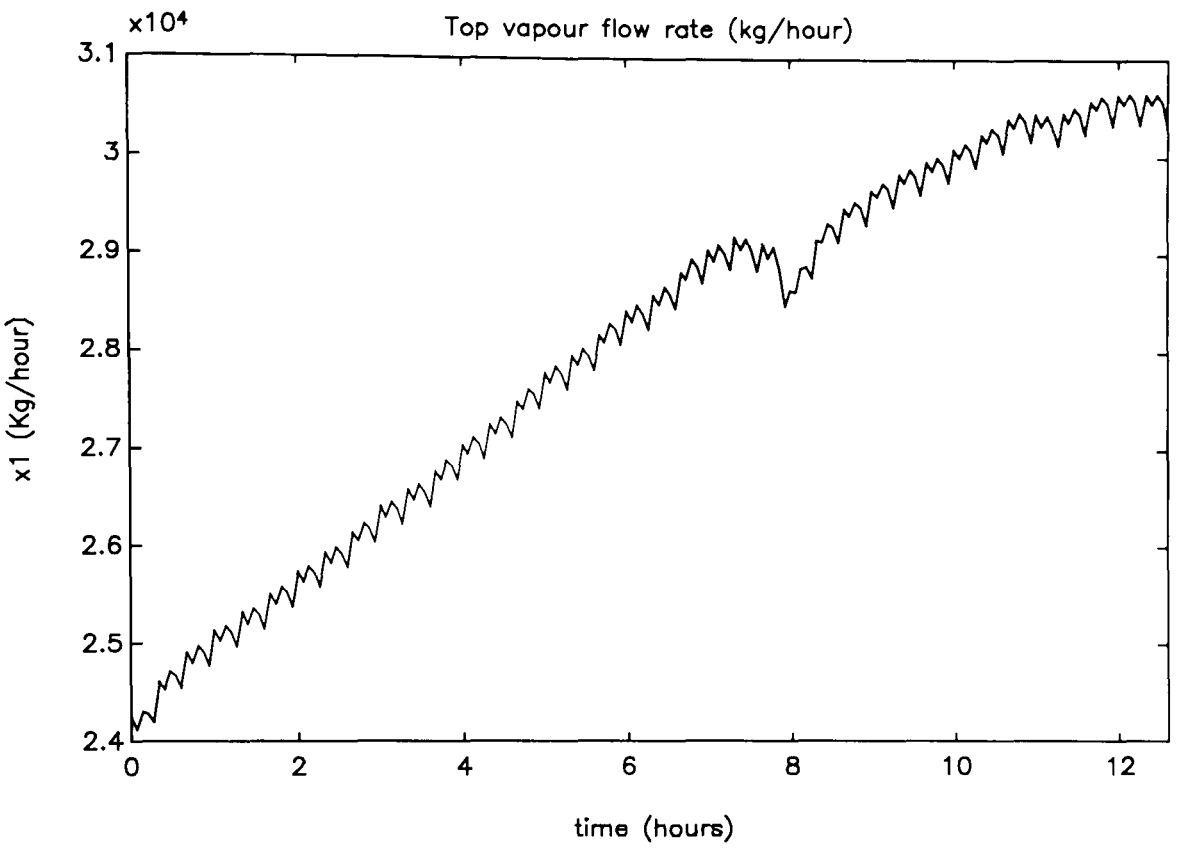

Figure 10.6.2.12: Example 10.6.2.b, top vapour flow rate trajectory

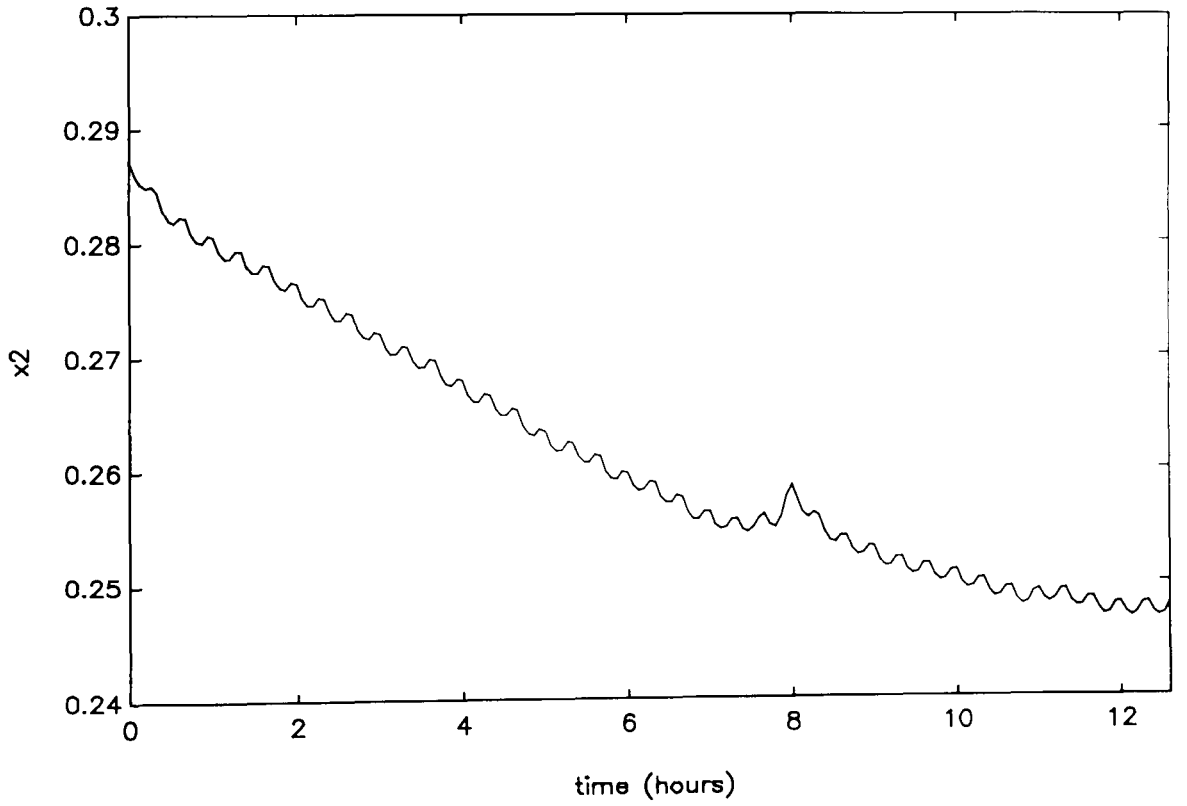

Figure 10.6.2.13: Example 10.6.2.b, concentration of methane trajectory

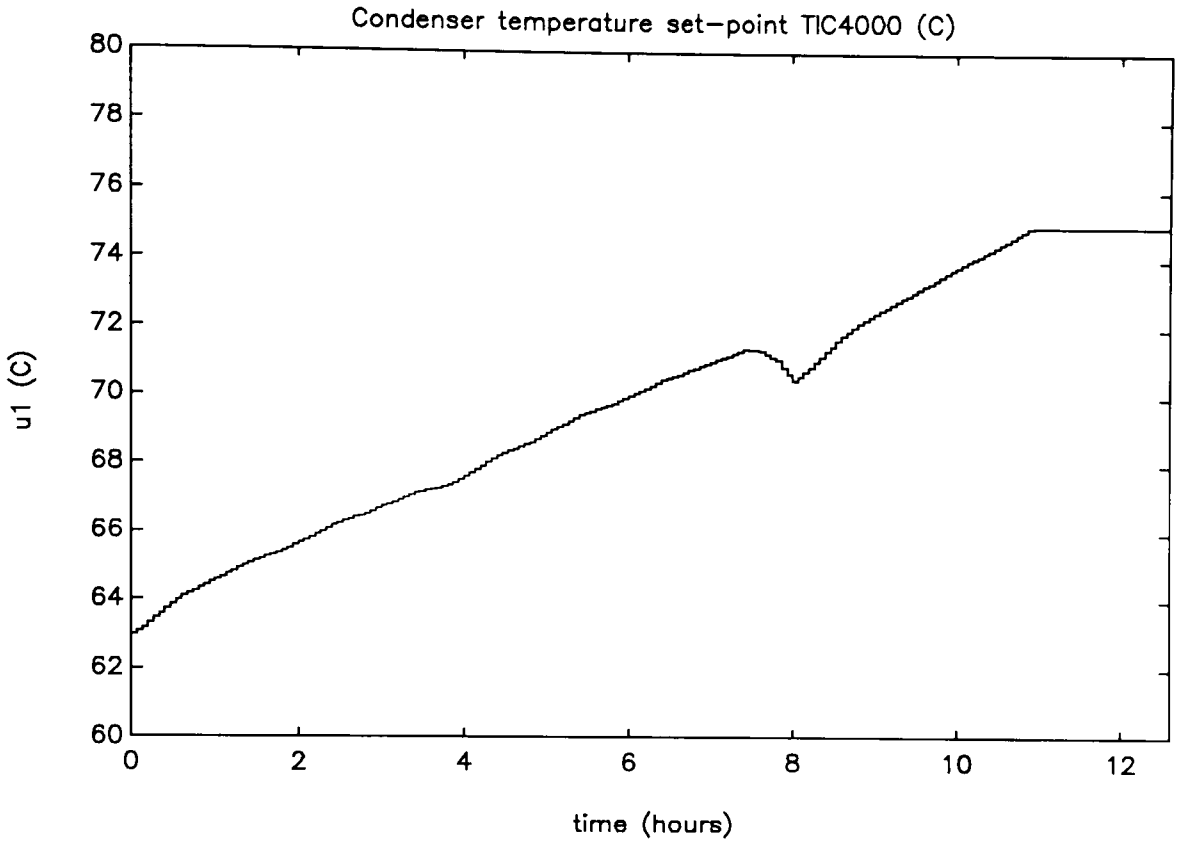

Figure 10.6.2.14: Example 10.6.2.b, condenser temperature controller set-point

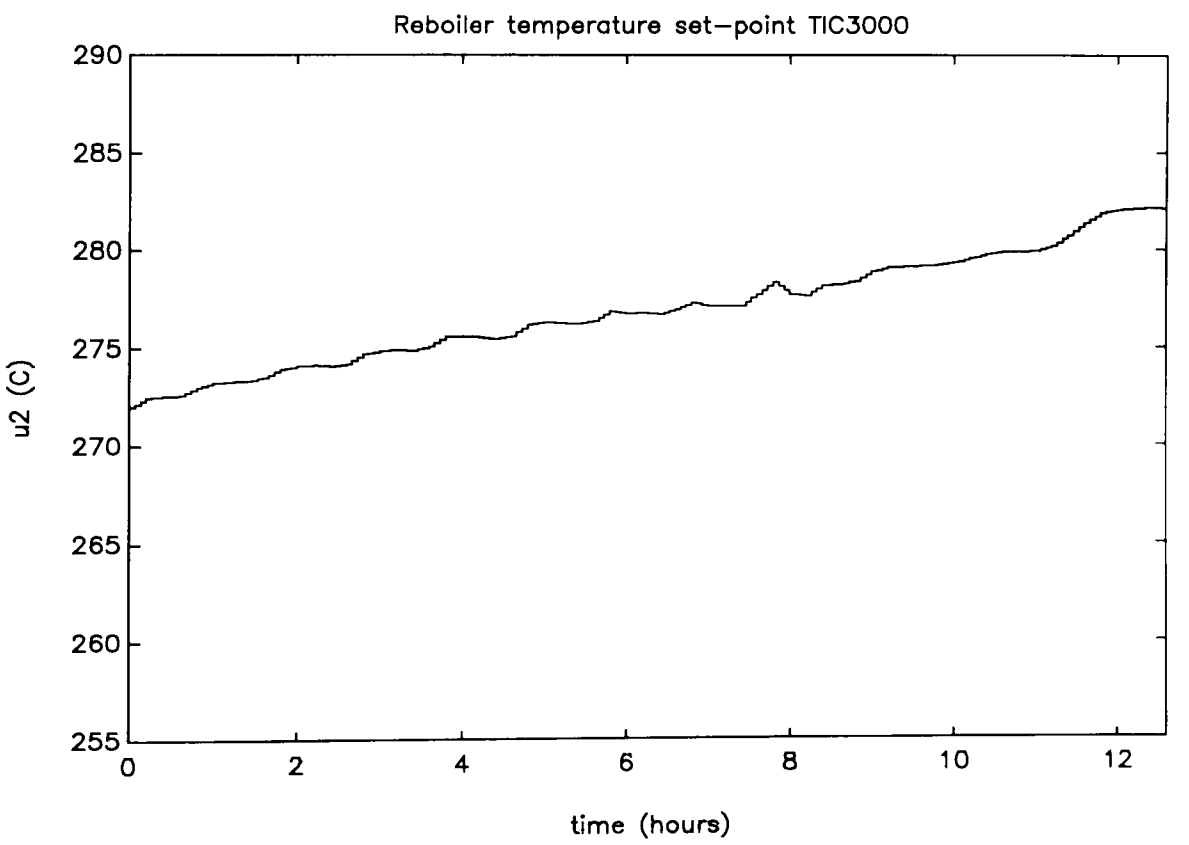

Figure 10.6.2.15: Example 10.6.2.b, reboiler temperature controller set-point

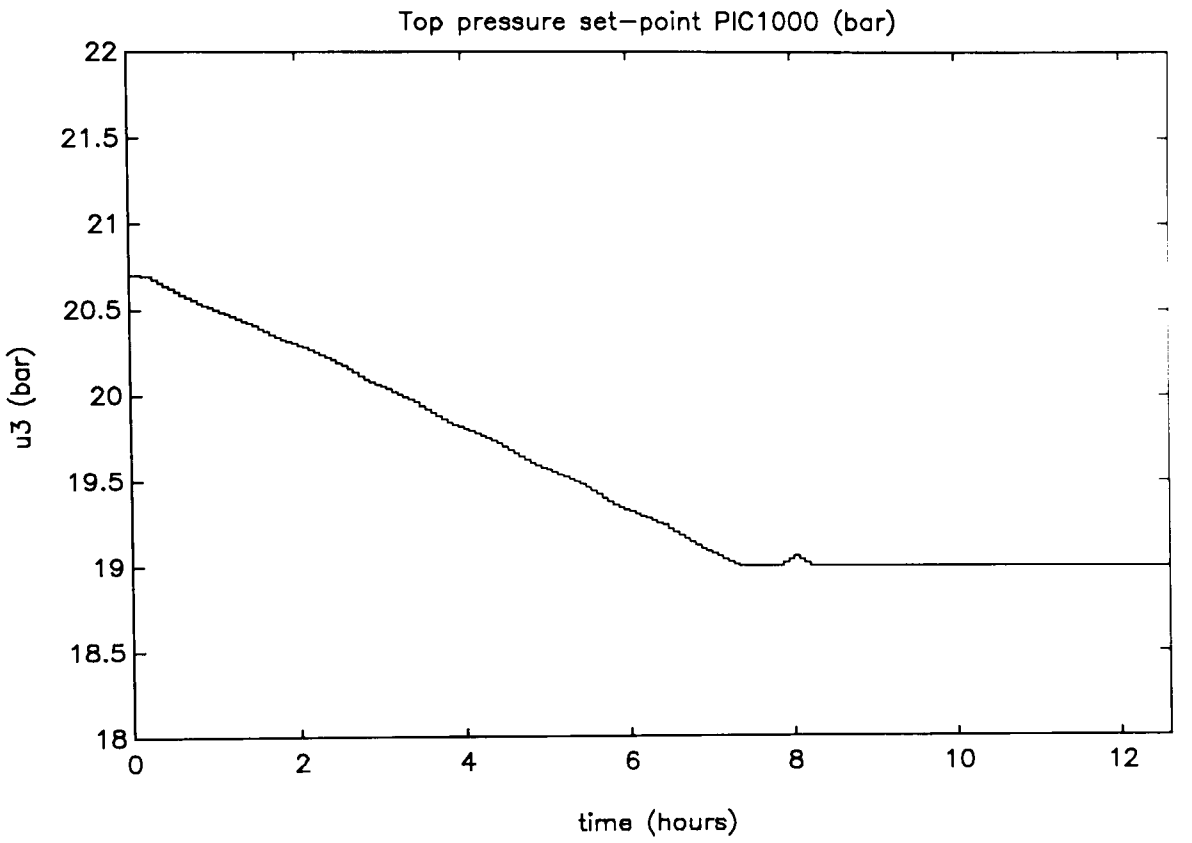

Figure 10.6.2.16: Example 10.6.2.b, top pressure controller set-point

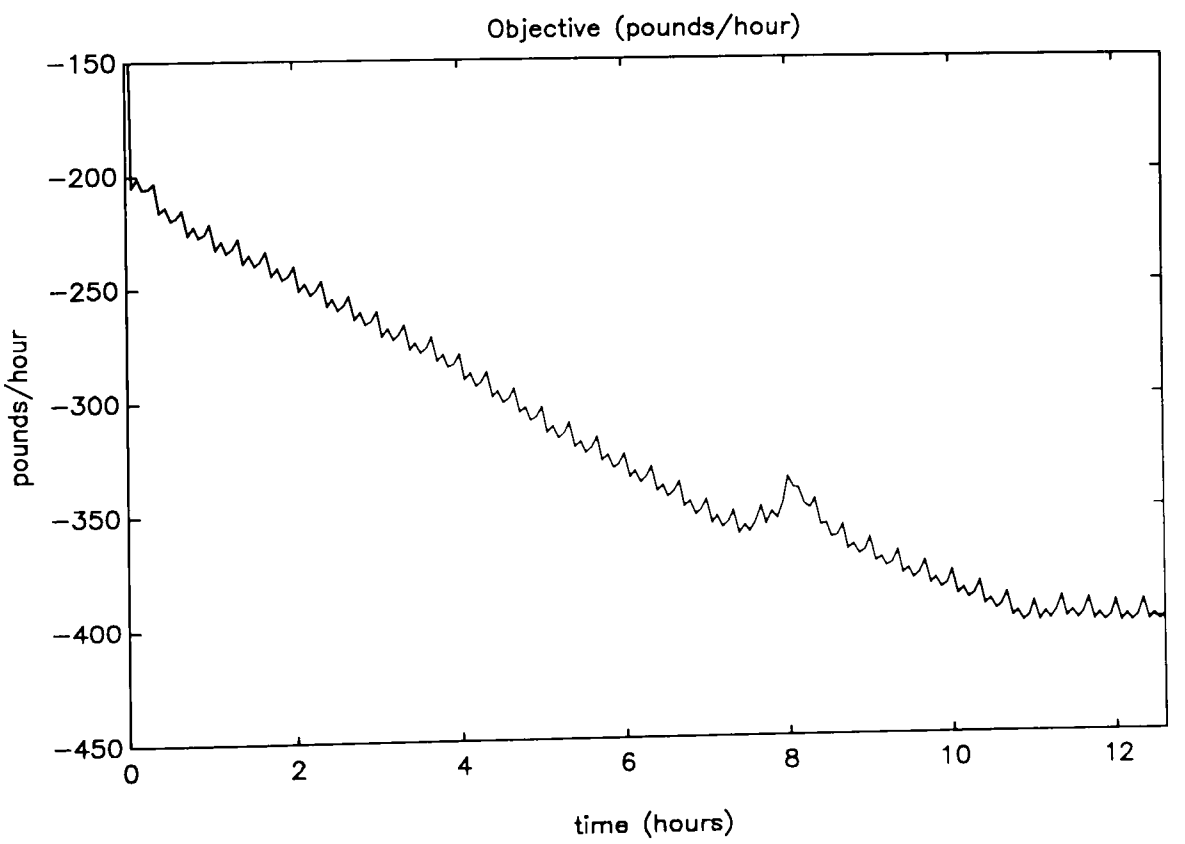

Figure 10.6.2.17: Example 10.6.2.b, evolution of the steady-state objective function

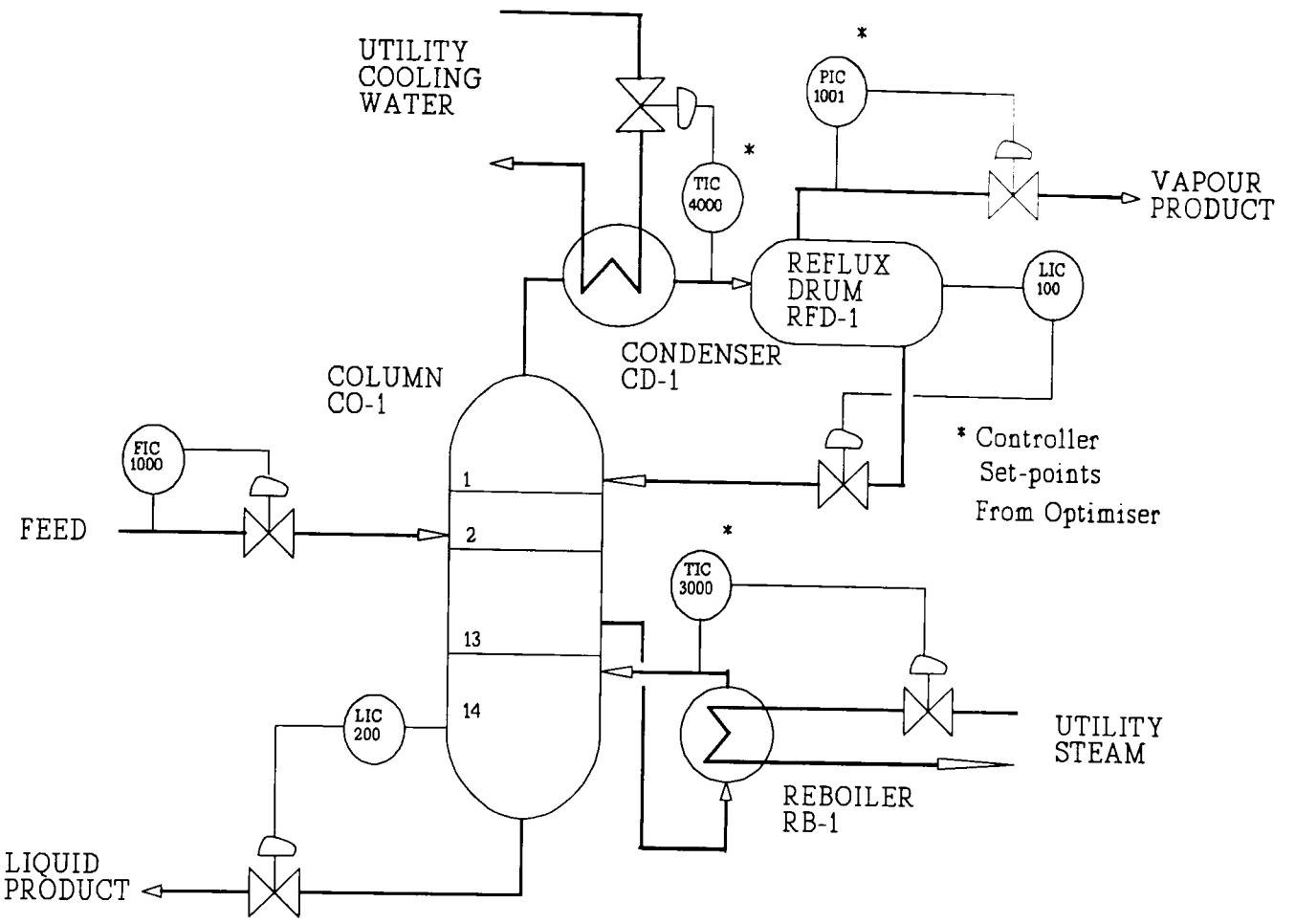

Figure 10.6.2.18: Example 10.6.2, schematic diagram of the process

Example 10.6.3: Optimisation of two CSTR in series

This example consists of the same dynamic system of Example 9.5.4. Here two CSTR's in series in which an exothermic autocatalytic reaction is taking place, interact in both directions due to the recycle of a 50% fraction of the product stream into the first reactor. Regulatory controllers are used to regulate the temperature at both reactors. The dynamics associated with these controllers are neglected. The steady-state performance index reflects economic objectives associated with the achievement of maximum production of substance B. The dynamic equations describing the process, the initial conditions of the system and the nomenclature used are described in Example 9.5.4. In order to drive the process to its steady-state optimum we will use the optimisers developed