CHAPMAN & HALL/CRC
Monographs and Surveys in
Pure and Applied Mathematics 110

ITERATIVE

DYNAMIC

PROGRAMMING

REIN LUUS

CHAPMAN & HALLKRC
Monographs and Surveys in Pure and Applied Mathematics

CHAPMAN & HALL/CRC
Monographs and Surveys in
Pure and Applied Mathematics 110

ITERATIVE DYNAMIC PROGRAMMING

REIN LUUS

CHAPMAN & HALL/CRC

Boca Raton London New York Washington, D.C.

Library of Congress Cataloging-in-Publication Data

Luus, Rein
Iterative dynamic programming / Rein Luus.
p. cm. -- (Monographs and surveys in pure and applied mathematics; 110)
Includes hibliographical references and index.
ISBN 1-58488-148-8 (alk. paper)
1. Dynamic programming. 2. Iterative methods (Mathematics) 1. Title. 11. Chapman & HalVCRC monographs and surveys in pure and applied mathematics; 110.

QA402.5 .L88 ZOOO
5 19.7'03-dc2 1 99-058886
CIP

International Standard Book Number 1-58488-148-8
Library of Congress Card Number 99-058886
Printed in the United States of America 1 2 3 4 5 6 7 8 9 0
Printed on acid-free paper

This book is dedicated to
Professor Rutherford Aris

Contents

About the author

Rein Luus received his B.A.Sc. degree in Engineering Physics in 1961 and M.A.Sc. in Chemical Engineering in 1962 from the University of Toronto, and an A.M. degree in 1963 and Ph.D. degree in 1964 from Princeton University. In 1964, he was granted a Sloan Postdoctoral Fellowship, and during his postdoctorate studies at Princeton University, he wrote, with Professor Leon Lapidus, the book *Optimal Control of Engineering Processes*. In 1965, he joined the University of Toronto where he is currently Professor of Chemical Engineering.

Professor Luus has published more than 100 papers in scientific journals. A large number of these papers deal with his recent developments in iterative dynamic programming. He has served as a consultant for Shell Canada, Imperial Oil, Canadian General Electric, Fiberglas Ltd., and Milltronics. He spent a sabbatical year in the research department at Steel Company of Canada, doing mathematical modelling, simulation, and data analysis. In 1976, he was awarded the Steacie Prize, and in 1980 the ERCO award. He has devoted more than 38 years to his profession as a researcher and teacher.

Preface

Dynamic programming, developed by Richard Bellmann, is a powerful method for solving optimization problems. It has the attractive feature of breaking up a complex optimization problem into a number of simpler problems. The solution of the simpler problems then leads to the solution of the original problem. Such stage-by-stage calculations are ideally suited for digital computers, and the global optimum is always obtained. The drawbacks consisting of the *curse of dimensionality* and *menace of the expanding grid*, coupled with interpolation problems, have limited dynamic programming to solving optimal control problems of very low dimension.

To overcome these limitations of dynamic programming, I suggested ten years ago to use dynamic programming in an iterative fashion, where the interpolation problem is eliminated by using the control policy that was optimal for the grid point closest to the state, and by clustering the grid points closer together around the best value in an iterative fashion. Such a scheme, however, was computationally not feasible, since a two-dimensional optimal control problem with a scalar control took over an hour to solve on the Cray supercomputer. However, a slight change made the computational procedure feasible. Instead of picking the grid points over a rectangular array, I generated the grid points by integrating the state equations with different values of control. For that two-dimensional optimal control problem the computational effort was reduced by a factor of 100, and the dimensionality of the state vector no longer mattered. This led to what now is termed *iterative dynamic programming*. In iterative fashion, dynamic programming can now be used with very high-dimensional optimal control problems. The goal of this book is to give a working knowledge of iterative dynamic programming (IDP), by providing worked out solutions for a wide range of problems.

A strong background in mathematical techniques and chemical engineering is not essential for understanding this book, which is aimed at the level of seniors or first-year graduate students. Although many of the examples are from chemical engineering, these examples are presented with sufficient background material to make them generally understandable, so that the optimal control problems will be meaningful.

In Chapter 1, the basic concepts involving mathematical models and solution of sets of nonlinear algebraic equations are presented. In Chapter 2, two steady-state optimization procedures that I have found very useful and which provide the necessary links to ideas pertaining to iterative dynamic programming are presented and illustrated. In Chapter 3, application of dynamic programming is illustrated with several examples to give the reader some appreciation of its attractive features. In Chapter 4, I present the basic ideas underlying iterative dynamic programming.

In Chapter 5, different ways of generating allowable values for control are examined. In Chapter 6, I examine in a preliminary fashion the effects of the parameters involved in IDP. Such evaluation of the parameters is continued throughout the book. In Chapter 7, it is shown that the use of piecewise linear continuous control leads to

great advantages when the control policy is smooth. Comparison of IDP with solution of the Riccati equation for a quadratic performance index shows the advantages of IDP. In Chapter 8, it will become obvious to the reader that the optimal control of time-delay systems presents no real difficulties. In Chapter 9, the use of variable stage lengths in optimal control problems is introduced to enable accurate switching. In Chapter 10, I consider the optimal control of singular control problems that are very difficult to solve by other methods. In Chapter 11, the application of penalty functions is illustrated for the optimal control of systems where there are state constraints present. The time optimal control problem is considered in Chapter 12, and, in Chapter 13, the optimal control of nonseparable problems is illustrated with two examples. Since sensitivity is such an important issue, I have discussed that aspect in some detail in Chapter 14. In Chapter 15, I consider some practical aspects of applying optimal control to physical systems in practice and outline some areas for further research.

To enable the reader to gain direct experience with the computations, I have given listings of typical computer programs in their entirety in the appendix. The computer programs make the logic discussed in the text easier to follow, and the programs may be used by the reader to actually run some cases. It is through this type of direct experience that one gains the most insight into the computational aspects. Throughout the book I have also given computation times for some runs to give the reader some idea of what to expect. Whether a particular problem takes a few seconds or a few hours to run is useful information for the user. I have not made any special effort to maximize the efficiency of the computer programs. This exercise is left for the reader.

I am grateful to Professor Rutherford Aris for suggesting that I write this book and for providing encouragement during the writing process. I am also grateful to Professor Árpad Pethö for organizing the annual workshops in Germany and Hungary to which he has invited me to present the continuing developments of IDP. My thanks also go to the Natural Sciences and Engineering Council of Canada for supporting some of this work.

Rein Luus

Notation

a_{ij}	element of the i^{th} row and j^{th} column of **A** matrix
A	state coefficient matrix $(n \times n)$
B	control coefficient matrix $(n \times m)$
c	constant
c_i	cost associated with i^{th} job
D	diagonal matrix of random numbers between -1 and 1
f	general function
f_i	i^{th} element of the vector f
f	general vector function
g_i	continuous function of state variables introduced for convenience
h	height
h_i	i^{th} equality constraint
H	Hamiltonian
I	performance index
I	identity matrix
J	augmented performance index
J_i	i^{th} job
L	length of a time stage
m	number of control variables
M	number of allowable values for each control variable chosen from uniform grid
n	number of state variables
N	number of grid points
P	number of time stages
q	pass number; raffinate solvent flow rate
Q	sum of squares of deviation
Q	weighting matrix $(n \times n)$
r	region vector over which allowable values of variables are chosen
R	number of randomly chosen values for control
R	weighting matrix $(m \times m)$
s	shifting term
s_i	shifting term corresponding to constraint i
S	sum of absolute values
t	time
t_f	final time of operation
u	scalar control
u_j	j^{th} element of control vector **u**
u	control vector $(m \times 1)$
v	variable stage length; velocity

x_i i^{th} state variable
$\mathbf{x}$ state vector $(n \times 1)$
z_i i^{th} adjoint variable
$\mathbf{z}$ adjoint vector $(n \times 1)$

Greek letters

α operator; positive constant
α_j lower bound on control variable u_j
β constant
β_j upper bound on the control variable u_j
γ region contraction factor by which the region is reduced after every iteration
δ a small perturbation
ϵ tolerance
η region restoration factor
θ penalty function factor
$\mathbf{\Theta}$ matrix $(n \times n)$
ρ penalty function factor
τ delay time
τ_i time to execute job i
ϕ integrand of performance index
Φ final value performance index
$\mathbf{\Phi}$ transition matrix
$\mathbf{\Psi}$ matrix $(n \times m)$

Subscripts

$_f$ final time
$_{fc}$ calculated final time
$_i$ index
$_{in}$ initial value
$_j$ index
$_k$ index
$_{new}$ new value
$_{old}$ previous value
$_p$ predicted

Superscripts

* best value obtained from previous iteration
^{d} desired value
^{j} iteration step
^{0} optimal value
$^{(0)}$ initial value
^{q} pass number
^{T} transpose

Chapter 1

Fundamental concepts

1.1 Introduction

Optimization, or optimal control, in the sense to be used in this book, is concerned with determining the largest value or the smallest value for some criterion of performance. For example, if we are dealing with economic benefit, then we would like to choose the conditions for operating the system so that the economic benefit would be maximized. If, however, the criterion of performance is chosen to be the cost, then the system should be operated to minimize the cost. In each case we seek the operating conditions that yield the extreme value for the performance criterion.