in this chapter: (a) DSSO, and (b) LP-DISOPE. The problem has been scaled for its solution. The results presented here may be compared with those obtained in Example 9.5.4. In case (b), the control applied to the plant when the set-points were updated consisted of the average of the first N_e predicted samples.

The steady-state objective function is given by:

$$N^*(x,u) = -x_4 \qquad \textbf{(10,83)}$$

where $x_4 = Cb_2$ is the concentration of B in the second reactor. In this example, the matrices $\hat{A}$, $\hat{B}$ were computed by finite differences. No identifier was used. The sampling period for computing these matrices was $T_s = 1$ min. Table 10.6.3.1 shows the tuning parameters of DSSO. Table 10.6.3.2 shows the tuning parameters of LP-DISOPE. Table 10.6.3.1 shows the steady-state optimum achieved in each case. Figure 10.6.3.1 shows the trajectory of concentration Cb_2 for case (a). Figure 10.6.3.2 shows the control signals for case a. Figure 10.6.3.3 shows the trajectory of concentration Cb_2 for case b. Figure 10.6.3.4 shows the control signals for case b.

Case	R	$\Delta u_{\max}(1)$ °C	$\Delta u_{\max}(2)$ °C	Update period $N_e T_s$ (min)
a	$0.4 I_2$	4	4	5

Table 10.6.3.1: DSSO tuning parameters

Case	Φ	Q	R	Control horizon $N_c T_s$	Prediction horizon NT_s	Update period $N_e T_s$ (min)
b	I_4	I_4	$0.8 I_2$	5	25	5

Table 10.6.3.2: LP-DISOPE tuning parameters

Case	Cb_2	T_1 (°C)	T_2 (°C)
a	0.072525	312	309.6562
b	0.072525	312	309.6530

Table 10.6.3.3: Steady-state results obtained

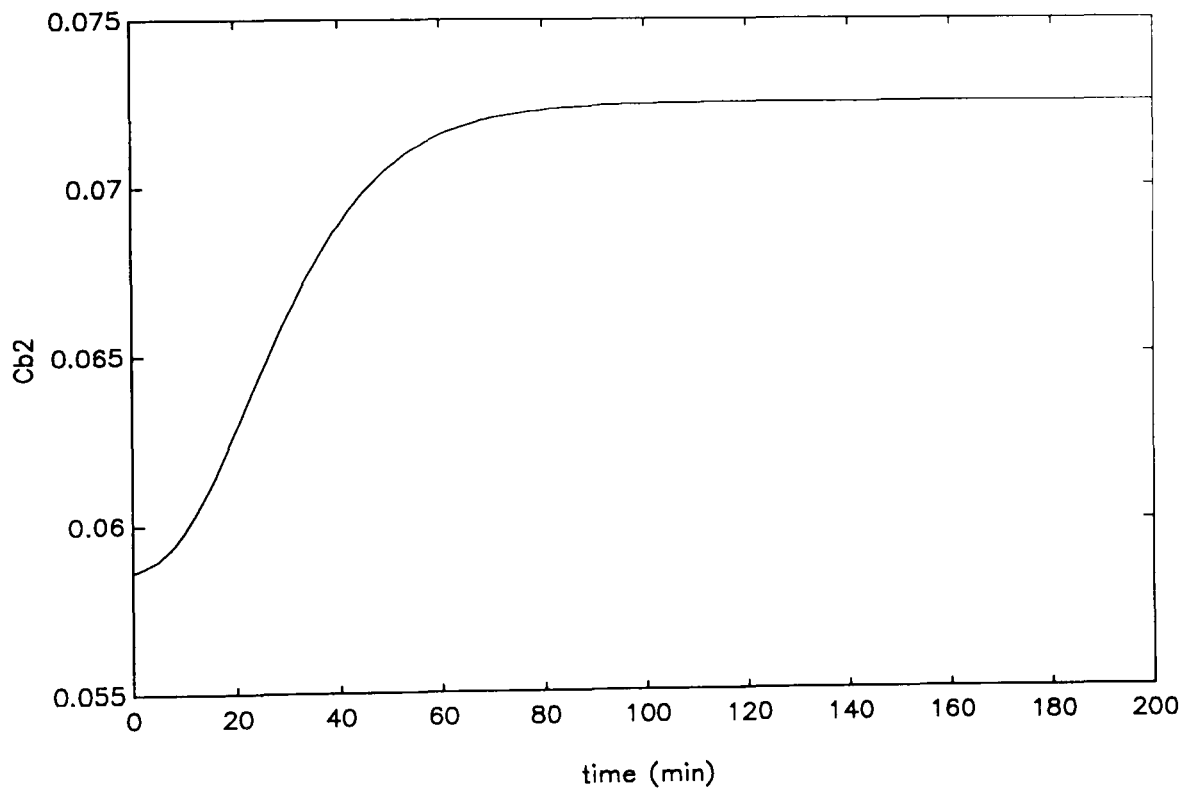

Figure 10.6.3.1: Example 10.6.3a, trajectory of concentration of B in the second reactor

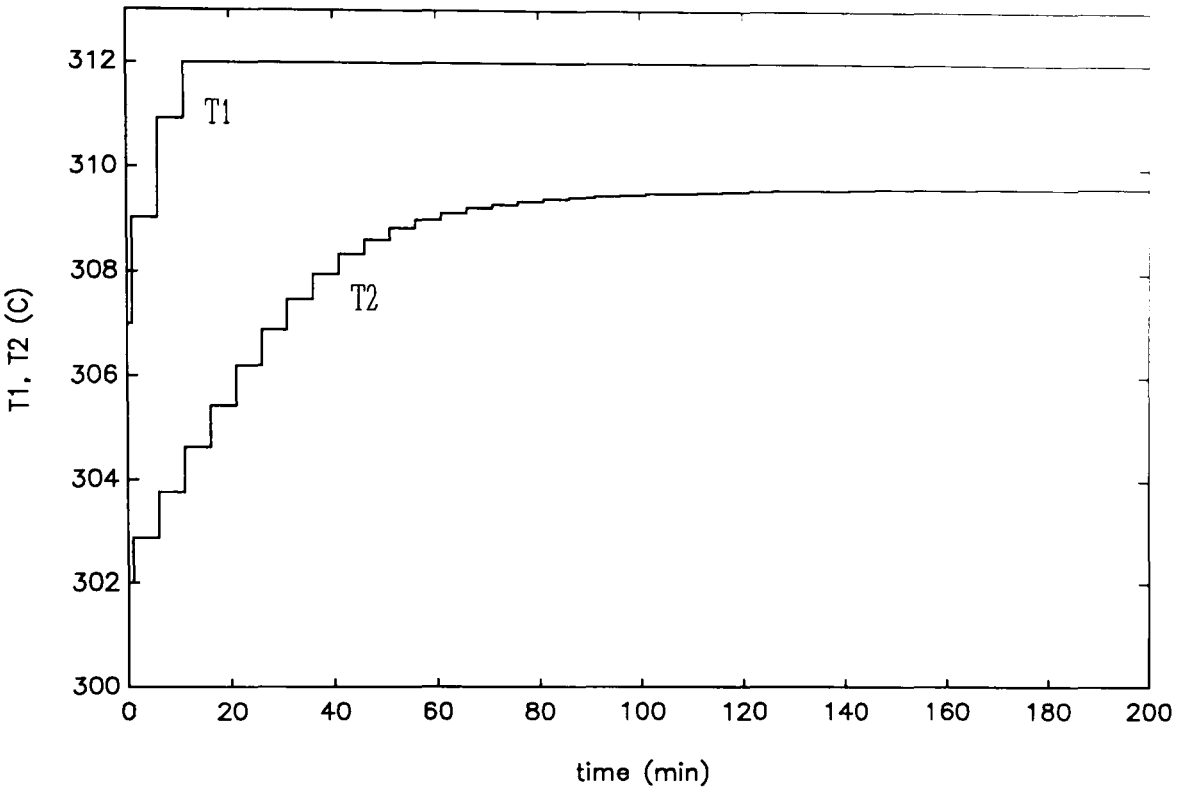

Figure 10.6.3.2: Example 10.6.3a, control signals, temperatures in reactors 1 and 2

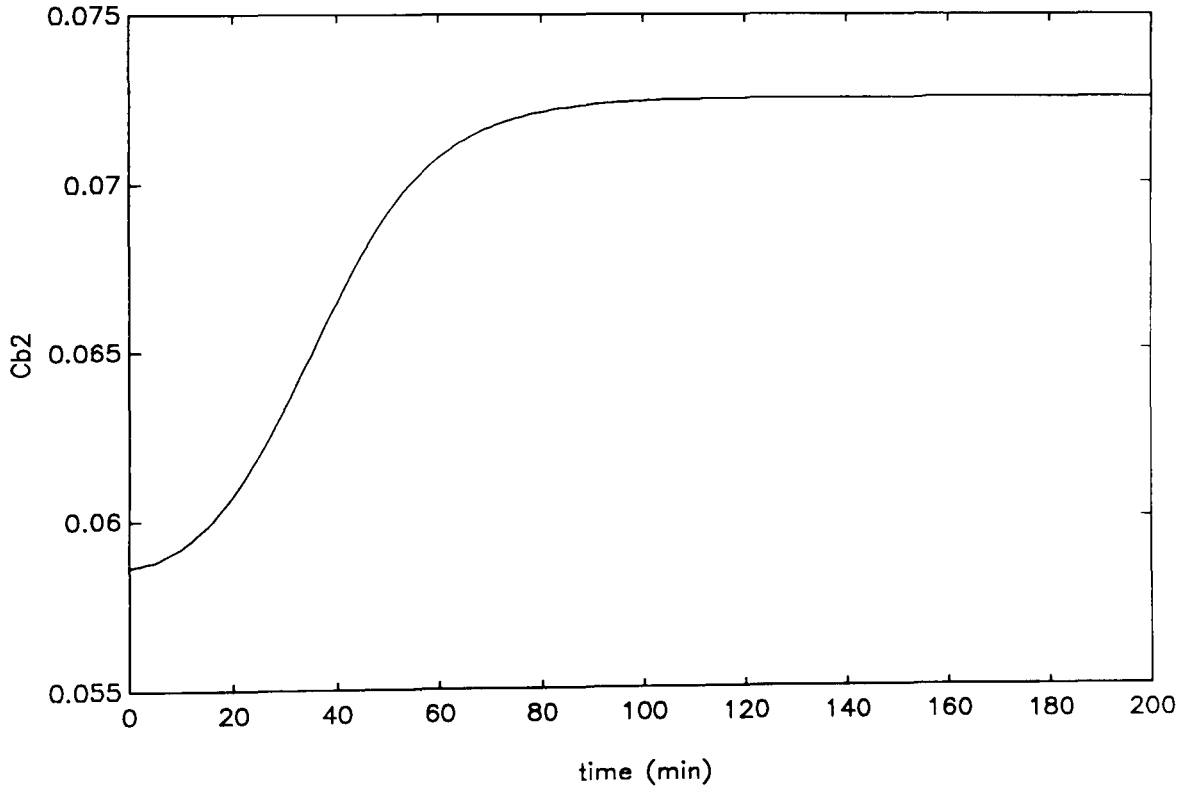

Figure 10.6.3.3: Example 10.6.3b, trajectory of concentration of B in the second reactor

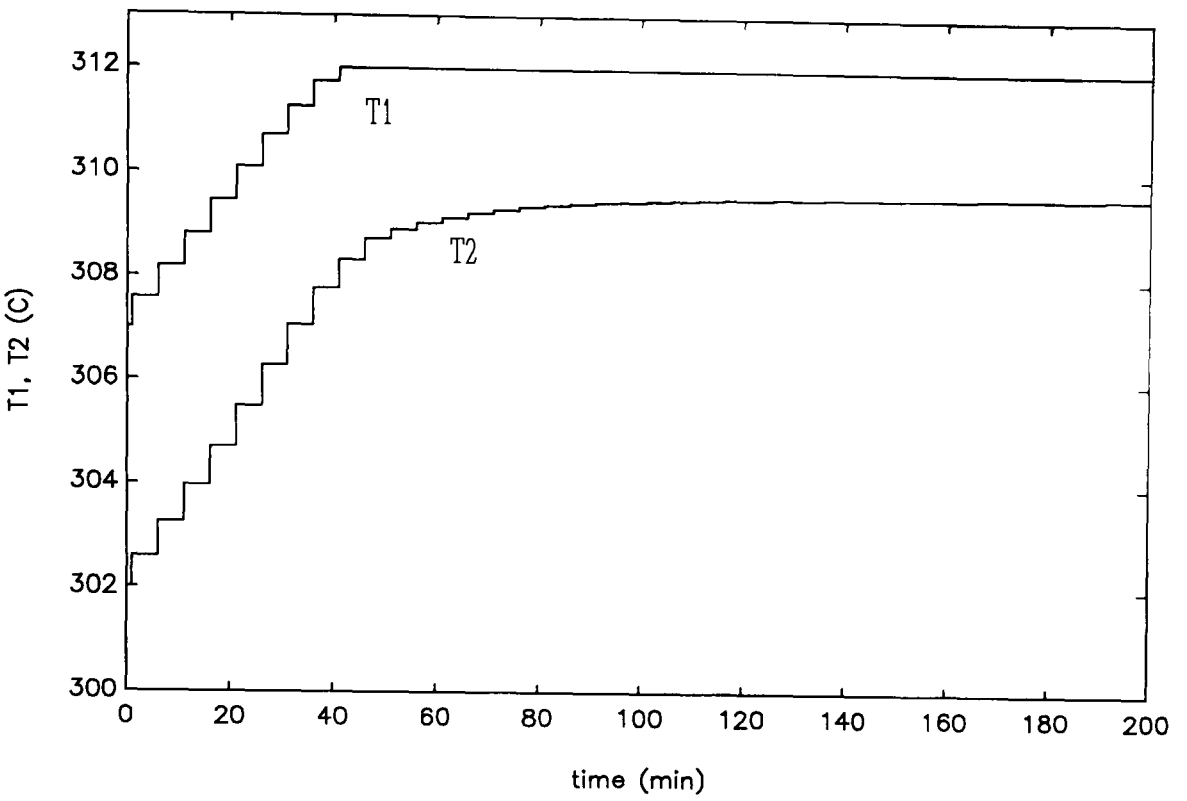

Figure 10.6.3.4: Example 10.6.3b, control signals, temperatures in reactors 1 and 2

10.7 REMARKS

It is important to point out that both optimising controllers (LP-DISOPE and DSSO) depend for their control action on the values of the identified system matrices. Therefore, the tuning of the identifier should be given careful consideration. In order to enhance identifiability, some suitable noise may be added to the set-points. In the case of a moving data window identifier, a sufficiently long data length and an appropriate update period should be specified, depending on the expected speed of change of the control variables. An appropriate forgetting factor should be specified in the case of a recursive identifier.

Notice that because of the short sampling period of the identifier used in the the distillation column case, some eigenvalues of $\hat{A}$ are close to the point (1,0) on the z-plane. This causes the matrix $(I_n - \hat{A}^T)$ (which inverse is used in the calculations of DSSO, see Section 10.4) to be near singular at one point along the state trajectory. This produces the sudden change of direction of the set-points noticeable in Figures 10.6.2.14 and 10.6.2.15. For this reason, DSSO had to be tuned for a slow response in order to avoid too drastic changes in the set-points in the face

of the above mentioned event. This was not a problem in the LP-DISOPE case, where state weighting was used. The use of state weighting with a sufficiently long prediction horizon allows stable predictions even when the system has local eigenvalues on or outside the unit circle. Therefore, state weighting has a direct stabilizing effect which should be further investigated (see Bitmead *et al*, 1990). On the other hand, notice that state weighting is not possible in DSSO.

The value of the control weighting matrix has an obvious effect on the speed of response of the controller (This effect is not shown in the simulations presented here). The set-point update period also affects the speed of response of the controller (if all the other parameters are fixed, the longer the set-point update period, the slower the speed).

Provided the assumptions stated hold, LP-DISOPE and DSSO are able to drive the plant to its steady state optimum in a continuous way, so avoiding the need to wait for the system to achieve steady-state after every set-point change. The optimum achieved is the real plant optimum as far as the assumptions stated hold and the relevant states chosen represent the dynamics of the plant in an accurate way (It is well known that the order of many real-world plants is in practical terms infinite and any assumptions regarding the order of the process are often convenient and unavoidable).

The predictive nature of LP-DISOPE gives it the ability to anticipate the saturation of constraints in the future and to take appropriate control action in the face of such events. The controller is able to handle control magnitude, control rate of change and state dependent constraints. The predictions may be displayed to plant operators for different purposes. The loop may be closed by the operator.

The calculations LP-DISOPE needs to perform every time the set-points are updated are iterative LQ methods. Its computational load depends on the prediction horizon, on the dimensions of the system, and on the activation of inequality constraints. On the other hand, the calculations DSSO needs to perform every time the set-points are updated simple and non-iterative. Thus its computational load is modest.

Both optimising controllers should be able to track in a rapid way changes in the optimal operating conditions due to disturbances or price changes.

Both LP-DISOPE and DSSO have been successfully applied to a simulated multicomponent distillation column at SAST Ltd. and the optimum of the plant has

been achieved in the simulations. Additional tests have been performed with low order examples with known solution and the correct optimum has also been achieved.

10.8 SUMMARY

In this chapter, two optimising controllers have been designed which are able to lead a process from a suboptimal operational condition to its steady-state optimum in a continuous way. The controllers use derivative and state information from the plant via a shadow model and an adaptive state-space linear model identifier. The new algorithms are developed from the basis that a nonlinear model of the process is not available for predictions and, hence, DSSO does not require predictions and LP-DISOPE uses predictions based on a linearized model of the process. The optimality of the procedures has been analyzed. Both techniques have been tested with simulation examples, including realistic industrial-scale simulations of the dynamic optimisation of a multicomponent distillation column using a rigorous process simulator.

CHAPTER 11

COMPARISONS OF DISOPE WITH OTHER ALGORITHMS

In this chapter the DISOPE algorithm is placed within the general classification of nonlinear optimal control approaches presented in Section 1.4. Furthermore, similarities and differences between a DISOPE algorithm and a well established approach such as quasilinearization are drawn. Moreover, comparisons with previous work by Hassan and Singh (1976) and Mahmoud *et al* (1980) are made, taking into account some similarities between the DISOPE technique and these methods, but also remarking the differences.

11.1 DISOPE AS AN OPTIMAL CONTROL ALGORITHM AND COMPARISONS WITH OTHER TECHNIQUES

DISOPE may be broadly classified as a function space algorithm, since the necessary optimality conditions obtained from the minimum principle (or a variational approach) are enforced iteratively using costate variables and the gradients of the Hamiltonian in the iterations. It is important to point out that in this thesis, following the objectives stated in Chapter 1, the DISOPE technique has been extended to handle practical problems taking into account the needs of industry. Therefore, extensions for handling of control magnitude constraints, state dependent constraints, as well as discrete time, hierarchical and set-point tracking versions have been developed. Its applications on the fields of nonlinear predictive control, steady state process optimisation based on dynamic information, and on the optimisation of batch processes have been studied.

It is not the purpose here to draw comparisons of DISOPE with every nonlinear optimal control technique proposed in the literature. Rather, we will concentrate on some techniques which, for several reasons and to different degrees, have similarities with the DISOPE approach. The techniques chosen for comparison

purposes are quasilinearization (see, for instance, Sage and White (1977)), the two-level approach to the dynamic optimisation of nonlinear systems by Hassan and Singh (1976) and the hierarchical approach to the joint problem of systems identification and dynamic optimisation by Mahmoud *et al* (1980). Comparisons regarding the performance of the techniques lie beyond the scope of this thesis.

11.1.1 Comparisons between DISOPE and the quasilinearization approach.

For comparison purposes we will use the following discrete time optimal control problem:

$$\min_{\substack{u(k) \\ k\in[N_0,N_f-1]}} J^* = \sum_{k=N_0}^{N_f-1} L^*(x(k),u(k),k)$$

subject to

$$x(k+1) = f^*(x(k),u(k),k)$$
$$x(N_0) = x_0$$

where $x\in\Re^n$ and $u\in\Re^m$ are state and control variables, respectively, $L^* : \Re^n\times\Re^m\times\Re\rightarrow\Re$ is a discrete performance measure function, $f^* : \Re^n\times\Re^m\times\Re\rightarrow\Re^n$ is a set of discrete-time state equations.

Application of the stationarity condition (see Chapter 5) gives the following expression for the control variable:

$$\hat{u}(k) = h(\hat{x}(k),\hat{p}(k),k) \qquad \textbf{(11,1)}$$

where the costate $p\in\Re^n$ and the state variables and are given by the solution of the following nonlinear two point boundary value problem (TPBVP):

$$x(k+1) = f^*(x(k)\ ,\ h(x(k),p(k))\ ,\ k),\ x(N_0)=x_o \qquad \textbf{(11,2)}$$
$$p(k+1) = g(x(k),p(k),k),\ p(N_f) = 0$$

By using a quasilinearization approach, a nominal solution vector is chosen $x^0(k)$, $p^0(k)$ so as to satisfy as many of the boundary conditions as possible. The

state and costate difference equations are linearized about the nominal trajectories and a succession of non-homogeneous, linear TPBVP's is solved until convergence is achieved. Define $X^i(k) = [\, x^i(k)^\tau,\, p^i(k)^\tau \,]^\tau$ as the solution vector for the i-th iteration and $F(X,k) = [\, f^*(X,k)^\tau \,,\, g(X,k)^\tau \,]^\tau$. The linear TPBVP to be solved at every iteration is as follows (Sage and White, 1977):

$$X^{i+1}(k+1) = A(k)^i X^{i+1}(k) + U^i(k) \qquad \textbf{(11,3)}$$

where:

$$\begin{aligned} A^i(k) &= \frac{\partial F(X^i(k),k)}{\partial X^i(k)} \\ U^i(k) &= F(X^i(k),k) - A(k)^i X^i(k) \\ x(N_o) &= x_0 \\ p(N_f) &= 0 \end{aligned} \qquad \textbf{(11,4)}$$

This TPBVP is linear with varying coefficients. It may be solved, for instance, by the sweep method. We may draw the following comparisons with the DISOPE approach with linear-quadratic model-based problem (Algorithm 5.3.1):

* The iterations in both quasilinearization and Algorithm 5.3.3 are based on the solution of a linear TPBVP and the successive solutions converge to the solution of a nonlinear optimal control problem.
* The idea of linearization has been used in Algorithm 5.3.3 to define the (constant) model-based dynamic matrices, however, such linearization is performed about a particular point in the state space and not about a trajectory. Then the matrix coefficients of the TPBVP solved by Procedure 5.3.1 are constant as opposed to those in the quasilinearization approach, which are time-varying.