It is obvious that the operating procedure is dictated by the choice of the criterion of operation. The choice of such criterion is not straightforward, since there are numerous factors that must be taken into consideration, such as productivity, profit, cost, environmental impact, reliability, yield of a reactor, quality of product, etc. We may want to have more than one criterion for optimization. For the present work, however, we assume that all the objectives can be expressed in terms of an appropriate scalar criterion of performance which we call *performance index*, with the understanding that the optimization results will be dependent on such a choice. It is also important to express this performance index in terms of the same variables that are used in the mathematical model of the physical system or process under consideration.

For the development of the *mathematical model* of the system, we need some insight into the behavior of the physical system, and how the variables at our disposal may be used to change its behavior. Such a relationship may be expressed in terms of algebraic equations, ordinary differential equations, difference equations, partial differential equations, integral equations, or combinations of them. The simplest situation arises, of course, if the model is described in terms of algebraic equations only. In this case we have a *steady-state optimization* problem, or we may simply call the process of finding the extreme value of the performance index *optimization*. If, however, the model consists of differential equations, difference equations, or integral

equations, the optimization is carried out over a *trajectory* and we call such a problem an *optimal control* problem. The mathematical procedure to be used for optimization is primarily dependent on the structure of the mathematical model. Since there is usually a considerable amount of freedom in the choice of a model, one usually tries to choose the simplest model that is suitable. The suitability of the model is usually established from the interpretation of the results obtained from the optimization.

In this book we are concerned mainly with optimal control problems. However, to gain insight into complex problems, it is useful to understand some fundamental ideas used in solving sets of algebraic equations and in solving steady-state optimization problems. Therefore, in the present chapter we examine the solution of sets of algebraic equations and in Chapter 2 we consider some aspects of steady-state optimization.

One of the most difficult problems in optimization or optimal control is establishing with some element of certainty that the global optimum has been obtained. The difficulty arises when the equations describing the system are highly nonlinear and a large number of local optima are present. Therefore, it is necessary to cross-check the results, even when one is quite certain that the global optimum has been obtained. If such cross-checking is done by a totally different optimization procedure, then the confidence in accepting the result is increased. The main goal of this book is to present a computational procedure called *iterative dynamic programming* and to show how it may be used to solve a variety of optimal control problems.

1.2 Fundamental definitions and notation

To present a unified picture throughout this book, we will present certain definitions and consistent notations. For purposes of clarity, it is worthwhile to briefly outline some of these items.

1.2.1 Operator

An operator α is a symbol for some mathematical procedure which changes one function into another function. For example let us consider different operators operating on the continuous function $f(x) = x^3 + \sin x$:

Operator operating on $f(x) = x^3 + \sin x$	Resulting function
c = const	$cx^3 + c\sin x$
$D = d/dx$	$3x^2 + \cos x$
square root	$(x^3 + \sin x)^{1/2}$

The operator α is *linear* if

$$\alpha[f(x) + g(x)] = \alpha f(x) + \alpha g(x)$$

and

$$\alpha[cf(x)] = c\alpha[f(x)],$$

where c is a constant. Thus the constant c and D operators above are linear. By contrast, an operator that is not linear is called *nonlinear*, an example being the square root operator.

If a relationship of the type

$$\alpha f(x) = cf(x)$$

exists, where c is a constant, then c is said to be an *eigenvalue* of the operator α. For example,

$$\frac{d}{dx}e^{3x} = 3e^{3x}, \tag{1.1}$$

so 3 is an eigenvalue of the operator d/dx and e^{3x} is the corresponding *eigenfunction*.

Another example of an eigenvalue relationship is

$$\frac{d^2}{dt^2}\sin at = -a^2\sin at, \tag{1.2}$$

and we see that $-a^2$ is an eigenvalue of the operator d^2/dt^2 with a corresponding eigenfunction $\sin at$. Note that the same relationship holds for $\cos at$, so the eigenfunction is not necessarily unique.

Also eigenvalues corresponding to an operator may not be unique. For example, when we write the energy balance for the hydrogen atom in the form

$$H\psi = E\psi, \tag{1.3}$$

where the Hamiltonian operator H expresses the sum of kinetic and potential energies and E denotes the total energy, we find numerous values for E for which the equation holds. These values of E, which are eigenvalues of the operator H, are frequently called "energy levels".

We will be encountering eigenvalues when dealing with matrices. In that case the operator is a matrix, operating on a vector to give a constant times the vector. Then the constant is said to be an eigenvalue of the matrix and the vector is called the *eigenvector* corresponding to that particular eigenvalue.

1.2.2 Vectors and matrices

Since we shall be dealing with systems of many dimensions in many variables, the convenient shorthand notation consistent with vector-matrix terminology will be used. We shall use small boldface letters to denote vectors and capital boldface letters to denote matrices. Thus $\mathbf{x}$ will denote a vector and $\mathbf{A}$ a matrix. All vectors are taken as

column vectors. To obtain a *row* vector, we take the *transpose* of the vector denoted by the superscript ^{T}. Similarly, the transpose of a matrix results in having the rows and columns interchanged. The designation $(m \times n)$ for either a vector or a matrix will mean m rows and n columns:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix}, \qquad \mathbf{x}^T = [x_1 \quad x_2 \quad ... \quad x_n].$$

$$(n \times 1) \qquad\qquad (1 \times n)$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \qquad \mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \cdots & \cdots & \cdots & \cdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}.$$

$$(m \times n) \qquad\qquad (n \times m)$$

It is relatively easy to verify that

$$(\mathbf{Ax})^T = \mathbf{x}^T\mathbf{A}^T. \tag{1.4}$$

When a square matrix (one in which $m = n$) equals its transpose,

$$\mathbf{A} = \mathbf{A}^T \tag{1.5}$$

the matrix $\mathbf{A}$ is called *symmetric*. Symmetric matrices have some desirable properties which help in calculations. If the columns of a square matrix $\mathbf{A}$ are *linearly independent*, inverse of the matrix denoted by $\mathbf{A}^{-1}$ exists, having the property

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}, \tag{1.6}$$

where $\mathbf{I}$ is the *identity* matrix, which is a diagonal matrix with 1's along the diagonal. Therefore, the identity matrix has the property

$$\mathbf{AI} = \mathbf{IA} = \mathbf{A}. \tag{1.7}$$

The convenience of using vector-matrix notation is realized when we consider the quadratic form

$$\mathbf{x}^T\mathbf{Ax} = \sum_{j=1}^{n}\sum_{i=1}^{n} x_i a_{ij} x_j, \tag{1.8}$$

where a_{ij} are the elements of $\mathbf{A}$.

Frequently, a situation arises where all the elements of a vector are equal to the same numerical value, let us say 2. Then, instead of writing $r_i = 2, i = 1, 2, \cdots, n$, or $\mathbf{r} = [\ 2\ 2\ \cdots\ 2\ ]^T$, we will use the shortform $\mathbf{r} = \mathbf{2}$. Sometimes, however, we wish to emphasize some entries in a table of results, by using boldface to make them stand out. No confusion with respect to this notation is expected.