11.1.2 Comparisons with previous work by Hassan and Singh (1976)

Algorithm 2.3.1 has similarities with the two-level method for optimisation of nonlinear dynamic systems proposed by Hassan and Singh (1976). The method

is described below in its centralized version and using an appropriate notation to facilitate comparisons with the DISOPE approach. In the work by Hassan and Singh (1976) augmentation terms are added to the performance index with the purpose of improving convergence, as is done in the DISOPE approach, but these are not included in the description below for the sake of simplicity.

It is desired to solve the following nonlinear optimal control problem with a quadratic performance index:

$$\min_{u(t)} \quad J^* = \int_{t_0}^{t_f} \frac{1}{2}\left(x(t)^\top Q x(t) + u(t)^\top R u(t) \right) dt$$

subject to

$$\dot{x} = f^*(x(t),\ u(t),\ t)$$
$$x(t_0) = x_0$$

where $x \in \Re^n$ and $u \in \Re^m$ are state and control variables, respectively, Q and R are weighting matrices of the appropriate dimensions.

Expanding by Taylor series the dynamic equation about the (assumed) equilibrium point $x = 0,\ u = 0$ and keeping only first order terms, the following is obtained:

$$\dot{x} = A x(t) + B u(t) + \alpha(x,u,t) \qquad \textbf{(11,5)}$$

where

$$\alpha(x,u,t) = f^*(x,u,t) - A x(t) - B u(t) \qquad \textbf{(11,6)}$$

Suppose now that an upper-level provides state and control predictions $z(t),\ v(t)$ which are used in (11,6) to obtain $\alpha(z,v,t)$ which is then fixed. The following condition additionally is required for an optimal solution:

$$u(t) = v(t)$$
$$z(t) = x(t) \qquad \textbf{(11,7)}$$

A Hamiltonian is then formed as follows:

$$H = \frac{1}{2}(x(t)^{\top} Q x(t) + u(t)^{\top} R u(t)) + p(t)^{\top}(Ax(t) + Bu(t) + \alpha(z, v, t)) + \beta^{\top}(t)(\, z(t) - x(t)\,) + \lambda(t)^{\top}(\, v(t) - u(t)\,) \qquad \textbf{(11,8)}$$

where $\lambda(t) \in \Re^m$, $\beta(t) \in \Re^n$ are Lagrange multipliers and $p(t) \in \Re^n$ is the costate vector. Now, using a variational approach, the following necessary optimality conditions are obtained, in addition to (11,7):

$$u(t) = -R^{-1}(\, B^{\top} p(t) - \lambda(t)\,) \qquad \textbf{(11,9)}$$

$$\dot{p} = -Qx(t) + A^{\top} p(t) - \beta(t)\ ,\ p(t_f) = 0 \qquad \textbf{(11,10)}$$

$$\dot{x} = Ax(t) + BR^{-1}(-B^{\top} p(t) - \lambda(t)\,) + \alpha(z, v, t)\ ,\ x(t_0) = x_0 \qquad \textbf{(11,11)}$$

$$\lambda(t) = (\, B - \frac{\partial f^*(z, v, t)}{\partial v}\,)^{\top} p(t)$$
$$\beta(t) = (\, A - \frac{\partial f^*(z, v, t)}{\partial z}\,)^{\top} p(t) \qquad \textbf{(11,12)}$$

Finally, based on the above analysis, an algorithm is formulated as follows:

Algorithm 11.1.2: Two-level method for optimisation of nonlinear dynamic systems by Hassan and Singh (1976)

Step 0 At the upper level, select an initial guess $v^0(t)$, $z^0(t)$, $p^0(t)$, $\lambda^0(t)$ and $\beta^0(t)$, $t \in [t_0, t_f]$. Set i=0.

Step 1 At the upper level, compute $\alpha(z, v, t)$ to satisfy (11,6).

Step 2 At the lower level, with specified $v(t)$, $z(t)$, $\alpha(z, v, t)$, $\lambda(t)$ and $\beta(t)$ solve the linear two point boundary value problem defined by (11,10) and (11,11) to find $x(t)$ and $p(t)$. Also find the control variable $u(t)$ from (11,9).

Step 3 At the upper level, if $u(t) = v(t)$ stop the iterations, else set $v(t) = u(t)$, $z(t) = x(t)$ and $i = i+1$.

Step 4 At the upper level, compute the multipliers $\lambda(t)$ and $\beta(t)$ from (11,12) and repeat from step 1.

The following comparisons between Algorithm 11.1.2 and the continuous time DISOPE algorithm (see Chapter 2, Algorithms 2.2.1 and 2.3.3) may be drawn:

* For the particular case in which the real and model-based performance indexes are identical in Algorithm 2.3.3 and, furthermore, assuming that the relaxation gains are all set to one and the convexification factors are set to zero, and provided that the initial guess for the multipliers in Algorithm 11.1.2 $\lambda(t)$ and $\beta(t)$ are computed from (11,12) given $v^0(t)$, $z^0(t)$, and $p^0(t)$, then Algorithms 11.1.2 and 2.3.3 are equivalent.
* DISOPE was originally developed as a dynamic extension of ISOPE, with a philosophy directed towards the handling of model-reality differences in dynamic optimisation. The general continuous time DISOPE algorithm (Algorithm 2.2.1) reflects these aims, and Algorithm 2.3.3 is a particular case

of Algorithm 2.2.1. Thus, even though it may be said that Algorithms 11.1.2 and 2.3.3 are similar, their origins and philosophy are different. The ability of DISOPE for handling model-reality differences is used in Chapter 8, where its application has been proposed for the dynamic optimisation of batch processes.

* In the DISOPE approach, the real performance index may be general, the difference between the real and model-based performance index being properly accounted for. This feature is important since it has allowed practical extensions of the DISOPE approach such as the handling of control magnitude constrains (see Chapter 3) and state dependent constraints (see Chapter 7), the use of incremental control weighting to provide zero steady state off-set for constant set-points (see Chapter 6), and the application of DISOPE in problems where the performance index is not quadratic. A modification of the method by Hassan and Singh (1976) has been proposed by Singh and Titli (1978) which allows for non-quadratic performance indexes to be handled. The approach is similar to that used in Algorithm 2.3.3. Therefore, it may be said that Algorithm 2.3.3 and the modified algorithm by Singh and Titli (1978) (in its centralized version) are equivalent.

In the original work by Hassan and Singh (1976), a version of Algorithm 11.2.1 for the dynamic optimisation of large-scale systems is presented. The approach uses a similar Taylor expansion as in Equation (11,5), but the matrices A and B are the block diagonal parts of the corresponding jacobian matrices. The off-diagonal parts of the jacobian are included in the calculation of $\alpha(x,u,t)$ and also appear in the multipliers $\lambda(t)$ and $\beta(t)$. It is assumed that the performance index is quadratic and additively separable. Then, at the lower level, N independent linear-quadratic sub-problems are solved and, at the upper level, the updating of the coordination vector $[\ v\ z\ \lambda\ \beta\ ]^T$ is carried out. Following this description, the following comparisons can be made with the hierarchical DISOPE algorithms (Algorithm 4.2.1 and 4.3.2):

* The algorithms may be compared in similar terms as with Algorithms 2.2.1 and 2.3.1 with Algorithm 11.1.2 above. However, the way of handling the

interactions between subsystems is different, since Algorithm 4.3.2 uses an approach based on the interaction prediction method (Jamshidi, 1983), an additional multiplier $\Omega(t)$ is introduced to take account of the interconnections, and the interaction vector $w(t)$ is part of the coordination vector. In both approaches, the resultant sub-problems to be solved at the lower level are independent.

11.1.3 Comparisons with previous work by Mahmoud et al (1980)

In the research work presented by Mahmoud *et al* (1980) a hierarchical technique for solving the joint problem of parameter estimation and dynamic optimisation is proposed. The solution of the problem is done by using a four level hierarchical structure. The technique is structurally different from the DISOPE approach, but since the type of problem addressed is apparently similar to that dealt with by DISOPE, namely the integration of system dynamic optimisation and parameter estimation, it is worth discussing the differences between the approaches.

The dynamic optimisation problem is defined as follows:

$$\min_{u(t)} \quad J = \int_{t_0}^{t_f} \tfrac{1}{2}\left(x(t)^{\mathsf{T}} Q x(t) + u(t)^{\mathsf{T}} R u(t) \right) dt$$

subject to

$$\begin{aligned} \dot{x} &= f^*(x(t),\ u(t),\ \alpha(t)) \\ x(t_0) &= x_0 \end{aligned} \qquad \textbf{(11,13)}$$

given the (time-varying) parameter vector $\alpha(t) \in \Re^r$, where $x \in \Re^n$ and $u \in \Re^m$ are state and control variables, respectively, Q and R are weighting matrices of the appropriate dimensions. The parameter identification problem is defined as follows:

$$\min_{\alpha(t)} \quad V = \int_{t_0}^{t_f} \tfrac{1}{2}\left((y_0(t) - y)^{\mathsf{T}} W(t) (y_0(t) - y(t)) \right) dt$$

subject to

$$y(t) = D(x(t),\ u(t),\ \alpha(t)) \tag{11.14}$$

where $y \in \Re^{n_o}$ is the output vector, $y_0 \in \Re^{n_o}$ is the vector of observations, and $W(t)$ is a suitable weighting matrix, and $D : \Re^n \times \Re^m \times \Re^r \rightarrow \Re^{n_0}$ is an output mapping.

Now, a joint problem which combines both problems is formulated as follows: "Determine $\alpha(t)$ and $u(t)$ so as to minimize J and V subject to the equality constraints (11,13) and (11,14)". An appropriate combined form of the twin objective is formulated using the parametric approach:

$$\min_{u(t),\alpha(t)} \quad Z = \varepsilon V + (1-\varepsilon)J = (1-\varepsilon)\int_{t_0}^{t_f} \frac{1}{2}\left(x(t)^\top Q x(t) + u(t)^\top R u(t) \right) dt$$

$$+ \varepsilon \int_{t_0}^{t_f} \frac{1}{2}\left((y_0(t) - D(x,u,\alpha))^\top W(t) (y_0(t) - D(x,u,\alpha)) \right) dt$$

subject to (11,13), where $0 < \varepsilon < 1$. The following separation variables are introduced:

$$z(t) = x(t)$$
$$v(t) = u(t)$$
$$\sigma(t) = \alpha(t)$$

Next, a penalty term $\frac{1}{2}\rho \|\alpha(t) - \sigma(t)\|^2$ is introduced in the joint performance index so as to further ensure convergence of the parameter estimates. In addition, the model dynamics (11,13) are expanded by Taylor series about the predicted trajectories $z(t),\ v(t),\ \sigma(t)$.

Thus:

$$\min_{u(t),\alpha(t)} \quad Z = \varepsilon V + (1-\varepsilon)J = (1-\varepsilon)\int_{t_0}^{t_f} \frac{1}{2}\left(x(t)^{\tau} Q x(t) + u(t)^{\tau} R u(t) \right) dt$$

$$+ \varepsilon \int_{t_0}^{t_f} \frac{1}{2}\left((y_0(t) - D(x,u,\alpha))^{\tau} W(t) (y_0(t) - D(x,u,\alpha)) + \rho \|\alpha(t) - \sigma(t)\|^2 \right) dt$$

subject to

$$\dot{x} = A(z,v,\sigma)x(t) + B(z,v,\sigma)u(t) + S(z,v,\sigma)\alpha(t) + \psi(z,v,\sigma) \qquad \textbf{(11,16)}$$

where

$$A(z,v,\sigma) = \frac{\partial f(z,v,\sigma)}{\partial z}$$

$$B(z,v,\sigma) = \frac{\partial f(z,v,\sigma)}{\partial v} \qquad \textbf{(11,17)}$$

$$S(z,v,\sigma) = \frac{\partial f(z,v,\sigma)}{\partial \sigma}$$

and

$$\psi(z,v,\sigma) = f(z,v,\sigma) - A(z,v,\sigma)x(t) - B(z,v,\sigma)u(t) - S(z,v,\sigma)\alpha(t) \qquad \textbf{(11,18)}$$

Then a Hamiltonian is formulated as follows:

$$\begin{aligned} H = {} & \frac{1}{2}(1-\varepsilon)\left(x(t)^{\tau} Q x(t) + u(t)^{\tau} R u(t) \right) \\ & + \frac{1}{2}\varepsilon\left((y_0(t) - D(x,u,\alpha))^{\tau} W(t) (y_0(t) - D(x,u,\alpha) + \rho \|\alpha(t) - \sigma(t)\|^2 \right) \\ & + p(t)^{\tau}\left(A(z,v,\sigma)x(t) + B(z,v,\sigma)u(t) + F(z,v,\sigma)\alpha(t) + \psi(z,v,\sigma) \right) \\ & + \lambda(t)^{\tau}(v(t) - u(t)) + \beta(t)^{\tau}(z(t) - x(t)) + \theta(t)^{\tau}(\sigma(t) - \alpha(t)) \end{aligned} \qquad \textbf{(11,19)}$$

where $\lambda(t) \in \Re^m$, $\beta(t) \in \Re^n$, $\theta(t) \in \Re^r$ are Lagrange multipliers and $p(t) \in \Re^n$ is the costate. Now, using a variational approach, the following necessary optimality conditions are obtained, in addition to (11,15) and (11,16):

$$u(t) = -\frac{1}{(1-\varepsilon)} R^{-1}(B(z,v,\sigma)^{\tau}p(t) - \lambda(t)) \qquad \textbf{(11,20)}$$

$$\dot{p} = -(1-\varepsilon)Qx(t) + A(z,v,\sigma)^{\tau}p(t) - \beta(t) \ , \ p(t_f) = 0 \qquad \textbf{(11,21)}$$

$$\lambda(t) = -(\frac{\partial Ax}{\partial v} + \frac{\partial Bu}{\partial v} + \frac{\partial S\alpha}{\partial v} + \frac{\partial \psi}{\partial v})^{\tau}p(t) + \varepsilon\frac{\partial D^{\tau}}{\partial v}W(y_o - D(z,v,\sigma))$$

$$\beta(t) = -(\frac{\partial Ax}{\partial z} + \frac{\partial Bu}{\partial z} + \frac{\partial S\alpha}{\partial z} + \frac{\partial \psi}{\partial z})^{\tau}p(t) + \varepsilon\frac{\partial D^{\tau}}{\partial z}W(y_o - D(z,v,\sigma))$$

$$\theta(t) = -(\frac{\partial Ax}{\partial \sigma} + \frac{\partial Bu}{\partial \sigma} + \frac{\partial S\alpha}{\partial \sigma} + \frac{\partial \psi}{\partial \sigma})^{\tau}p(t) + \varepsilon\frac{\partial D^{\tau}}{\partial \sigma}W(y_o - D(z,v,\sigma)) + \varepsilon\rho(\alpha-\sigma) \qquad \textbf{(11,22)}$$

$$\alpha(t) = \frac{1}{\varepsilon\rho}(\varepsilon\rho\sigma(t) - F(z,v,\sigma)^{\tau}p(t) + \theta(t)) \qquad \textbf{(11,23)}$$

Finally, a hierarchical algorithm is proposed for solving the joint problem as follows:

Algorithm 11.1.3: Hierarchical algorithm for solving the joint problem of systems identification and dynamic optimisation by Mahmoud *et al* (1980)

Step 0 — At level 4, select an initial guess for ε and send it to the lower levels. Set i=0.

Step 1 — At level 3, an initial estimate for the variables $L^0 = [\ z,\ v,\ \sigma,\ \lambda,\ \beta,\ \theta\ ]^{\tau}$ is predicted and sent to the lower levels. Set j=0.

Step 2 — At level 2, an initial trajectory for the costate $p(t)$ is predicted and sent to level 1. Set k=0.

Step 3 — At the level 1, using the supplied information, the control $u(t)$ is determined from (11,20), and the unknown vector $\alpha(t)$ is obtained

from (11,23). Then the state vector $x(t)$ is computed from (11,13). These results are transferred to level 2.

Step 4 — At level 2, compute the new costate $p^{k+1}(t)$ from (11,21). If $p^{k+1}(t) = p^{k}(t)$ then transfer $[\ x,\ u,\ \alpha,\ p\ ]$ to level 3. Else set $k = k+1$, send $p^{k+1}(t)$ to level one and repeat from step 3.

Step 5 — At level 3, a new set of variables $L^{j+1} = [\ z,\ v,\ \sigma,\ \lambda,\ \beta,\ \theta\ ]^{T}$ is determined from Equations (11,15) and (11,23). If $L^{j+1} = L^{j}$ continue with step 6, else set $j = j+1$, send L^{j+1} to levels 1 and 2 and repeat from step 3.

Step 6 — At level 4, update ε with $\varepsilon^{i+1} = k_{\varepsilon}\varepsilon^{i}$, where $k_{\varepsilon} \in (0,1)$ is a given step size, send ε to the lower levels and go to step 5.

As explained in the paper by Mahmoud *et al* (1980), a sequence of decreasing values of ε starting from a value lower than one to a value close to zero produces a solution which solves what the authors interpret as the joint problem of system identification and dynamic optimisation. After reading the introduction of the paper and the initial formulation of the joint problem a reader of that research work would expect that the resultant algorithm would involve some kind of interaction with the real plant in order to obtain the information from the output measurements necessary for identification purposes. However, this kind of interaction does not appear in Algorithm 11.1.3, and in the simulation example provided by the authors a fixed set of measured output data $y_o(t)$ is used. Thus, it appears that Algorithm 11.1.3 starts by finding a set of parameters $\alpha(t)$ and control vector $u(t)$ so that the model-based output variables are close to the set of previously measured output data $y_o(t)$, and then, as ε is gradually decreased, the dynamic performance index J is minimized. If this interpretation is correct, the identification stage, in the sense of computing the best parameters for a model structure given a set of input-output

information, is inadequate. Furthermore, the authors do not appear to take account of the input data from which the fixed set of output data was generated.

On the other hand, in the DISOPE approach interaction with the real system occurs, since at every iteration the values of the state variables and the real derivative information necessary for the optimisation is measured using the current values of the control trajectory (see, for instance, Chapter 8). Therefore the set of output data is not fixed but is measured as the iterations progress. The difference between the model-based and real derivative information is compensated, in addition to the difference between the real state response and the model based state predictions, and the real optimum of the plant is achieved in spite of the inaccuracies in the dynamic model of the process.

Finally, it appears that the type of joint system dynamic optimisation and parameter estimation problems dealt with by Mahmoud *et al* (1980) is different from that treated in this thesis.

11.2 SUMMARY

In this chapter the DISOPE technique has been placed within the general classification of nonlinear optimal control approaches as a function space method. Furthermore, similarities and differences between a DISOPE approach and a well established method such as quasilinearization have been drawn. Moreover, a comparison have been made with previous work by Hassan and Singh (1976), which consists of a two-level approach for the optimisation of nonlinear systems, where similarities between this approach and DISOPE have been discussed and the differences have been pointed out. Finally, an analysis has been made of the work of Mahmoud *et al* (1980), consisting of a hierarchical approach to the joint problem of systems identification and dynamic optimisation, and important differences between this approach and DISOPE have also been discussed.

CHAPTER 12

CONCLUSIONS

12.1 CONCLUSIONS

The development and applications of novel optimal control algorithms has been the central subject of the research work described in this thesis. An optimal control technique known as DISOPE (dynamic integrated system optimisation and parameter estimation), developed by Roberts (1992) in its continuous time, unconstrained and centralized form, has been the starting point for the project. Being the dynamic extension of ISOPE (Roberts, 1979), DISOPE was originally devised with the purpose of handling model-reality differences in dynamic optimisation. Following the objectives stated in Chapter 1, several extensions of the technique have been developed and tested in this thesis. Furthermore, potential applications of the algorithms developed have been proposed. Particular emphasis has been placed on possible uses of the techniques within the process industry.

Bearing in mind that in real processes control signals are usually constrained, an extension of DISOPE for handling control dependent constraints has been developed. The approach reduces to the handling of the control dependent constraints at the *model-based* level (or lower level), which would require the use of an iterative optimal control algorithm at this level. For computational reasons it may be convenient to use noniterative linear-quadratic methods at the model-based level. In the case of bound constraints on the control variables a simple saturation function has been used, such that this nonlinear function becomes part of the nonlinearity of the *real* problem, in such a way that convenient linear quadratic methods may be used at the lower level.

Many industrial processes are considered as being of large-scale. In order to address large-scale dynamic optimisation problems using DISOPE, a hierarchical extension of the technique has been proposed. The overall system is decomposed into more manageable sub-systems according to suitable criteria. The technique differentiates between real and model based problems, and the real dynamic optimum

is achieved by using a two-level hierarchical technique in which the resultant problem for each sub-system is independent from the others. The technique uses the interaction-prediction approach for handling the (linear) interactions between subsystems, while the internal dynamics of each subsystem may be nonlinear. The algorithm is suitable for parallel or distributed processing and savings in terms of memory storage are possible.

With the increasing sophistication and decreasing cost of computers, most modern control algorithms are implemented using digital computers. Digital controls are usually designed taking into account the discrete-time nature of the operation of computers. A discrete-time DISOPE algorithm has been proposed in this thesis. The discrete-time DISOPE algorithm with a linear quadratic model-based problem is computational attractive since the equations to be solved at the lower level are difference equations, as opposed to differential equations which arise in a continuous time formulation. Continuous dynamics are handled in an exact way by using an *exact discretization scheme*, based on modern differential equation solvers employed at the upper-level (in contrast with the use of crude discretizations of continuous time systems, such as Euler's method), while convenient linear quadratic discrete-time methods are used at the lower level.

When certain variables of a system are required to follow or track a reference trajectory a set-point tracking control problem arises. An extension of discrete-time DISOPE with linear quadratic model-based problem has been developed in such a way that a tracking problem with nonlinear dynamics and a not necessarily quadratic performance index may be solved by using standard linear-quadratic tracking formulations at the lower level. The use of quadratic incremental control weighting has been used for eliminating steady-state off-sets for constant reference trajectories.

In some cases and for several reasons (quality, safety, environmental regulations, etc.), some variables of a process (states or functions of the states) are constrained to lie within a particular region of the state space. The ability of an optimal control algorithm for handling this type of constraint is considered important. A technique for handling state dependent constraints within the DISOPE framework has been proposed in this work. The technique uses the penalty relaxation approach by which a penalty term is added to the real performance index. This penalty term is activated only when a constraint is violated. At the model-based level, the problem remains unconstrained and then standard linear quadratic methods may be used for

solving a path constrained optimal control problem. A technique for improving the convergence behaviour about the constrained optimum has been proposed.

The application of DISOPE for the dynamic optimisation of batch processes has been proposed. The algorithm achieves the correct dynamic optimum of the batch process in spite of deficiencies in the mathematical model used in the computations. The iterations of DISOPE are integrated with the batchwise operation of the plant, in such a way that there is a correspondence between batches and iterations. The use of the innovative *shadow model concept* (Griffiths, 1993) has been proposed for solving the problem of acquisition of unmeasured state variables and also of the derivative information required by DISOPE.

Model-based predictive control (based on linear input-output models of the process) has gained wide acceptance in the process industry in the last two decades (Soeterboek, 1992). Assuming that a nonlinear state-space model of the process is available for on-line prediction purposes, a nonlinear predictive controller has been developed, implemented and tested with simulations of chemical reactors. The technique uses the set-point tracking DISOPE algorithm in a receding horizon framework for the long-range predictions and dynamic optimisation. An Extended Kalman Filter is used for the estimation of states and uncertain parameters, in such a way that the controller has adaptive features. The certainty equivalence principle (Åström, 1970) has been used since the estimated parameters are considered to be exact when computing the optimal controls. Both set-point tracking and economic performance indexes have been used, which indicates the flexibility of the approach, and the controller is able to handle bound constraints on the controls and well as state dependent constraints.

Research on steady-state process optimisation using dynamic information has been presented in this thesis. In some cases, a nonlinear model of the process may not be available for on-line prediction purposes. Then, a linear state-space model of the process may be identified (Lin, 1993) using state measurements from the shadow model of the process, in such a way that the linear model adapts itself in the face of changes in operating conditions. With this information available two procedures have been developed for gradually driving a process from a suboptimal operating condition to its steady-state optimum. Firstly, a steady-state optimiser based on dynamic information (DSSO) has been developed. The approach has been derived using standard results from optimisation theory and discrete-time optimal control.

Based on the information available, a recursive control law has been derived and its steady-state optimality has been analyzed. Algorithms based on the control law derived have been formulated. The technique naturally handles control bound constraints as well as constraints on the rate of change of the control signals. The approach has been related with gradient descent approaches proposed in the past (Bamberger and Isermann, 1978). Secondly, the linear identified model has been used for producing (locally valid) long range predictions. The steady state objective is included in the dynamic performance index and DISOPE is used for solving a problem with linear dynamics, but taking into account control magnitude, control rate-of-change and state dependent constraints, in such a way that the optimal control problem to be solved in a receding horizon framework is nonlinear. The approach is abbreviated as LP-DISOPE. The steady-state optimality of the procedure has been analyzed and the real steady state optimum of the process may be achieved provided the prediction horizon is long enough. Predictions of important variables may be displayed to operators for different purposes (such as the evaluation of the performance of the optimiser and possible decisions based on such an evaluation). A clear advantage of the long range predictive control approach is that the saturation of (state and control dependent) constraints may be anticipated and appropriate action may be taken by the controller in advance. Both approaches have the advantage that it is not necessary to wait for the system to settle after every set-point change. Since the steady-state objective is included in the dynamic performance index, no separate steady-state optimiser is required. Furthermore, both optimising controllers should be able to track in a rapid way changes in the optimal operating conditions due to disturbances or price changes. Both LP-DISOPE and DSSO have been successfully applied to realistic simulations of an industrial multicomponent distillation column using a rigorous process simulator and a shadow model consisting of thousands of differential and algebraic equations, using thirty representative states for the linear model.

In summary, it may be said that the objectives stated in Chapter 1 have been achieved. Firstly, novel optimal control algorithms have been developed in this thesis, extending the initial formulation of the DISOPE approach for handling relevant and practical problems. Secondly, important applications within the process industry of the techniques developed have been proposed. The LP-DISOPE technique has been found attractive by an important firm within the process area, which has

allowed the technique to achieve a degree of maturity in terms of software implementation that its on-line application to a real process is feasible.

12.2 SUGGESTIONS FOR FURTHER RESEACH

As has been indicated, the existing knowledge of the DISOPE approach has been significantly enhanced during the research work reported in this thesis. Furthermore, relevant applications of the approach have been proposed. However, the author considers that research in this area is far from finished. On the contrary, important questions are yet to be answered. Some relevant areas for further research are described below.

(a) Convergence analysis of DISOPE. Even though work has been carried out in this particular area (Becerra and Roberts, 1994; Roberts, 1994) convergence proofs for the general case with a nonlinear real problem are yet to be developed. Furthermore, techniques for improving local convergence and ensuring a decrease of the performance index at each iteration should be developed.

(b) Further research is recommended in the hierarchical extension of the DISOPE approach. Particular emphasis should be made on its applicability to real systems or processes.

(c) The extension of the DISOPE approach for handling time-delays in the dynamics of the (in general nonlinear) real optimal control problem is suggested.

(d) The extension of the DISOPE approach for handling systems described by mixed differential and algebraic equations is recommended.

(e) Further studies are required regarding the feasibility of the application of DISOPE for the optimisation of batch processes. The computer hardware technology required for implementing the approach proposed is available in

the market. Partnership between academics and an interested industrial firm would be convenient for this kind of study.

(f) Given the potential and good performance of the nonlinear predictive control approach, as has been shown in this thesis and elsewhere (Wright and Edgar (1994); Balchen *et al*, (1992)), further research should be carried out on the application of DISOPE in this area. Particular emphasis should be made on practical applications.

(g) The application of LP-DISOPE for steady-state process optimisation using long-range predictions based on adaptive linear dynamic models of the process has achieved an important degree of maturity in terms of software implementation and realistic simulation experience. Given the performance of the approach (shown in the simulation studies of Chapter 10), and the attractive features associated with its predictive nature, further studies are recommended directed towards its on-line implementation on real industrial processes. Studies regarding the conditions for convergence of the LP-DISOPE approach and the (probably beneficial) effects of state weighting on the stability of the solution are also required.

REFERENCES

Amini-Largani Z. (1990). Algorithm development and analysis for on-line optimising control of large-scale industrial processes. PhD Thesis: City University. Department of Electrical, Electronic and Information Engineering, U.K.

Anderson B.D.O. and Moore J.B. (1989). *Optimal Control: Linear Quadratic Methods*. (London: Prentice Hall)

Arkun Y. and Stephanopoulos G. (1980). Optimising control of industrial chemical processes: state of the art review. In: *American Control Conference 1980 Proceedings*. Paper No. WP5-A.

Åström K.J. (1970). *Introduction to Stochastic Control Theory*. (New York: Academic Press).

Åström K.J. and Hägglund T. (1988). *Automatic Tuning of PID Controllers*. (Instrument Society of America).

Åström K.J. and Wittenmark B. (1989). *Adaptive Control*. (Reading, Massachusetts: Addison-Wesley).

Åström K.J. and Wittenmark B. (1990). *Computer Controlled Systems*. Second Edition. (Englewood, Cliffs: Prentice-Hall).

Balchen J.G., Ljungquist D. and Strand S. (1992). State space predictive control. *Chemical Engineering Science*. Vol 47 No. 4 pp. 787-807

Balchen J.G. and Mumme K.I. (1988). *Process Control: Structures and Applications*. (New York: Van Nostrand Reinhold Company)

Bamberger W. and Isermann R. (1978). Adaptive on-line steady-state optimisation of slow dynamic processes. *Automatica*. Vol. 14, pp. 223-230.