1.2.3 Differentiation of a vector

Given a vector

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ . \\ f_n \end{bmatrix},$$

we may differentiate this vector with respect to $\mathbf{x}$ by first taking the transpose and then simply using matrix algebra, i.e.,

$$\frac{\partial \mathbf{f}^{\,T}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ . \\ . \\ . \\ \frac{\partial}{\partial x_n} \end{bmatrix} [f_1 \quad f_2 \quad \cdots \quad f_n] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_2} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}.$$

The second derivative of a continuous *scalar* function $f(\mathbf{x})$ with respect to $\mathbf{x}$ is

$$\frac{\partial^2 f}{\partial \mathbf{x}^2} = \frac{\partial}{\partial \mathbf{x}}((\frac{\partial f}{\partial \mathbf{x}})^T) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

It is noted that this $(n \times n)$ matrix is symmetric, since the order of differentiation does not matter for a continuous function.

1.2.4 Taylor series expansion

It is convenient at times to expand a vector function $\mathbf{f}(\mathbf{x})$ in terms of a small perturbation $\delta\mathbf{x}$. Such an approximation can be represented by

$$\mathbf{f}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{f}(\mathbf{x}) + (\frac{\partial \mathbf{f}^T}{\partial \mathbf{x}})^T \delta\mathbf{x} + \cdots \tag{1.9}$$

and a scalar continuous function $f(\mathbf{x})$ can be conveniently expanded as

$$f(\mathbf{x} + \delta\mathbf{x}) = f(\mathbf{x}) + (\frac{\partial f}{\partial \mathbf{x}})^T \delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^T \frac{\partial^2 f}{\partial \mathbf{x}^2}\delta\mathbf{x} + \cdots \tag{1.10}$$

1.2.5 Norm of a vector

The norm of a vector $\mathbf{x}$ denoted by $\|\mathbf{x}\|$ may be thought of as the length of the vector. It is a scalar function which assigns to every vector $\mathbf{x}$ in the vector space a real number denoted by $\|\mathbf{x}\|$ such that the following four properties hold:

(1)	$\|\mathbf{x}\| > 0$	for all	$\mathbf{x} \neq \mathbf{0}$
(2)	$\|\mathbf{x}\| = 0$	if and only if	$\mathbf{x} = \mathbf{0}$
(3)	$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$	for all	$\mathbf{x}$ and $\mathbf{y}$
(4)	$\|\lambda\mathbf{x}\| = \|\lambda\|\|\mathbf{x}\|$	for all	$\mathbf{x}$ and λ

where λ is a real scalar constant.

Although the best-known norm is the Euclidean measure of length

$$\|\mathbf{x}\|_2 = [\mathbf{x}^T\mathbf{x}]^{\frac{1}{2}} = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}}, \tag{1.11}$$

in this book we wish to use also the sum of the absolute values norm

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| = S. \tag{1.12}$$

It is relatively easy to show that both of these expressions satisfy the four properties of the norm. They are, in fact, special cases of the general q-norm defined by

$$\|\mathbf{x}\|_q = (\sum_{i=1}^{n} |x_i|^q)^{\frac{1}{q}}. \tag{1.13}$$

We see that when we take $q = \infty$, the ∞-norm is simply the maximum absolute value of the element amongst all the elements in $\mathbf{x}$.

1.2.6 Sign definite

The scalar function of a vector $V(\mathbf{x})$ is said to be *positive definite* if $V(\mathbf{x}) > 0$ for $\mathbf{x} \neq \mathbf{0}$ and $V(\mathbf{x}) = 0$ for $\mathbf{x} = \mathbf{0}$. The negative of a positive definite function is called *negative definite*. If $V(\mathbf{x}) \geq 0$ for $\mathbf{x} \neq \mathbf{0}$ and $V(\mathbf{x}) = 0$ for $\mathbf{x} = \mathbf{0}$ then $V(\mathbf{x})$ is said to be *positive semi-definite*.

The quadratic form $V(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x}$, where $\mathbf{Q}$ is symmetric, is positive definite if all the eigenvalues of $\mathbf{Q}$ are positive. Then we say that the matrix $\mathbf{Q}$ is positive definite. A beautiful feature about a positive definite matrix is the existence of inverse.

1.2.7 Stationary and maxima (minima) points

If $\partial f(\mathbf{x})/\partial \mathbf{x} = \mathbf{0}$ at some value $\mathbf{x} = \mathbf{x}_0$, then $\mathbf{x}_0$ is said to be a *stationary* point. However, this is not necessarily the maximum or a minimum of the function $f(\mathbf{x})$. If $\frac{\partial^2 f}{\partial \mathbf{x}^2}$ is negative definite, then $\mathbf{x}_0$ is a *local* maximum and if $\frac{\partial^2 f}{\partial \mathbf{x}^2}$ is positive definite, then $\mathbf{x}_0$ is a *local* minimum. The point $\mathbf{x}_0$ is an *absolute* or *global* maximum if

$$f(\mathbf{x}) \leq f(\mathbf{x}_0)$$

and a global minimum if the inequality sign is reversed. For an unconstrained continuous function with continuous first derivative, the global maximum or minimum may be obtained by examining all the stationary points of the function and comparing the values of the function at those stationary points.

1.3 Steady-state system model

For a steady-state model we assume the system under consideration can be represented in terms of the variables $x_1, x_2, \cdots, x_n$. The model is given by a set of equality constraints

$$\phi_i(x_1, x_2, ..., x_n) = 0, \qquad i = 1, 2, ..., m \tag{1.14}$$

where $m \leq n$, and a set of inequality constraints

$$g_j(x_1, x_2, ..., x_n) \leq 0, \qquad j = 1, 2, ..., s. \tag{1.15}$$

1.4 Continuous-time system model

In the major portion of this book, we shall assume that the physical system is described mathematically by a set of deterministic lumped-parameter ordinary differential equations, where the initial conditions are given. There are two types of variables under consideration. The state variables, denoted by x_i, represent the state of the system. The other type of variables, denoted by u_j, represent the control variables, or inputs into the system. The model showing the effect of the inputs on the state of the system is assumed to have the general form

$$\frac{dx_1}{dt} = f_1(x_1, x_2, ..., x_n, u_1, u_2, ..., u_m, t) \tag{1.16}$$

$$\frac{dx_2}{dt} = f_2(x_1, x_2, ..., x_n, u_1, u_2, ..., u_m, t) \tag{1.17}$$

$$\cdots \qquad \cdots \quad \cdots \quad \cdots$$

$$\frac{dx_n}{dt} = f_n(x_1, x_2, ..., x_n, u_1, u_2, ..., u_m, t) \tag{1.18}$$

or, in vector form, the system can be described more compactly by

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{1.19}$$

with the initial state specified

$$\mathbf{x}(0) = \mathbf{c}. \tag{1.20}$$

Here $\mathbf{x}$ is an $(n \times 1)$ state vector and $\mathbf{u}$ is an $(m \times 1)$ control vector.

We can remove the explicit time dependence on the right hand side of Eq. (1.19) by introducing an additional state variable x_{n+1} by the differential equation

$$\frac{dx_{n+1}}{dt} = 1, \quad x_{n+1}(0) = 0 \tag{1.21}$$

and incorporating x_{n+1} into the state vector $\mathbf{x}$. Then we have a simpler form

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{1.22}$$

with

$$\mathbf{x}^T(0) = [\mathbf{c}^T \quad 0]. \tag{1.23}$$

Therefore, the number of state variables describing a particular system is not unique, since auxiliary state variables may be introduced for convenience to describe the same system.