Becerra V.M. (1993a). Dynamic Integrated System Optimisation and Parameter Estimation for optimal control of processes with model-reality differences - C++ version. City University, Control Engineering Research Centre, London. Internal Report CERC/VB/120: March 1993.

Becerra V.M. (1993b). Dynamic Integrated System Optimisation and Parameter Estimation for discrete time optimal control of nonlinear systems. City University, Control Engineering Research Centre, London. Internal Report CERC/VB/123: March 1993.

Becerra V.M. (1993c). On a hierarchical extension of the DISOPE algorithm. City University, Control Engineering Research Centre, London. Internal Report CERC/VB/129: May 1993.

Becerra V.M. (1993d). On the solution of the nonlinear tracking problem by using the DISOPE algorithm. City University, Control Engineering Research Centre, London. Internal Report CERC/VB/130: May 1993.

Becerra V.M. (1993e). An adaptive nonlinear predictive controller using the DISOPE algorithm and an Extended Kalman Filter. City University, Control Engineering Research Centre, London. Internal Report CERC/VB/131: August 1993.

Becerra V.M. and Roberts P.D. (1993). Extension of the DISOPE algorithm for handling optimal control problems with input-dependent inequality constraints. City University, Control Engineering Research Centre, London, Internal Report CERC/VMB-PDR/127: April 1993.

Becerra V.M. and Roberts P.D. (1994a). Dynamic integrated system optimisation and parameter estimation for discrete time optimal control of nonlinear systems. Paper submitted to: *International Journal of Control*.

Becerra V.M. and Roberts P.D. (1994b). A new approach for the solution of the optimal set-point tracking problem for nonlinear systems. Paper submitted to: *Control 94*, Coventry U.K.

Bequete B.W. (1991). Nonlinear Control of Chemical Processes: A review. *Industrial Engineering Chemical Research.* Vol 30 pp.1391-1413.

Bitmead R.R., Gevers M. and Wertz V. (1990). *Adaptive Optimal Control: The thinking man's GPC.* (New York: Prentice Hall)

Bellman R.E. and Dreyfus S.E. (1962). *Applied Dynamic Programming*. (Princeton, New Jersey: Princeton University Press).

Biegler L.T. (1984). Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers and Chemical Engineering*. Vol.8, No. 3/4, pp. 243-248.

Brdyś M. and Roberts P.D. (1987). Convergence and optimality of modified two-step algorithm for integrated system optimisation and parameter estimation. *International Journal of Systems Science*, Vol. 18, pp 213-217.

Brdyś M., Ellis J.E. and Roberts P.D. (1987). Augmented Integrated System Optimisation and Parameter Estimation technique: derivation, optimality and convergence. *IEE Proceedings*. Vol 14, Part D, No. 3, pp. 201-209.

Bryson A.E. and Ho Y.C. (1975). *Applied Optimal Control.* Revised Printing (New York: Hemisphere Publishing Corporation).

Clarke D.W., Mohtadi C. and Tuffs P.S. (1987). Generalized Predictive Control-part I. The basic algorithm. *Automatica*. Vol. 23, No. 2, pp. 137-148.

Cutler C.R. and Remaker B.L. (1979). Dynamic Matrix Control: a computer control algorithm. In: *AIChE 86th National Meeting*, Houston, Texas: April 1979.

Durbeck R.C. (1965). Principles for simplification of optimising control models. PhD Thesis: Case Western Reserve University, U.S.A.

Eaton J.W. and Rawlings J.B. (1990). Feedback control of chemical processes using on-line optimisaton techniques. *Computers and Chemical Engineering*. Vol. 14. No. 4/5, pp. 469-479.

Edgar T.F. and Himmelblau D.M. (1988). *Optimisation of Chemical Processes*. (New York: McGraw-Hill).

Fletcher R. (1981). *Practical Methods of Optimisation 2*. (Chichester: Wiley)

Frankin G.F., Powell J.D. and Workman M.L. (1990). *Digital Control of Dynamic Systems*. (Reading, Massachusetts: Addison-Wesley).

Foord A.G. (1974). On-line optimisation of a petrochemical complex. PhD Thesis: University of Cambridge, U.K.

Garcia C.E. and Morshedi A.M. (1986). Quadratic programming solution of Dynamic Matrix Control (QDMC). *Chemical Engineering Communications*. Vol. 46 pp. 073-087

Garcia C.E., Prett D.M. and Morari M. (1989). Model predictive control: theory and practice -a survey. *Automatica*. Vol 25 No.3 pp. 335-348

Garcia C.E. and Morari M. (1981). Optimal operation of integrated processing systems. Part I: open-loop on-line optimising control. *AICHE Journal*. Vol 27. N0. 6, pp. 960-968.

Gattu G. and Zafiriou E. (1992). Nonlinear Quadratic Dynamic Matrix Control with state estimation. *Industrial Engineering Chemical Research*. Vol. 31, pp. 1096-1104.

Gorlen K.E., Orlow S.M. and Plexico P.S. (1990). *Data Abstraction and Object-Oriented Programming in C++*. (Chichester: Wiley).

Griffiths G.W. (1993). The Woodside Trunkline Management System. Paper Presented at: *12th International OMAE Symposium.* Glasgow: 20-24 June, 1993.

Hassan M. and Singh M.G. (1976). Optimisation of nonlinear systems using a new two level method. *Automatica.* Vol. 12, pp. 359-363.

Isermann R., Lachmann K.H. and Matko. D. (1992). *Adaptive control systems.* (London: Prentice-Hall).

Jamshidi M. (1983). *Large-Scale Systems: Modelling and Control.* (New York: Elsevier Science Publishing).

Jang S.S. Joseph B. and Mukay H. (1987). On-line optimisation of constrained multivariable chemical processes. *AICHE Journal.* Vol 33, No. 1, pp. 2635.

Johnson A. (1993). LQG applications in the process industry. *Chemical Engineering Science.* Vol.48, No. 16, pp. 2829-2838.

Jones D.I. and Finch J.W. (1984). Comparison of optimisation algorithms. *International Journal of Control.* Vol 40, No. 4, pp. 747-761.

Kirk D.E. (1970). *Optimal Control Theory.* (Englewood, Cliffs: Prentice-Hall).

Lapidus L. and Luus R. (1967). *Optimal Control of Engineering Processes.* (Waltham, Massachusetts: Ginn/Blaisdell).

Leigh J.R. (1992). *Applied Digital Control: Theory, Design and Implementation.* 2nd. Edition. (New York: Prentice Hall).

Lewis F.L. (1986a). *Optimal Control.* (New York: John Wiley & Sons).

Lewis F.L. (1986b). *Optimal Estimation.* (New York: John Wiley & Sons).

Lin J. (1993). Personal communication.

Lin J., Kambhampati C. and Roberts P.D. (1989). A new two model approach for optimising control of large-scale steady state systems. *IMA Journal of Mathematical Control and Information*. Vol. 6, pp. 39-61.

Lin J. and Griffiths G.W. (1992). An approach to on-line optimisation. SAST Technical Bulletin No. 3. SAST Limited, U.K.: May 1992.

Ljung L. (1987). *System Identification - Thery for the User.* (Englewood Cliffs: Prentice Hall)

Mahmoud M.S., Hassan M.F. and Singh M.G. (1980). A new hierarchical approach to the joint problem of systems identification and optimisation. *Large Scale Systems*. Vol. 1. pp. 159-60.

Mahmoud M.S., Vogt W.G., and Mikcle M.H. (1978). Multilevel control and optimisation using generalized gradient techniques. *International Journal of Control*. Vol. 25, pp. 525-543.

Maybeck P.S. (1982). *Stochastic Models, Estimation and Control* **2**. (New York: Academic Press).

Mayne D.Q. and Michalska H. (1990). Receding Horizon Control of nonlinear systems. *IEEE Transactions on Automatic Control*. Vol. 35, No. 7, pp. 814-824.

Papageorgiou M. and Schmidt G. (1980). On the hierarchical solution of nonlinear optimal control problems. *Large Scale Systems*. Vol 1., pp. 265-271.

Pontryagin L.S., Boltyanskii R.V., Gamkrelidze R.V. and Mishchenko E.F. (1962). *The Mathematical Theory of Optimal Processes*. (New York: Wiley Interscience).

Press W.H., Flannery B.P., Teukolsky S.A. and Vetterling W.T. (1992). *Numerical Recipes in C*. Second Edition. (Cambridge: Cambridge University Press).

Quintana V.H. and Davison E.J. (1974). Clipping-off gradient algorithms to compute optimal controls with constrained magnitude. *International Journal of Control*. Vol. 20, No. 2, pp. 243-255.

Ray W.H. (1981). *Advanced Process Control.* (New York: McGraw-Hill).

Richalet J.A., Rault A., Testud J.D. and Papon J (1978). Model Predictive Heuristic Control: applications to industrial processes. *Automatica*. Vol. 14. pp. 413-428.

Rijnsdorp J.E. (1991). *Integrated Process Control and Automation.* (Amsterdam: Elsevier Science Publishers)

Roberts P.D. (1979). An algorithm for steady state system optimisation and parameter estimation. *International Journal of Systems Science*, Vol. 10, pp.719-734.

Roberts P.D. (1992). Optimal control of non-linear systems with model-reality differences". In: *31st. Conference on Decision and Control Proceedings*, pp. 257-258. Tucson, Arizona: December 16-18 1992

Roberts P.D. (1993a). An algorithm for optimal control of nonlinear systems with model-reality differences. In: *IFAC World Congress 1993 Proceedings*. Vol. 8, pp. 407-412. Sydney, Australia: August 1993.

Roberts P.D. (1993b). Convergence analysis of the DISOPE algorithm. Unpublished manuscript.

Roberts P.D. (1994). Unit Memory Repetitive Processes and iterative optimal control algorithms. Paper submitted to: *Control 94*, Coventry, U.K.

Roberts P.D., Wan B. and Lin J. (1992). Steady state hierarchical control of large scale industrial processes: a survey. Plenary paper presented at: *IFAC/IFORS/IMACS Symposium: Large Scale Systems: Theory and Applications*. Beijing, P.R. China: August 23-25, pp. 1-10.

Rockafellar R.T. (1970). *Convex Analysis.* (Princeton, New Jersey: Princeton University Press).

Rockafellar R.T. (1974). Aumented Lagrange multiplier functions and duality in nonconvex programming. *SIAM Journal of Control.* Vol. 12, pp. 268-285.

Rolf M.J. and Henry C.L. (1984). Adaptive on-line optimisation for continuous bioreactors. *Chemical Engineering Communications.* Vol. 29, pp. 229-255.

Sage A.P. and White C.C. (1977). *Optimum Systems Control.* Second Edition. (Englewood Cliffs, New Jersey: Prentice Hall)

Sakawa Y. and Shindo Y. (1980). On global convergence of an algorithm for optimal control. *IEEE Transactions on Automatic Control.* Vol. AC-25, No. 6, pp. 1149-1153.

Sargent R.W.H. and Sullivan G.R. (1978). The development of an efficient optimal control package. In: *8th IFIP Conference on Optimisation Techniques Proceedings.* Part 2, pp. 158-168.

Seborg D.E., Edgar T.F. and Mellichamp D.A. (1989). *Process Dynamics and Control* (New York: John Wiley & Sons).

Singh M.G. (1980). *Dynamical Hierarchical Control.* (Amsterdam: North-Holland Publishing Company).

Singh M.G. and Titli A. (1978). *Systems: Decomposition, Optimisation and Control.* (Oxford: Pergamon Press).

Sisirena H.R. and Tan K.S. (1974). Computation of constrained optimal controls using parametrization techniques. *IEEE Transactions on Automatic Control*, Vol. AC-19, pp. 431-433.

Sistu P.B. and Bequette W. (1992). Nonlinear predictive control of uncertain processes: application to a CSTR. *AICHE Journal.* Vol 37, No. 11, pp. 1711-1723.

Soeterboek R. (1992). *Predictive Control: a Unified Approach.* (London: Prentice Hall).

Sötherstrom T. and Stoica P. (1989). *System Identification.* (New York: Prentice-Hall).

Strand S. (1991). Dynamic optimisation in state space predictive control schemes. Dr. Ing. Thesis: Norwegian Institute of Technology, Division of Engineering Cybernetics, Report No. NHT9111W, Norway.

Teo K.L., Goh C.J. and Wong K.H. (1991). *A Unified Computational Approach to Optimal Control Problems.* (London: Longman Scientific & Technical).

Wright G.T. and Edgar T.F. (1994). Nonlinear model predictive control of a fixed-bed water-gas shift reactor: an experimental study. *Computers in Chemical Engineering*. Vol. 18, No. 2, pp. 83-102.

Zafiriou E. and Zhu J. (1989). Optimal feed rate profile determination for fed-batch fermentations in the presence of model-plant mismatch. In: *American Control Conference 1989 Proceedings*. Paper No. FA4-11:15, pp. 2006-2009.

APPENDIX

A. DERIVATION OF PROCEDURE 2.3.1

Here we want to solve the following two-point boundary value problem (TPBVP)

$$\begin{aligned} \dot{x} &= Ax(t)+B\bar{R}^{-1}(B^{\top}p(t)-\bar{\lambda}(t)) + \alpha(t) \\ \dot{p} &= -\bar{Q}x(t)-A^{\top}p(t) + \bar{\beta}(t) \end{aligned} \tag{A,1}$$

with border conditions:

$$\begin{aligned} x(t_0) &= x_0 \\ p(t_f) &= \Phi x(t_f) \end{aligned} \tag{A,2}$$

where all the quantities are defined in Chapter 2 and the control law is given by:

$$u(t) = -\bar{R}^{-1}(B^{\top}p(t) - \bar{\lambda}(t)) \tag{A,3}$$

This linear TPBVP can be solved by using the backward sweep method (Bryson and Ho, 1975; Lewis, 1986a). The key is to assume the relationship between costate and state as

$$p(t) = K(t)x(t) + k(t) \tag{A,4}$$

where $K(t)$ is a nxn matrix and $k(t)\in\Re^n$. Then we have:

$$\dot{p}(t) = \dot{K}(t)x(t) + K(t)\dot{x}(t) + \dot{k}(t) \tag{A,5}$$

From (A,1) we obtain

$$\dot{p}(t) = \dot{K}(t)x(t) + K(t)(Ax(t)-B\bar{R}^{-1}(B^{\top}p(t)-\bar{\lambda}(t)) + \alpha(t)) + \dot{k}(t) \tag{A,6}$$

$$\dot{p}(t) = \dot{K}(t)x(t) + K(t)Ax(t) - K(t)B\bar{R}^{-1}B^{\top}(K(t)x(t) - k(t)) + K(t)B\bar{R}^{-1}\bar{\lambda}(t) - K(t)\alpha(t) + \dot{k}(t) \quad \textbf{(A,7)}$$

$$\dot{p}(t) = (\dot{K}(t) + K(t)A - K(t)B\bar{R}^{-1}B^{\top}K(t))x(t) - K(t)B\bar{R}^{-1}B^{\top}k(t) + K(t)B\bar{R}^{-1}\bar{\lambda}(t) - K(t)\alpha(t) + \dot{k}(t) \quad \textbf{(A,8)}$$

But, from (A,1) we have

$$\dot{p}(t) = (-\bar{Q} - A^{\top}K(t))x(t) - A^{\top}k(t) + \bar{\beta}(t) \quad \textbf{(A,9)}$$

Comparison between (A,8) and (A,9) gives

$$\dot{K}(t) = K(t)B\bar{R}^{-1}B^{\top}K(t) - K(t)A - A^{\top}K(t) - \bar{Q} \quad \textbf{(A,10)}$$

$$\dot{k}(t) = K(t)B\bar{R}^{-1}B^{\top}k(t) - A^{\top}k(t) - K(t)B\bar{R}^{-1}\bar{\lambda}(t) - K(t)\alpha(t) + \bar{\beta}(t) \quad \textbf{(A,11)}$$

with terminal conditions $K(t_f) = \Phi$ and $k(t_f) = 0$. Substitution of (A,4) in (A,3) gives:

$$u(t) = -\bar{R}^{-1}B^{\top}K(t)x(t) - \bar{R}^{-1}B^{\top}k(t) - \bar{R}^{-1}\bar{\lambda}(t) \quad \textbf{(A,12)}$$

Define:

$$G(t) = \bar{R}^{-1}B^{\top}K(t)$$
$$g(t) = -\bar{R}^{-1}(B^{\top}k(t) - \bar{\lambda}(t)) \quad \textbf{(A,13)}$$

Then we can write the control law as

$$u(t) = -G(t)x(t) + g(t) \quad \textbf{(A,14)}$$

Substitution of (A,14), (B,23) in the state equation $\dot{x} = Ax(t) + Bu(t) + \alpha(t)$ gives:

$$\dot{x} = (A - BG(t))x(t) + Bg(t) + \alpha(t) \quad \textbf{(A,15)}$$

A straightforward reasoning on the dependence of the different variables involved, gives rise to Procedure 2.3.1.

B. DERIVATION OF PROCEDURE 2.3.2

Here we will include terminal constraints in the values of the state variables $x_i(t_f)=0,\ i\in[1,q]$, which can be written as follows

$$\psi(x(t_f)) = Cx(t_f) = 0 \tag{B,1}$$

where $C = [I_q|0]$ is a qxn matrix.

Thus, we want to solve the following two-point boundary value problem (TPBVP)

$$\begin{aligned} \dot{x} &= Ax(t)+B\bar{R}^{-1}(B^{\top}p(t)-\bar{\lambda}(t)) + \alpha(t) \\ \dot{p} &= -\bar{Q}x(t)-A^{\top}p(t) + \bar{\beta}(t) \end{aligned} \tag{B,2}$$

with border conditions:

$$x(t_0) = x_0 \tag{B,3}$$

$$p(t_f) = \Phi x(t_f) + C^{\top}\nu$$

where $\nu\in\Re^q$ is a Lagrange multiplier to be determined such that (B,1) is satisfied. The control law is given by:

$$u(t) = -\bar{R}^{-1}(B^{\top}p(t) - \bar{\lambda}(t)) \tag{B,4}$$

This linear TPBVP can be solved by using the backward sweep method (Bryson and Ho, 1975; Lewis, 1986a). The key is to assume the relationship between costate and state as

$$p(t) = K(t)x(t)+k(t)+F(t)\nu \tag{B,5}$$

and express the (assumed fixed) terminal constraint function as

$$\psi = F(t)^{\tau}x(t) + W(t)\nu + \eta(t) \qquad \textbf{(B,6)}$$

where $K(t)$ is a *nxn* matrix, $F(t)$ is a *nxq* matrix, $W(t)$ is a *qxq* matrix, $k(t)\in\Re^n$, and $\eta(t)\in\Re^q$. Then we have:

$$\dot{p}(t) = \dot{K}(t)x(t) + K(t)\dot{x}(t) + \dot{k}(t) + \dot{F}(t)\nu \qquad \textbf{(B,7)}$$

From (B,2) we obtain

$$\dot{p}(t) = \dot{K}(t)x(t) + K(t)(Ax(t) - B\bar{R}^{-1}(B^{\tau}p(t) - \bar{\lambda}(t)) + \alpha(t)) + \dot{k}(t) + \dot{F}(t)\nu \qquad \textbf{(B,8)}$$

$$\dot{p}(t) = \dot{K}(t)x(t) + K(t)Ax(t) - K(t)B\bar{R}^{-1}B^{\tau}(K(t)x(t) + k(t) + F(t)\nu) + K(t)B\bar{R}^{-1}\bar{\lambda}(t) - K(t)\alpha(t) + \dot{k}(t) + \dot{F}(t)\nu \qquad \textbf{(B,9)}$$

$$\dot{p}(t) = (\dot{K}(t) + K(t)A - K(t)B\bar{R}^{-1}B^{\tau}K(t))x(t) - K(t)B\bar{R}^{-1}B^{\tau}k(t) + (K(t)B\bar{R}^{-1}B^{\tau}F(t) + \dot{F}(t))\nu + K(t)B\bar{R}^{-1}\bar{\lambda}(t) - K(t)\alpha(t) + \dot{k}(t) \qquad \textbf{(B,10)}$$

But, from (B,2) we have

$$\dot{p}(t) = (-\bar{Q} - A^{\tau}K(t))x(t) - A^{\tau}k(t) + \bar{\beta}(t) - A^{\tau}F(t)\nu \qquad \textbf{(B,11)}$$

Comparison between (B,10) and (B,11) gives

$$\dot{K}(t) = K(t)B\bar{R}^{-1}B^{\tau}K(t) - K(t)A - A^{\tau}K(t) - \bar{Q} \qquad \textbf{(B,12)}$$

$$\dot{k}(t) = K(t)B\bar{R}^{-1}B^{\top}k(t) - A^{\top}k(t) - K(t)B\bar{R}^{-1}\bar{\lambda}(t) - K(t)\alpha(t) + \bar{\beta}(t) \quad \textbf{(B,13)}$$

$$\dot{F}(t) = K(t)B\bar{R}^{-1}B^{\top}F(t) - A^{\top}F(t) \quad \textbf{(B,14)}$$

with terminal conditions $K(t_f) = \Phi$, $k(t_f) = 0$ and $F(t_f) = C^{\top}$. In addition, since we have that the terminal constraint function (B,6) is assumed constant, then its time derivative vanishes:

$$\frac{d\psi}{dt} = \frac{d}{dt}(F(t)^{\top}x(t) + W(t)\nu + \eta(t)) = \dot{F}^{\top}(t)x(t) + F(t)^{\top}x(t) + \dot{W}(t)\nu + \dot{\eta}(t) = 0 \quad \textbf{(B,15)}$$

Substitution of (B,2) in (B,15) gives, using (B,5) and grouping terms

$$(\dot{F}(t) + F^{\top}(t)A - F^{\top}(t)B\bar{R}^{-1}B^{\top}K(t))x(t) + (-F^{\top}(t)B\bar{R}^{-1}B^{\top}F(t) + \dot{W}(t))\nu + (-F(t)B\bar{R}^{-1}B^{\top}k(t) + F^{\top}(t)B\bar{R}^{-1}\bar{\lambda}(t) + F^{\top}(t)\alpha(t) + \eta(t)) = 0 \quad \textbf{(B,16)}$$

Equating coefficients in (B,16) to zero results in a repeat of (B,14) plus the following differential equations which can be solved backwards from $W(t_f) = 0$ and $\eta(t_f) = 0$.

$$\dot{W}(t) = F(t)^{\top}B\bar{R}^{-1}B^{\top}F(t) \quad \textbf{(B,17)}$$

$$\dot{\eta}(t) = F(t)^{\top}B\bar{R}^{-1}(B^{\top}k(t) - \bar{\lambda}(t)) - F(t)^{\top}\alpha(t) \quad \textbf{(B,18)}$$

We need to find an expression for the multiplier ν. Then, from the terminal constraint function (B,6) we obtain:

$$\nu = W(t)^{-1}(F(t)^{\tau}x(t) - \eta(t)) \qquad \textbf{(B,19)}$$

which is valid for all $t\in[t_0,t_f]$. However, noticing that $W(t_f)$ is singular, we may write, for instance:

$$\nu = W(t_0)^{-1}(F(t_0)^{\tau}x_0 - \eta(t_0)) \qquad \textbf{(B,20)}$$

If $W(t)$ is singular for all $t\in[t_0,t_f]$, then the system is not controllable and the terminal constraint may not be satisfied (Bryson and Ho, 1975).

Substitution of (B,5) in (B,4) gives:

$$u(t) = -\bar{R}^{-1}B^{\tau}K(t)x(t) \; -\bar{R}^{-1}B^{\tau}(k(t) + F(t)^{\tau}\nu) - \bar{R}^{-1}\bar{\lambda}(t) \qquad \textbf{(B,21)}$$

Define:

$$\begin{aligned} G(t) &= \bar{R}^{-1}B^{\tau}K(t) \\ g(t) &= -\bar{R}^{-1}(\, B^{\tau}(k(t)+F(t)\nu) - \bar{\lambda}(t)\,) \end{aligned} \qquad \textbf{(B,22)}$$

Then we can write the control law as

$$u(t) = -G(t)x(t) + g(t) \qquad \textbf{(B,23)}$$

Substitution in the state equation $\dot{x} = Ax(t) + Bu(t) + \alpha(t)$ gives:

$$\dot{x} = (A - BG(t))x(t) + Bg(t) + \alpha(t) \qquad \textbf{(B,24)}$$

A straightforward reasoning on the dependence of the different variables involved, gives rise to Procedure 2.3.2.

C. A CONJUGATE-GRADIENT OPTIMAL CONTROL ALGORITHM

A multiple-input extension of the single-input conjugated-gradient algorithm presented in (Quintana and Davison, 1974) is used as an auxiliary algorithm for solving the modified model-based problem in Algorithm 3.2.1. The algorithm is easy

to implement and its convergence has been proven for any arbitrary and feasible initial estimate of the optimal control (Quintana and Davison, 1974). The key concepts of this algorithm are the numerical integration of the state and costate equations, and a gradient in function space to update the controls which are clipped-off at the bounds so as to minimize the Hamiltonian. The Quintana-Davison (QD) algorithm solves the following class of optimal control problems:

$$\min_{u(t)} J = \phi(x(t_f)) + \int_{t_0}^{t_f} \mathfrak{L}(x(t),u(t),t)\,dt$$

subject to

$$\dot{x} = F\ (x(t),u(t),t)$$

$$x(t_0)=x_o$$

$$u_{\min} \le u(t) \le u_{\max}$$

where $\phi : \Re^n \to \Re$ is a given terminal weighting, $\mathfrak{L} : \Re^n \times \Re^m \times \Re \to \Re$ is a performance measure and $F : \Re^n \times \Re^m \times \Re \to \Re^n$ is a dynamic model.

Define:

$$\mathrm{H} = \mathfrak{L}(x(t),u(t),t) + p^{\mathrm{T}}(t) F\ (x(t),u(t),t) \qquad \textbf{(C,1)}$$

where

$$\dot{p}(t) = -\nabla_x \mathrm{H};\quad p(t_f) = \nabla_x \phi_{t=t_f} \qquad \textbf{(C,2)}$$

Define the signum function as:

$$sgn(\chi)=\begin{cases} 1 & \chi \ge 0 \\ -1 & \chi < 0 \end{cases} \qquad \textbf{(C,3)}$$

where χ is a scalar argument.

Define the j-th diagonal element of the *mxm* saturation matrix $S(t)$ as:

$$S_{jj}^{i}(t) = \begin{cases} 0: & u_j^{i}(t)=u_{\max_j} \text{ or } u_j^{i}(t)=u_{\min_j} \text{ and } sgn(u_j^{i}(t))=-sgn(g_{u_j}^{i}(t)) \\ 1: & \text{otherwise} \end{cases} \quad \textbf{(C,4)}$$

and the off-diagonal element as:

$$S_{jk}^{i}(t)=0;\ j\neq k$$

Define the vector saturation function as:

$$SAT(u(t)) = \begin{bmatrix} sat(u_1) \\ \dots \\ sat(u_m) \end{bmatrix} \quad \textbf{(C,6)}$$

where the scalar saturation function is given by:

$$sat(u_j) = \begin{cases} u_{\max_j} & u_j>u_{\max_j} \\ u_j & u_{\min_j}\leq u_j\leq u_{\max_j} \\ u_{\min_j} & u_j<u_{\min_j} \end{cases} \quad \textbf{(C,7)}$$

Define the following quantity:

$$\zeta^{i} = \begin{cases} \kappa_0^{i} & \kappa_0^{i}<\kappa_m^{i} \\ 0.5\kappa_m^{i} & \kappa_0^{i}\geq\kappa_m^{i} \\ 0 & \pi_2^{i}=0 \text{ or } \pi_3^{i}=0 \end{cases} \quad \textbf{(C,8)}$$

where

$$\kappa_0^{i}=\frac{\pi_1^{i}}{\pi_2^{i}},\ \pi_2^{i}\neq 0$$
$$\kappa_m^{i}=\frac{\pi_1^{i}}{\pi_3^{i}},\ \pi_3^{i}\neq 0 \quad \textbf{(C,9)}$$

where the inner products π_1^{i}, π_2^{i} and π_3^{i} are given by:

$$\pi_1^i = \int_{t_0}^{t_f} g_u^i(t)^T S^i(t) g_u^i(t) dt$$

$$\pi_2^i = \int_{t_0}^{t_f} g_u^{i-1}(t)^T S^i(t) g_u^{i-1} dt \qquad \textbf{(C,10)}$$

$$\pi_3^i = \int_{t_0}^{t_f} g_u^i(t)^T S^i(t) d^i(t) dt$$

Given the above definitions, the following multiple-input extension of the Quintana-Davison (QD) algorithm is described as follows:

Algorithm C.1: Multiple input extension of the conjugate-gradient algorithm by Quintana and Davison (1974)

Step a — Choose an initial estimate $u^0(t) \in \Omega$, determine $g_u^0 = \nabla_{u^0} H$ and set $d^0(t) = -g_u^0(t)$, $S^0(t) = I_m$, $t \in [t_0, t_f]$.

Step b — Choose the step length $\theta^0 \geq 0$ to minimize $J[SAT(u^0(t) + \theta^0 S^0(t) d(t))]$. Let $u^1(t) = SAT(u^0(t) + \theta^0 S^0(t) d^0(t))$ and $i = 0$.

Step c — Set $i = i+1$, compute the new saturation matrix $S^i(t)$ and determine the direction of search as follows: $d^i(t) = -g_u^i(t) + \zeta^i S^i(t) d^{i-1}(t)$.

Step d — Choose $\theta^i \geq 0$ to minimize $J[SAT(u^i(t) + \theta^i S^i(t) d^i(t))]$. Let $u^{i+1}(t) = SAT(u^i(t) + \theta^i S^i(t) d^i(t))$.

Step e — Repeat steps c and d until $|J[u^{i+1}] - J[u^i]| \leq \delta J[u^{i+1}]$ where δ is a given tolerance.

In order to improve convergence, the algorithm may be reset to steepest descent (i.e. set $\zeta^i = 0$) every certain number of iterations (typically 2-10).