1.5 Discrete-time system model

We assume that the discrete-time system model of a physical system can be written in the form

$$\mathbf{x}(k+1) = \mathbf{h}(\mathbf{x}(k), \mathbf{u}(k)) \tag{1.24}$$

with the given initial state

$$\mathbf{x}(0) = \mathbf{c}. \tag{1.25}$$

Such a difference equation can arise if the system consists of a number of stages or units, as in an extraction train in chemical engineering. We can also obtain a difference equation by integrating the differential equation repeatedly over a sampling period. For example, suppose we have a linear continuous-time system

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{1.26}$$

and suppose $\mathbf{u}$ is kept constant at its value $\mathbf{u}(0)$ over the entire time interval from 0 to T. Then integration of Eq. (1.26) gives

$$\mathbf{x}(T) = \mathbf{\Phi}(T)\mathbf{c} + \int_0^T \mathbf{\Phi}(t-\lambda)\mathbf{B}\mathbf{u}(0)d\lambda \tag{1.27}$$

where the transition matrix $\mathbf{\Phi}(T)$ is given by

$$\mathbf{\Phi}(T) = \exp(\mathbf{A}T) = \sum_{i=0}^{\infty} \frac{(\mathbf{A}T)^i}{i!}. \tag{1.28}$$

Eq. (1.27) can be written as

$$\mathbf{x}(T) = \mathbf{\Phi}(T)\mathbf{c} + \mathbf{\Psi}(T)\mathbf{u}(0) \tag{1.29}$$

where we have introduced the $(n \times m)$ matrix

$$\mathbf{\Psi}(T) = \int_0^T \exp[\mathbf{A}(T-\lambda)]d\lambda\mathbf{B} = T\sum_{i=0}^{\infty} \frac{(\mathbf{A}T)^i}{(i+1)!}\mathbf{B}. \tag{1.30}$$

Let us now repeat the procedure but integrate from $t = T$ to $t = 2T$ using a constant value of $\mathbf{u}(T)$. Since we are integrating again over a time interval of the same length as before, we get

$$\mathbf{x}(2T) = \mathbf{\Phi}(T)\mathbf{x}(T) + \mathbf{\Psi}(T)\mathbf{u}(T). \tag{1.31}$$

Continuing the procedure, we obtain

$$\mathbf{x}[(k+1)T] = \mathbf{\Phi}(T)\mathbf{x}(kT) + \mathbf{\Psi}(T)\mathbf{u}(kT), \quad k = 0, 1, 2, \cdots. \tag{1.32}$$

The sampling time T is usually dropped to give a simpler looking equation

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Psi}\mathbf{u}(k), \tag{1.33}$$

which is a special case of a linear difference equation. If the series expansion for $\mathbf{\Phi}$ is asymptotic, then the most convenient way of getting the matrices is to take a truncated series of $N + 1$ terms:

$$\mathbf{\Theta} = \sum_{i=0}^{N} \frac{(\mathbf{A}T)^i}{(i+1)!}. \tag{1.34}$$

Then

$$\mathbf{\Phi} = \mathbf{I} + \mathbf{\Theta}\mathbf{A}T \tag{1.35}$$

and

$$\mathbf{\Psi} = \mathbf{\Theta}\mathbf{B}T. \tag{1.36}$$

Such a procedure with $N = 20$ was used by Luus and Smith [23] to convert a differential equation describing a gas absorber into a difference equation. If the series is not asymptotic, we can always use the identity

$$\exp(\mathbf{A}T) = [\exp(\frac{\mathbf{A}T}{\beta})]^{\beta} \tag{1.37}$$

and choose the constant β sufficiently large to make the series asymptotic.

1.6 The performance index

In most of the book we denote the performance index by I. There are times, however, when an augmented form of the performance index is used, and a different letter such as J may be used, or when a profit function is implied, the letter P is sometimes used.

When the model describing the system is a set of algebraic equations, the performance index is an algebraic expression in the form

$$I = f(x_1, x_2, ..., x_n). \tag{1.38}$$

When the system is described by differential equations or difference equations, there are different forms that the performance index can take. For example, we may have the performance index

$$I = \int_0^{t_f} \phi(\mathbf{x}, \mathbf{u}) dt, \tag{1.39}$$

where t_f denotes final time, or we may have a performance index of the form

$$I = \mathbf{k}^T \mathbf{x}(t_f) = \sum_{i=1}^{n} k_i x_i(t_f). \tag{1.40}$$

Both of these are equivalent, as we will now show. Suppose we define a new state variable x_{n+1} by the equation

$$\frac{dx_{n+1}}{dt} = \phi(\mathbf{x}, \mathbf{u}), \quad x_{n+1}(0) = 0. \tag{1.41}$$

Then the performance index in Eq. (1.39) is equivalent to

$$I = x_{n+1}(t_f). \tag{1.42}$$

This is a special form of Eq. (1.40). Therefore, for most of our work, we choose the performance index as an explicit function of the state at the final time in the form

$$I = \Phi(\mathbf{x}(t_f)), \tag{1.43}$$

with the understanding that we have the freedom to choose the state variables accordingly. There are situations, however, when the performance index can not be expressed as a function of the final state. Then we have a *nonseparable* performance index that requires special attention. That will be illustrated and discussed in Chapter 13.

1.7 Interpretation of results

An important part of optimization is the interpretation of the results. It is realized that the optimal solution is dependent on the mathematical model and the performance index. Therefore, the results of the optimization must be considered with care. If the mathematical model is an inadequate representation of the physical system, the results are not optimal with respect to the application that has been intended. In the same way, the results are dependent on the choice of the performance index. The choice of the final time t_f has a substantial effect on the optimal control policy [14], but it is frequently chosen quite arbitrarily. Only thorough familiarity with the physical system will ensure successful application of optimization.

Another important aspect of optimization is to determine whether a local optimum or a global optimum has been obtained. In a highly nonlinear problem this can be quite a challenge. The best way to assure oneself that a global optimum has been obtained is to cross-check the optimization by using a different optimization procedure. Simply taking different starting conditions may not yield the global optimum if there is a very large number of local optima. This was recently pointed out in [19] in the optimization of a bifunctional catalyst blend along a tubular reactor to maximize the yield of a desired product. Although 100 different starting conditions chosen at random were used with sequential quadratic programming (SQP), the global optimum was not obtained with that well-established optimization procedure. With iterative dynamic programming, however, the global optimum was readily obtained by Luus and Bojkov [17].

When numerical methods are used for optimization of nonlinear systems, some form of iteration is usually used. There is therefore the additional feature that must be considered, namely the convergence of the iterative procedure itself. The iterations may be ended prematurely, or the iterative procedure itself may fail to converge because of the nature of the problem or the parameters chosen. The problem of convergence is very challenging when a large change in the control policy has a very small effect on the performance index, and one is tempted to accept a highly fluctuating control policy as being optimal. Establishing the optimal control policy for low sensitivity problems is difficult [15]. These are some of the aspects of optimization that we wish to explore in later chapters.

1.8 Examples of systems for optimal control

1.8.1 Linear gas absorber

A linear system that has been widely used for optimal control studies is the six-plate gas absorber that was used for optimal control studies by Lapidus and Luus [11]. The model was used for time-optimal control studies by Bashein [2] and for time sub-optimal control by Bennett and Luus [3]. More recently it has been used for

investigation of the viability of iterative dynamic programming by Luus and Smith [23] and for time-optimal control studies by Bojkov and Luus [4]. Here, based on that six-plate gas absorber model, we consider a more general case of an n-plate gas absorber controlled by inlet feed stream concentrations (see Figure 1.1). A material

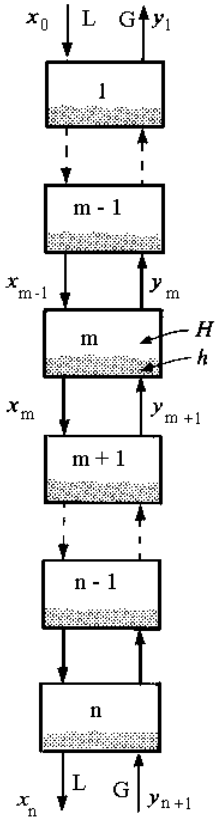

Figure 1.1: Gas absorber system consisting of n equilibrium stages

balance around the mth plate of the absorption tower yields

$$H\frac{dy_m}{dt} + h\frac{dx_m}{dt} = L(x_{m-1} - x_m) + G(y_{m+1} - y_m), \quad m = 1, 2, ..., n, \tag{1.44}$$

where

x_m, y_m = composition of liquid and vapor leaving the mth plate (kg solute/kg inert),
h, H = inert liquid and vapor hold-ups on each plate (assumed constant) (kg),
L = flow rate of inert liquid absorbent (kg/min),
G = flow rate of inert gas stream (kg/min),
t = time of operation (min).

Expressing the flow rates in terms of inerts ensures that L and G remain constant from stage to stage, even though the physical flow rates change because of the absorption

process. If a linear relationship is assumed between the compositions in the liquid and vapor in every stage

$$y_m = ax_m, \tag{1.45}$$

the mass balance equation can be converted to

$$\frac{dx_m}{dt} = \frac{d}{e}x_{m-1} - (\frac{d+1}{e})x_m + \frac{1}{e}x_{m+1}, \quad m = 1, 2, \cdots, n \tag{1.46}$$

where $d = L/Ga$ and $e = (Ha + h)/Ga$. The use of the convenient set of parameters: $a = 0.72, L = 40.8$ kg/min, $G = 66.7$ kg/min, $H = 1.0$ kg, and $h = 75$ kg yields the system equations in vector-matrix form

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{1.47}$$

where $\mathbf{A}$ is a tridiagonal coefficient matrix given by

$$\mathbf{A} = \text{tridiag}[0.538998 \quad -1.173113 \quad 0.634115] \tag{1.48}$$

and the $(n \times 2)$ control coefficient matrix $\mathbf{B}$ is given by

$$\mathbf{B}^T = \begin{bmatrix} 0.538998 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0.634115 \end{bmatrix}. \tag{1.49}$$

The initial condition is chosen as

$$x_i(0) = -0.0307 - (\frac{i-1}{n-1})(0.1273 - 0.0307), \quad i = 1, 2, \cdots, n. \tag{1.50}$$

The performance index to be minimized is

$$I = \int_0^{t_f} (\mathbf{x}^T\mathbf{x} + \mathbf{u}^T\mathbf{u})dt. \tag{1.51}$$

Two values for the final time have been used recently [16], namely $t_f = 10$ min and $t_f = 50$ min. Also, a range for the number of stages from $n = 10$ to $n = 1000$ has been used to examine the increase of computational effort of iterative dynamic programming as the dimensionality of the problem increases.

1.8.2 Nonlinear continuous stirred tank reactor

Let us consider the system consisting of a first-order irreversible chemical reaction carried out in a continuous stirred tank reactor (CSTR) as first formulated by Aris and Amundson [1] and used for control studies by Lapidus and Luus [11]. Control of the reactor is achieved by manipulation of the flow-rate of the cooling fluid through

a coil inserted in the reactor. The heat and mass balances for the reaction A $\rightarrow$ B in such a CSTR are given by

$$Vc_P\rho\frac{dT_R}{d\theta} = qc_P\rho(T_0 - T_R) - VU^* + (-\Delta H)VR \tag{1.52}$$

$$V\frac{dc_A}{d\theta} = q(c_{A0} - c_A) - VR \tag{1.53}$$

where

c_A = concentration of species A leaving reactor
c_{A0} = concentration of species A in feed stream
T_R = temperature of reactants leaving reactor
T_0 = temperature of feed stream to reactor
θ = time
V = volume of reactor
q = volumetric flow rate of feed and output streams
R = reaction rate term
$-\Delta H$ = heat of reaction
c_P = heat capacity
ρ = density
U^* = heat transfer term corresponding to cooling coil in reactor

As shown by Lapidus and Luus [11], these equations may be put into dimensionless form and, through a choice of a convenient set of parameters, transformed into the equations

$$\frac{dx_1}{dt} = -(2+u)(x_1 + 0.25) + (x_2 + 0.5)\exp[\frac{25x_1}{x_1+2}] \tag{1.54}$$

$$\frac{dx_2}{dt} = 0.5 - x_2 - (x_2 + 0.5)\exp[\frac{25x_1}{x_1+2}] \tag{1.55}$$

where x_1 represents deviation from dimensionless steady-state temperature and x_2 represents deviation from dimensionless steady-state concentration. In the present work, the initial conditions $x_1(0) = 0.09$ and $x_2(0) = 0.09$ are used. We consider the case where the control u is unbounded. The optimal control problem is to determine u in the time interval $0 \leq t < t_f$ that will minimize the performance index

$$I = \int_0^{t_f} (x_1^2 + x_2^2 + 0.1u^2)dt \tag{1.56}$$

where the dimensionless final time t_f is specified as 0.78. As was shown by Luus and Cormack [18], this system exhibits a local minimum $I = 0.24425$ and a global minimum $I = 0.133094$ when Pontryagin's maximum principle is used, and the resulting boundary value problem is solved by control vector iteration. Therefore, this system is an interesting test problem for optimal control procedures.

1.8.3 Photochemical reaction in CSTR

A more challenging problem of higher dimension consists of a photochemical reaction in an isothermal CSTR, as formulated by Jensen [10] and used for optimal control studies in [11] and [25]. It served as an ideal system for testing iterative dynamic programming for moderate dimensional nonlinear optimal control problems [13]. The mechanism for the reactions among the species A,B,C,D,E,F, and G is given by

$$\begin{aligned} A + B &\xrightarrow{k_1} 2D \\ C + B &\xrightarrow{k_2} CB \\ CB + B &\xrightarrow{k^*} 3E \\ E + D &\xrightarrow{k_3} 2F \\ A + F &\xrightarrow{k_4} 2G \end{aligned}$$

where the fifth reaction is photochemical and its rate is proportional to the square root of the light intensity. The intermediate compound CB is postulated to be present in immeasurably small quantities, so it satisfies the stationary condition, namely $d(CB)/dt = 0$. By using a convenient set of parameters, Lapidus and Luus [11] showed that the system is described by the eight differential equations

$$\frac{dx_1}{dt} = u_4 - qx_1 - 17.6x_1x_2 - 23x_1x_6u_3 \tag{1.57}$$

$$\frac{dx_2}{dt} = u_1 - qx_2 - 17.6x_1x_2 - 146x_2x_3 \tag{1.58}$$

$$\frac{dx_3}{dt} = u_2 - qx_3 - 73x_2x_3 \tag{1.59}$$

$$\frac{dx_4}{dt} = -qx_4 + 35.2x_1x_2 - 51.3x_4x_5 \tag{1.60}$$

$$\frac{dx_5}{dt} = -qx_5 + 219x_2x_3 - 51.3x_4x_5 \tag{1.61}$$

$$\frac{dx_6}{dt} = -qx_6 + 102.6x_4x_5 - 23x_1x_6u_3 \tag{1.62}$$

$$\frac{dx_7}{dt} = -qx_7 + 46x_1x_6u_3 \tag{1.63}$$

$$\frac{dx_8}{dt} = 5.8(qx_1 - u_4) - 3.7u_1 - 4.1u_2 + q(23x_4 + 11x_5 + 28x_6 + 35x_7) - 5u_3^2 - 0.099 \tag{1.64}$$

where

$x_1, x_2, ..., x_7$ = weight fractions of species A, B, ..., G
$k_1, k_2, ..., k_4$ = reaction rate constants
u_1, u_2, u_4 = inlet feed rate of pure components A, B and C
u_3 = square root of light intensity

$q = u_1 + u_2 + u_4$ is the total flow rate of inlet streams.

The initial state is

$$\mathbf{x}(0) = [0.1883 \quad 0.2507 \quad 0.0467 \quad 0.0899 \quad 0.1804 \quad 0.1394 \quad 0.1046 \quad 0.0]^T. \tag{1.65}$$

The constraints on the control variables are

$$0 \leq u_1 \leq 20 \tag{1.66}$$

$$0 \leq u_2 \leq 6 \tag{1.67}$$

$$0 \leq u_3 \leq 4 \tag{1.68}$$

$$0 \leq u_4 \leq 20. \tag{1.69}$$

The optimal control problem is to choose u_1, u_2, u_3, and u_4 in the time interval $0 \leq t < t_f$, so that the performance index

$$I = x_8(t_f), \tag{1.70}$$

which gives the economic benefit, is maximized. The final time t_f is specified as 0.2 h. This problem served as a good test problem for iterative dynamic programming [13].