D. DERIVATION OF PROCEDURE 5.3.1

Here we want to solve the following discrete-time two-point boundary value problem (TPBVP)

$$\begin{aligned} x(k+1) &= Ax(k)+B\bar{R}^{-1}(B^{\top}p(k+1)-\bar{\lambda}(k)) + \alpha(k) \\ p(k) &= -\bar{Q}x(k) + A^{\top}p(k+1) - \bar{\beta}(k) \end{aligned} \qquad \textbf{(D,1)}$$

with border conditions:

$$\begin{aligned} x(N_0) &= x_0 \\ p(N_f) &= \Phi x(N_f) \end{aligned} \qquad \textbf{(D,2)}$$

where all the quantities are defined in Chapter 5 and the control law is given by:

$$u(k) = -\bar{R}^{-1}(B^{\top}p(k+1) - \bar{\lambda}(k)) \qquad \textbf{(D,3)}$$

This linear TPBVP can be solved by using the backward sweep method (Bryson and Ho, 1975; Lewis, 1986a). The key is to assume the relationship between costate and state as

$$p(k) = S(k)x(k)+h(k) \qquad \textbf{(D,4)}$$

where $S(k)$ is a nxn matrix and $h(k)\in\Re^n$. Substituting (D,4) in the first equation in (D,1) we obtain:

$$x(k+1) = Ax(k)+B\bar{R}^{-1}(B^{\top}(S(k+1)x(k+1)+h(k+1))-\bar{\lambda}(k)) + \alpha(k) \qquad \textbf{(D,5)}$$

Grouping terms results in the following:

$$\begin{aligned} x(k+1) = (I_n+B\bar{R}^{-1}B^{\top}S(k+1))^{-1}(Ax(k) + B\bar{R}^{-1}\bar{\lambda}(k) \\ + \alpha(k) - B\bar{R}^{-1}B^{\top}h(k+1)) \end{aligned} \qquad \textbf{(D,6)}$$

Substituting (D,4) in the second equation in (D,1) we have

$$S(k)x(k) + h(k) = \bar{Q}x(k) + A^{\top}S(k+1)x(k+1) + A^{\top}h(k+1)-\bar{\beta}(k) \qquad \textbf{(D,7)}$$

Substituting (D,6) in (D.7) and grouping terms:

$$
\begin{aligned}
&[\,-S(k)+\bar{Q}+A^{\top}S(k+1)[I_n+B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}A\,]\,x(k)\\
&+[A^{\top}-A^{\top}S(k+1)[I_n+B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}B\bar{R}^{-1}B^{\top}]\,h(k+1)-h(k)-\bar{\beta}(k)\\
&+A^{\top}S(k+1)[I_n+B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}[B\bar{R}^{-1}\bar{\lambda}(k)+\alpha(k)] \;=\; 0
\end{aligned}
\qquad \textbf{(D,8)}
$$

Equating coefficients to zero in (D,8) results in the following set of difference equations which can be solved backwards from the terminal conditions $S(N_f) = \Phi$ and $h(N_f) = 0$.

$$
S(k) \;=\; \bar{Q} \;+\; A^{\top}S(k+1)[I_n \;+\; B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}A \qquad \textbf{(D,9)}
$$

$$
\begin{aligned}
h(k) \;=\; &[A^{\top}-A^{\top}S(k+1)[I_n \;+\; B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}B\bar{R}^{-1}B^{\top}]\,h(k+1)\\
&-\bar{\beta}(k) \;+\; A^{\top}S(k+1)[I_n+B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}[B\bar{R}^{-1}\lambda(k)+\alpha(k)]
\end{aligned}
\qquad \textbf{(D,10)}
$$

By using the matrix inversion lemma (See, for instance, Lewis (1986a)) we obtain the following equivalence:

$$
[I_n \;+\; B\bar{R}^{-1}B^{\top}S(k+1)]^{-1} \;=\; I_n \;-\; B[\bar{R} \;+\; B^{\top}S(k+1)B]^{-1}B^{\top}S(k+1) \qquad \textbf{(D,11)}
$$

and define $G(k)$ as:

$$
G(k) \;=\; [\bar{R} \;+\; B^{\top}S(k+1)B]^{-1}B^{\top}S(k+1)A \qquad \textbf{(D,12)}
$$

Now, we can re-write (D,9) and (D,10) as follows:

$$
S(k) \;=\; \bar{Q} \;+\; A^{\top}S(k+1)(A-BG(k)) \qquad \textbf{(D,13)}
$$

$$
\begin{aligned}
h(k) \;=\; &(A-BG(k))^{\top}h(k+1) \;+\; (A-BG(k))^{\top}S(k+1)\,\alpha(k)\\
&-\bar{\beta}(k) \;+\; G(k)^{\top}\bar{\lambda}(k)
\end{aligned}
\qquad \textbf{(D,14)}
$$

Substituting (D,4) in (D,3) we obtain:

$$u(k) = -\bar{R}^{-1}B^{\tau}[S(k+1)x(k+1) + h(k+1)] + \bar{R}^{-1}\lambda(k) \qquad \textbf{(D,15)}$$

which can be re-written as follows:

$$u(k) = -G(k)x(k) + g(k) \qquad \textbf{(D,16)}$$

where

$$g(k) = [\bar{R} + B^{\tau}S(k+1)B]^{-1}[-B^{\tau}S(k+1)\alpha(k) - B^{\tau}h(k+1) + \bar{\lambda}(k)] \qquad \textbf{(D,17)}$$

Finally, using (D,16), the state equation $x(k+1) = Ax(k) + Bu(k) + \alpha(k)$ can be written as

$$x(k+1) = (A - BG(t))x(k) + Bg(k) + \alpha(k) \qquad \textbf{(D,18)}$$

A straightforward reasoning on the dependence of the different variables involved, gives rise to Procedure 5.3.1.

E. DERIVATION OF PROCEDURE 5.3.2

Here we will include terminal constraints in the values of the state variables $x_i(N_f)=0$, $i \in [1,q]$, which can be written as follows

$$\psi(x(N_f)) = Cx(N_f) = 0 \qquad \textbf{(E,1)}$$

where $C = [I_q|0]$ is a qxn matrix.

Thus, we want to solve the following two-point boundary value problem (TPBVP)

$$
\begin{aligned}
x(k+1) &= Ax(k)+B\bar{R}^{-1}(B^{\top}p(k+1)-\bar{\lambda}(k)) + \alpha(k) \\
p(k) &= -\bar{Q}x(k) + A^{\top}p(k+1) - \bar{\beta}(k)
\end{aligned}
\tag{E,2}
$$

with border conditions:

$$
\begin{aligned}
x(N_0) &= x_0 \\
p(N_f) &= \Phi x(N_f) + C^{\top}\nu
\end{aligned}
\tag{E,3}
$$

where $\nu \in \Re^q$ is a Lagrange multiplier to be determined such that (E,1) is satisfied. The control law is given by:

$$
u(k) = -\bar{R}^{-1}(B^{\top}p(k+1) - \bar{\lambda}(k)) \tag{E,4}
$$

This linear TPBVP can be solved by using the backward sweep method (Bryson and Ho, 1975; Lewis, 1986a). The key is to assume the relationship between costate and state as

$$
p(k) = S(k)x(k)+h(k) + F(k)\nu \tag{E,5}
$$

and express the (assumed fixed) terminal constraint function as

$$
\psi = F(k)^{\top}x(k)+W(k)\nu+\eta(k) \tag{E,6}
$$

where $S(k)$ is a nxn matrix, $F(k)$ is a nxq matrix, $W(k)$ is a qxq matrix, $h(k) \in \Re^n$, and $\eta(k) \in \Re^q$.

Substituting (E,5) in the first equation in (E,2) we obtain:

$$
\begin{aligned}
x(k+1) = Ax(k)+B\bar{R}^{-1}(& B^{\top}(S(k+1)x(k+1) \\
& +h(k+1)+F(k)\nu)-\bar{\lambda}(k)) + \alpha(k)
\end{aligned}
\tag{E,7}
$$

Grouping terms results in the following:

$$
\begin{aligned}
x(k+1) = (I_n + B\bar{R}^{-1}B^{\top}S(k+1))^{-1}(Ax(k) & + B\bar{R}^{-1}\bar{\lambda}(k) \\
+ \alpha(k) - B\bar{R}^{-1}B^{\top}(h(k+1)+F(k+1)\nu)) &
\end{aligned}
\tag{E,8}
$$

Substituting (E,5) in the second equation in (E,2) we have

$$S(k)x(k) + h(k) + F(k)\nu = \bar{Q}x(k) + A^{\top}S(k+1)x(k+1) + A^{\top}h(k+1) + A^{\top}F(k+1)\nu - \bar{\beta}(k) \qquad \textbf{(E,9)}$$

Substituting (E,8) in (E,9) and grouping terms:

$$\begin{aligned} &[-S(k)+\bar{Q}+A^{\top}S(k+1)[I_n+B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}A\ ]x(k) \\ &+[A^{\top}-A^{\top}S(k+1)[I_n+B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}B\bar{R}^{-1}B^{\top}]h(k+1)-h(k)-\bar{\beta}(k) \\ &+A^{\top}S(k+1)[I_n+B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}[B\bar{R}^{-1}\bar{\lambda}(k)+\alpha(k)] \\ &[A^{\top}F(k+1)-F(k)-A^{\top}S(k+1)[I_n+B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}B\bar{R}^{-1}B^{\top}F(k+1)]\nu = 0 \end{aligned} \qquad \textbf{(E,10)}$$

Equating coefficients to zero in (E,10) results in the following set of difference equations which can be solved backwards from the terminal conditions $S(N_f) = \Phi, h(N_f) = 0$ and $F(N_f) = C^{\top}$.

$$S(k) = \bar{Q} + A^{\top}S(k+1)[I_n + B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}A \qquad \textbf{(E,11)}$$

$$\begin{aligned} h(k) = {} & [A^{\top}-A^{\top}S(k+1)[I_n + B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}B\bar{R}^{-1}B^{\top}]h(k+1) \\ & - \bar{\beta}(k) + A^{\top}S(k+1)[I_n+B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}[B\bar{R}^{-1}\lambda(k)+\alpha(k)] \end{aligned} \qquad \textbf{(E,12)}$$

$$F(k) = [A^{\top}-A^{\top}S(k+1)[I_n+B\bar{R}^{-1}B^{\top}S(k+1)]^{-1}B\bar{R}^{-1}B^{\top}]F(k+1) \qquad \textbf{(E,13)}$$

In addition, we have assumed that the constraint function (E,6) is constant and hence:

$$\psi = F(k)^{\tau}x(k) + W(k)\nu + \eta(k) = F(k+1)^{\tau}x(k+1) + W(k+1)\nu + \eta(k+1) \qquad \textbf{(E,14)}$$

Substitution of (E,8) in (E,14) and grouping terms, results in a repeat of equation (E,13) plus the following difference equations which may be solved backwards from the terminal conditions $W(N_f) = 0$ and $\eta(N_f) = 0$:

$$W(k) = W(k+1) - F(k+1)^{\tau}[I_n + B\bar{R}^{-1}B^{\tau}S(k+1)]^{-1}B\bar{R}^{-1}B^{\tau}F(k+1) \qquad \textbf{(E,15)}$$

$$\eta(k) = \eta(k+1) + F(k+1)^{\tau}[I_n + B\bar{R}^{-1}B^{\tau}S(k+1)]^{-1}[B\bar{R}^{-1}\bar{\lambda}(k) - B\bar{R}^{-1}B^{\tau}h(k+1) + \alpha(k)] \qquad \textbf{(E,16)}$$

We need to find an expression for the multiplier ν. Then from the terminal constraint function (E,6) we obtain:

$$\nu = W(k)^{-1}(F(k)^{\tau}x(k) - \eta(k)) \qquad \textbf{(E,17)}$$

which is valid for all $k \in [N_0, N_f]$. However, noticing that $W(N_f)$ is singular, we may write, for instance:

$$\nu = W(N_0)^{-1}(F(N_0)^{\tau}x_0 - \eta(N_0)) \qquad \textbf{(E,18)}$$

If $W(k)$ is singular for all $k \in [N_0, N_f]$, then the system is not controllable and the terminal constraint may not be satisfied (Bryson and Ho, 1975).

By using the matrix inversion lemma (See, for instance, Lewis (1986a)) we obtain the following equivalence:

$$[I_n + B\bar{R}^{-1}B^{\tau}S(k+1)]^{-1} = I_n - B[\bar{R} + B^{\tau}S(k+1)B]^{-1}B^{\tau}S(k+1) \qquad \textbf{(E,19)}$$

and define $G(k)$ as:

$$G(k) = [\bar{R} + B^{\top}S(k+1)B]^{-1}B^{\top}S(k+1)A \qquad \textbf{(E,20)}$$

Now, we can re-write (E,11) and (E,12) as follows:

$$S(k) = \bar{Q} + A^{\top}S(k+1)(A - BG(k)) \qquad \textbf{(E,21)}$$

$$\begin{aligned} h(k) = {} & (A - BG(k))^{\top}h(k+1) + (A - BG(k))^{\top}S(k+1)\alpha(k) \\ & - \bar{\beta}(k) + G(k)^{\top}\bar{\lambda}(k) \end{aligned} \qquad \textbf{(E,22)}$$

Substituting (E,5) in (E,4) we obtain:

$$u(k) = -\bar{R}^{-1}B^{\top}[S(k+1)x(k+1) + h(k+1) + F(k+1)\nu] + \bar{R}^{-1}\lambda(k) \qquad \textbf{(E,23)}$$

which can be re-written as follows:

$$u(k) = -G(k)x(k) + g(k) \qquad \textbf{(E,24)}$$

where

$$\begin{aligned} g(k) = {} & [\bar{R} + B^{\top}S(k+1)B]^{-1}[-B^{\top}S(k+1)\alpha(k) \\ & -B^{\top}F(k+1)\nu - B^{\top}h(k+1) + \bar{\lambda}(k)] \end{aligned} \qquad \textbf{(E,25)}$$

Finally, using (E,24), the state equation $x(k+1) = Ax(k) + Bu(k) + \alpha(k)$ can be written as

$$x(k+1) = (A - BG(t))x(k) + Bg(k) + \alpha(k) \qquad \textbf{(E,26)}$$

A straightforward reasoning on the dependence of the different variables involved, gives rise to Procedure 5.3.2.

F. DERIVATION OF PROCEDURE 6.2.1

Here we want to solve the following discrete-time two-point boundary value problem (TPBVP)

$$\begin{aligned} x(k+1) &= Ax(k)+B\bar{R}^{-1}(B^{\top}p(k+1)-\bar{\lambda}(k)) + \alpha(k) \\ p(k) &= -\bar{Q}x(k) + A^{\top}p(k+1) - \bar{\beta}(k) \end{aligned} \qquad \textbf{(F,1)}$$

with border conditions:

$$\begin{aligned} x(N_0) &= x_0 \\ p(N_f) &= C^{\top}\Phi Cx(N_f) - C^{\top}\Phi r_0(N_f) \end{aligned} \qquad \textbf{(F,2)}$$

where all the quantities are defined in Chapter 6 and the control law is given by:

$$u(k) = -\bar{R}^{-1}(B^{\top}p(k+1) - \bar{\lambda}(k)) \qquad \textbf{(F,3)}$$

This TPBVP is identical in structure to (D,1), with a different terminal condition for the costate. The solution may also be found by the sweep method (Bryson and Ho, 1975). The key is to assume the relationship between costate and state as

$$p(k) = S(k)x(k)+h(k) \qquad \textbf{(F,4)}$$

where $S(k)$ is a nxn matrix and $h(k)\in\Re^n$. The derivation is identical to that presented in Appendix D, the only difference being the terminal conditions in $S(k)$ and $h(k)$ which are as follows:

$$\begin{aligned} p(N_f) &= C^{\top}\Phi C \\ h(N_f) &= -C^{\top}\Phi r(N_f) \end{aligned}$$