1.8.4 Production of secreted protein in a fed-batch reactor

Determination of the optimal feed rate into a fed-batch reactor to yield the maximum amount of the desired product is a very difficult optimal control problem. The model for the production of secreted protein in a fed-batch reactor presented by Park and Ramirez [24] has been used by several researchers in developing robust methods for handling problems of this nature. The differential equations describing the reactor are

$$\frac{dx_1}{dt} = g_1(x_2 - x_1) - \frac{u}{x_5}x_1 \tag{1.71}$$

$$\frac{dx_2}{dt} = g_2 x_3 - \frac{u}{x_5}x_2 \tag{1.72}$$

$$\frac{dx_3}{dt} = g_3 x_3 - \frac{u}{x_5}x_3 \tag{1.73}$$

$$\frac{dx_4}{dt} = -7.3 g_3 x_3 - \frac{u}{x_5}(20 - x_4) \tag{1.74}$$

$$\frac{dx_5}{dt} = u \tag{1.75}$$

where

$$g_3 = \frac{21.87 x_4}{(x_4 + 0.4)(x_4 + 62.5)} \tag{1.76}$$

$$g_2 = \frac{x_4 e^{-5x_4}}{(x_4 + 0.1)} \tag{1.77}$$

$$g_1 = \frac{4.75 g_3}{(g_3 + 0.12)} \tag{1.78}$$

and

x_1 = level of secreted SUC-s2 in culture, arbitrary units/L
x_2 = level of total SUC-s2 in culture, arbitrary units/L
x_3 = culture cell density, g/L
x_4 = culture glucose level, g/L
x_5 = culture volume, L
u = feed flow rate, L/h.

The initial condition, as given by Park and Ramirez [24], is

$$\mathbf{x}(0) = [0.0 \quad 0.0 \quad 1.0 \quad 5.0 \quad 1.0]^T. \tag{1.79}$$

It should be noted that the three algebraic equations have been arranged in the "reverse" order so that they can be solved readily by direct substitution. Frequently, there are situations where the set of algebraic equations can not be rearranged to provide a solution in a straightforward manner. Therefore, in the next section we examine methods of solving a general set of algebraic equations numerically.

The control is bounded by

$$0 \leq u \leq 2. \tag{1.80}$$

The performance index to be maximized is the total amount of the desired product at the final time, given by

$$I = x_1(t_f) x_5(t_f) \tag{1.81}$$

where the final time t_f is specified as 15 h. Although there is only a scalar control variable, the optimal control policy is quite difficult to obtain, and therefore this problem is a good test for optimization procedures [20].

These four systems provide the reader a feel for the types of problems we wish to consider in this book. In later chapters a wider variety of systems, such as systems involving time delays, will be considered as well.

1.9 Solving algebraic equations

It is well known that the most efficient way of solving a set of *linear* algebraic equations is Gaussian elimination method, which consists of putting the equations into an echelon form and then using back-substitution to obtain the variables in reverse order. However, the solution of a set of *nonlinear* algebraic equations often requires some

ingenuity. Research in the development of different ways of solving a set of nonlinear algebraic equations has been increasing during the past decade. In spite of the research done over the past 40 years, there still does not appear to be any agreement on the best way of solving an arbitrary set of nonlinear equations. In this section we show that a fundamental approach developed and used for simulation of multi-component distillation columns over 25 years ago (see for example the paper by Seppala and Luus [27]) still provides a very reliable and efficient way of dealing with algebraic equations.

1.9.1 Separation of the equations into two groups

Let us consider a general system of N algebraic equations in N unknowns

$$g_i(x_1, x_2, ..., x_N) = 0, \qquad i = 1, 2, ..., N. \tag{1.82}$$

The problem is to find a set of x_i such that all of the N equations are satisfied within an acceptable tolerance. We may further impose conditions to make sure that the solution is also feasible so that we do not have negative mole fractions, or unrealistic temperature for the reaction such as $10K$. If these physical constraints are put directly into the mathematical formulation, then in addition to Eq. (1.82) we also have a set of inequality constraints and the problem can become computationally more difficult. Often these physical constraints are left for the engineer to examine, once a solution to Eq. (1.82) in absence of any additional constraints is obtained.

Let us group the set of N equations into two groups. In the first group we include $N - n$ equations that we call *simple equations*, and in the second group we have the remaining n *difficult equations*. We shall rename the variables, if necessary, so that the class of n difficult equations can be written as

$$f_i(x_1, x_2, ..., x_n, x_{n+1}, ..., x_N) = 0, \qquad i = 1, 2, ..., n. \tag{1.83}$$

The problem is now reduced to finding an efficient way of obtaining the variables $x_1, x_2, ..., x_n$ such that these n algebraic equations are satisfied. One of the best methods is Newton's method which is well suited for solving sets of nonlinear algebraic equations (see, for example, [5]), and one generally does not have to be concerned about having "good" initial estimates even for sets of highly nonlinear and very complex equations [21].

Let us assume that at iteration j we have a value of $\mathbf{x}$, denoted by $\mathbf{x}^j$, for which Eq.(1.83) is not satisfied within the specified tolerance. Then we try to choose a perturbation $\delta\mathbf{x}$ such that

$$\mathbf{f}(\mathbf{x}^j + \delta\mathbf{x}) = \mathbf{0}. \tag{1.84}$$

By using Taylor series, we therefore choose $\delta\mathbf{x}$ such that

$$(\frac{\partial \mathbf{f}^T}{\partial \mathbf{x}})^T \delta\mathbf{x} = -\mathbf{f}(\mathbf{x}^j). \tag{1.85}$$

Eq. (1.85) is a linear equation which can be easily solved by Gaussian elimination for $\delta\mathbf{x}$ and the next value of $\mathbf{x}$ is then chosen as

$$\mathbf{x}^{j+1} = \mathbf{x}^j + \delta\mathbf{x}. \tag{1.86}$$

It is noted that the partial derivatives are calculated in the presence of the simple equality constraints, so that these are difficult to obtain analytically, because $x_{n+1}, x_{n+2}, ..., x_N$ are functions of the particular variable under consideration. This is illustrated by the first example. Computationally, however, the partial derivatives are easily obtained by perturbing the particular variable, and we can write

$$\frac{\partial f_j(x_1, x_2, ..., x_i, ..., x_N)}{\partial x_i} = \frac{f_j(x_1, x_2, ..., x_i + h_i, ..., x_N) - f_j(x_1, x_2, ..., x_i - h_i, ..., x_N)}{2h_i} \tag{1.87}$$

where h_i is taken as a very small perturbation such as $10^{-6}x_i$ if $x_i \neq 0$.

One great advantage of this method is the very rapid convergence in the vicinity of the solution. There is also considerable freedom in choosing how many and which equations to include in each of the two groups. Generally we try to have as many equations as possible in the first group, so that we have the smallest number of difficult equations. Since this method is so simple to program, and does not encounter the types of problems reported by the various methods considered by Shacham [28], it should really be the "first" approach to try.

1.9.2 Numerical examples

All computations were performed in double precision on a Pentium/120 personal digital computer using WATCOM version 9.5 FORTRAN compiler. For all the examples the perturbation h_i for evaluating the derivatives was $10^{-6}x_i$ if $x_i \neq 0$ and 10^{-10} if $x_i = 0$. The computation times reported were obtained by reading the clock at the beginning and at the end of the calculations and therefore include the time taken to write to the screen and to the output files. These computation times are given only to provide an approximate idea of the computational effort required. We choose several examples for which difficulties have been reported in the recent literature.

Example 1

Let us consider the example involving the oxidation of sulphur dioxide to sulphur trioxide presented by Fogler [6] and studied by [28], [29], and [31], where the set of algebraic equations can be arranged into the two groups:

(a) Set of two simple equations

$$x_2 = \frac{43260x_1 + 105128}{1.84x_1 + 77.3} \tag{1.88}$$

$$x_3 = \exp[\frac{42300}{x_2} - 24.2 + 0.17\ln(x_2)] \tag{1.89}$$

and

(b) one difficult equation

$$f_1 = \frac{0.91 - 0.5x_1}{9.1 - 0.5x_1} - \frac{x_1^2}{(1-x_1)^2 x_3} = 0. \tag{1.90}$$

Here we have only a single difficult equality constraint, and thus one free variable. Since x_1 is the conversion, a reasonable initial choice would be 0.4, but also the values 0.9 and 0.0, as used by Shacham [28], were used. As is shown in Figure 1.2, for each of these three starting values convergence to $|f_1| < 10^{-10}$ was readily obtained, resulting in the same value of $\mathbf{x} = [0.533373 \quad 1637.703 \quad 17.9400]^T$. The initial choice of 0.4 yielded convergence to $|f_1| < 10^{-10}$ in 0.05 s and the initial choice $x_1 = 0$ yielded convergence in 0.16 s. These computation times are quite negligible.

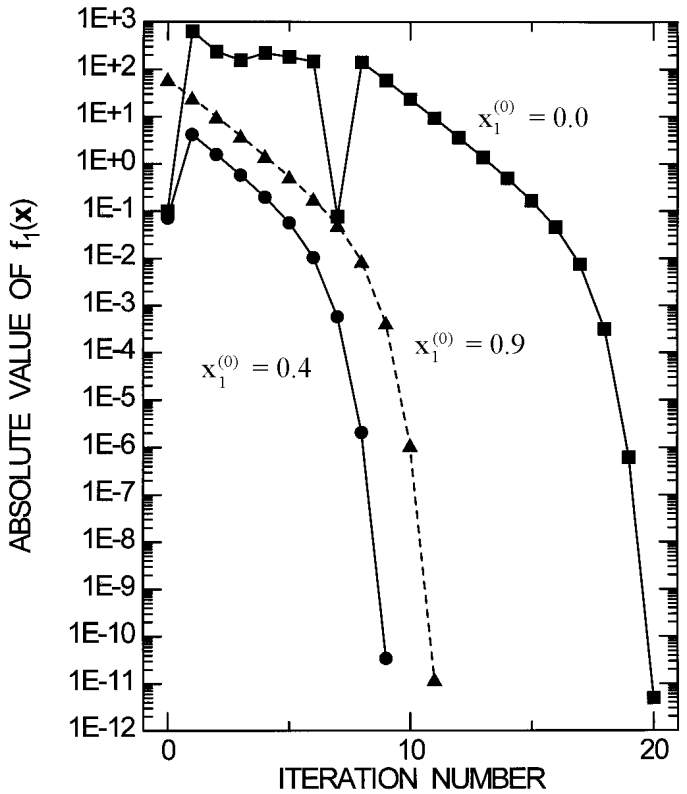

Figure 1.2: Convergence profile for Example 1, where the equations are arranged so that there is only one difficult equality, showing the effect of the starting conditions

Now let us rearrange the same set of equations into the following structure:

(a) Set of one simple equation

$$x_3 = \exp[\frac{42300}{x_2} - 24.2 + 0.17\ln(x_2)] \tag{1.91}$$

and

(b) two difficult equations

$$f_1 = \frac{0.91 - 0.5x_1}{9.1 - 0.5x_1} - \frac{x_1^2}{(1-x_1)^2 x_3} = 0 \tag{1.92}$$

$$f_2 = x_2(1.84x_1 + 77.3) - 43260x_1 - 105128 = 0. \tag{1.93}$$

Here we now have two difficult equations. We solve these with the same initial values for x_1 as used before, and for x_2 we choose $x_2 = 1600$ initially. As with the other arrangement of these equations, convergence to $|f_1| + |f_2| < 10^{-10}$ was readily obtained in almost the same computation times, namely 0.11 s, 0.11 s and 0.28 s, respectively for the three starting points. Although the computational effort per iteration is larger in the latter configuration, fewer iterations are required to reach the convergence criterion for the first two starting points, as is shown in Figure 1.3. Therefore, for this particular example it does not really matter in which of these two ways the equations are arranged. Although this problem was difficult to solve with several software packages by Shacham [28] and by Shacham *et al.* [29], no difficulties were experienced here, and one is not restricted to choosing initial values for x_1 in the very narrow range $0.5 < x_1 < 0.53337$. The listing of the program with the output is given in Appendix A.

Example 2

Let us consider the last example of Gupta [7], which involves 12 variables. The equations are arranged into the structure where the two difficult equations are preceded by 10 simple equality constraints:

$$x_{11} = x_1/(3.846 \times 10^{-5}) \tag{1.94}$$

$$x_{10} = 3 - x_2 \tag{1.95}$$

$$x_6 = 2.597 \times 10^{-3}\sqrt{x_2 x_{11}} \tag{1.96}$$

$$x_7 = 3.448 \times 10^{-3}\sqrt{x_{10} x_{11}} \tag{1.97}$$

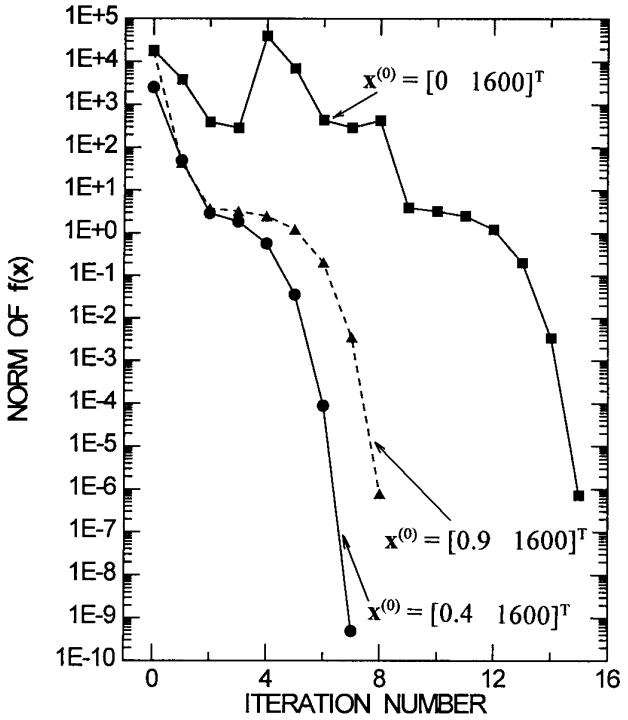

Figure 1.3: Convergence profile for Example 1, where the equations are arranged so that there are two difficult equalities, showing the effect of the starting conditions

$$x_4 = \frac{4 - 0.5(x_6 + x_7)}{1 + 0.193x_2/x_{10}} \tag{1.98}$$

$$x_5 = 0.193x_2x_4/x_{10} \tag{1.99}$$

$$x_{12} = 40 \tag{1.100}$$

$$x_3 = 2x_{12} - 0.5x_5 \tag{1.101}$$

$$x_8 = \frac{1.799x_4x_1}{3.846x_2} \tag{1.102}$$

$$x_9 = \frac{2.155 \times 10^{-4}x_{10}\sqrt{x_3x_{11}}}{x_2} \tag{1.103}$$

$$f_1 = 2(x_1 + x_{10}) + x_2 + x_4 + x_7 + x_8 + x_9 - x_{12} = 0 \tag{1.104}$$

$$f_2 = \sum_{i=1}^{10} x_i - x_{11} = 0. \tag{1.105}$$

Here we have two difficult equality constraints. The free variables are x_1 and x_2. Two sets of initial values for these variables were taken, namely (0.1, 0.1) and (0.5, 0.5). The only precaution required was to keep x_1 and x_2 positive so that the square root could be calculated. Thus, if x_1 or x_2 were calculated to be less than 10^{-4}, then 10^{-4} was arbitrarily assigned. This precaution eliminates any possibility of numerical

difficulty. As is shown in Figure 1.4, convergence from each of these starting values is very rapid and we obtain $|f_1| + |f_2| < 10^{-10}$ in only 11 iterations, requiring 0.11 s of computation time. At the eleventh iteration the norm was zero, so the point is not plotted in the figure. The solution is $\mathbf{x}^T = [0.0044998\ 0.0023645\ 79.999698\ 3.966427\ 0.000604\ 0.001366\ 0.064573\ 3.530816\ 26.431550\ 2.997635\ 116.999533\ 40.000000]$. Here the starting point is not very important, but it is noted that after the norm is less than 1, the convergence rate is very fast.

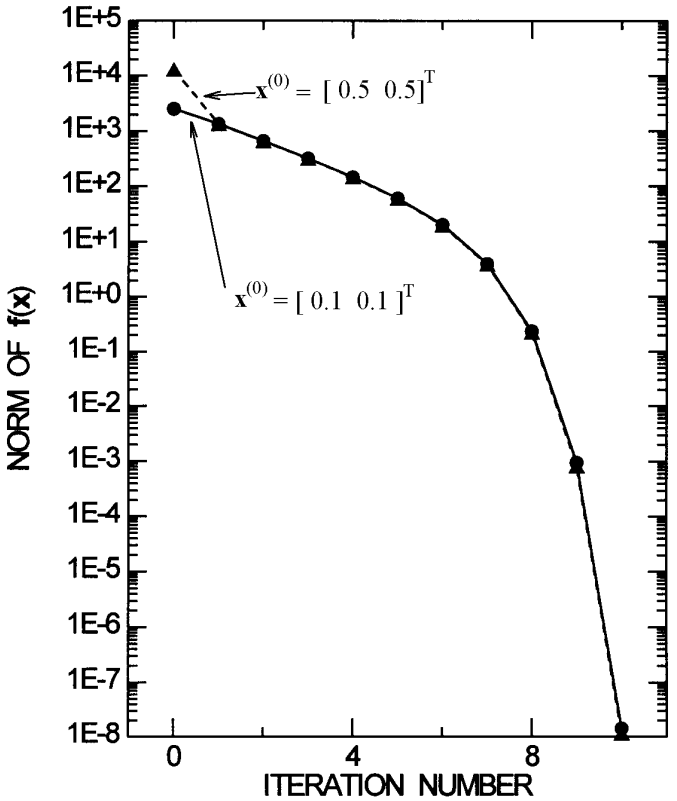

Figure 1.4: Convergence profile for Example 2, showing the effect of the choice of starting conditions

Example 3

Let us consider the set of algebraic equations describing a chemical equilibrium resulting from partial oxidation of methane with oxygen, as considered by Carnahan *et al.* [5]. This example was also used by Shacham [28] to test several computer codes for solving algebraic equations. The equations may be arranged as follows:

$$x_4 = \frac{x_1 x_3}{2.6058 x_2} \tag{1.106}$$

$$x_5 = \frac{400 x_1 x_4^3}{1.7837 \times 10^5 x_3} \tag{1.107}$$

$$x_7 = \frac{2}{x_3 + x_4 + 2x_5} \tag{1.108}$$

$$x_6 = x_7(0.5(x_1 + x_3) + x_2) \tag{1.109}$$

$$f_1 = x_1 + x_2 + x_5 - \frac{1}{x_7} = 0 \tag{1.110}$$

$$\begin{aligned} f_2 &= -28837x_1 - 139009x_2 - 78213x_3 + 18927x_4 + 8427x_5 \\ &+ \frac{13492 - 10690x_6}{x_7} = 0 \end{aligned} \tag{1.111}$$

$$f_3 = x_1 + x_2 + x_3 + x_4 + x_5 - 1 = 0. \tag{1.112}$$

The equations may also be arranged into the structure where we have five simple equalities and only two difficult equality constraints:

$$x_3 = \frac{5.2116x_2(x_1 + x_2)}{x_1 + 2.6058x_2} \tag{1.113}$$

$$x_4 = 2(x_1 + x_2) - x_3 \tag{1.114}$$

$$x_5 = \frac{400x_1x_4^3}{1.7837 \times 10^5 x_3} \tag{1.115}$$

$$x_7 = \frac{2}{x_3 + x_4 + +2x_5} \tag{1.116}$$

$$x_6 = x_7(0.5(x_1 + x_3) + x_2) \tag{1.117}$$

$$\begin{aligned} f_1 &= -28837x_1 - 139009x_2 - 78213x_3 + 18927x_4 + 8427x_5 \\ &+ \frac{13492 - 10690x_6}{x_7} = 0 \end{aligned} \tag{1.118}$$

$$f_2 = x_1 + x_2 + x_3 + x_4 + x_5 - 1 = 0. \tag{1.119}$$

Since the first 5 variables denote mole fractions, it should be possible to make reasonable choices for the first 3 variables in the first arrangement and for the first 2 variables in the second arrangement. Although, as reported by Shacham [28] this problem was too difficult for most of the programs he evaluated, there were no difficulties with the recommended procedure. For stability in calculations, if any of the x_i, $i = 1, \cdots, n$ would be less than 10^{-4}, the corresponding variable was put to 10^{-4}, and similarly the upper clipping limit of 0.95 was used. Convergence profiles from the starting point $x_1 = 0.22, x_2 = 0.075, x_3 = 0.001$ for the first formulation of this problem with three difficult equality constraints and from $x_1 = 0.22, x_2 = 0.075$ for the second formulation are shown in Figure 1.5.

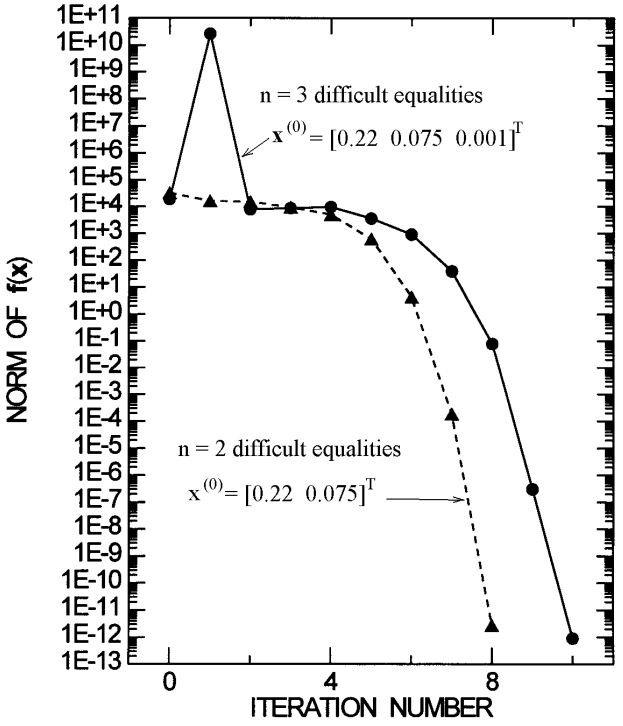

Figure 1.5: Effect of the formulation of the problem in solving Example 3

The use of two difficult equality constraints, rather than three, tends to provide faster convergence. The difference in computation time could not be measured accurately, because each required only about 0.06 s of computation time as read from the clock on the Pentium/120. In each case convergence to $\mathbf{x}^T = [\ 0.322871\ 0.009224\ 0.046017\ 0.618172\ 0.003717\ 0.576715\ 2.977863\ ]$ was obtained. These are the same results as reported by Carnahan *et al.* [5]

As is shown in Figure 1.6, similar convergence characteristics are obtained with the initial starting point of $x_i = 0.1$ for $i = 1, ..., n$. It is interesting to note that once the norm is below 1, the convergence rate is very fast. This suggests that another means could be used to get to the vicinity of the solution, and then a switch could be made to the Newton's method. One such means is to use the direct search optimization suggested by Luus and Jaakola [22] to establish a good starting point to be used for Newton's method. This optimization procedure is outlined in Chapter 2, where we examine steady-state optimization.

Example 4

Let us consider the esterification reaction of monobutyl-phthalate with n-butanol to produce dibutyl phthalate and water in two continuous stirred tank reactors in

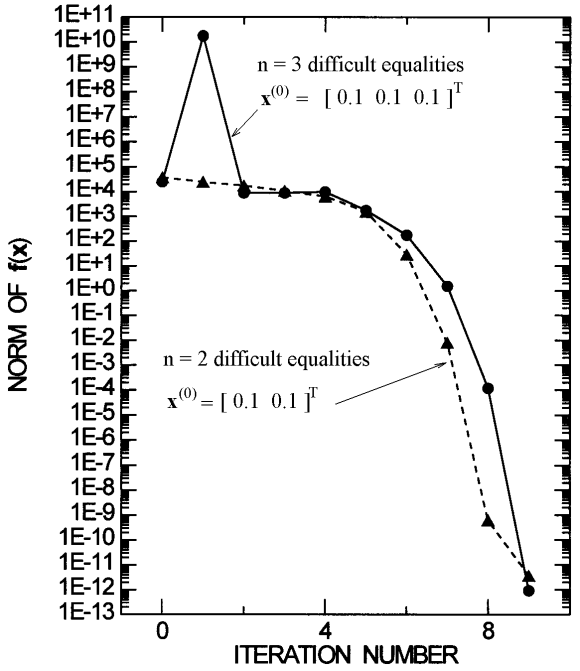

Figure 1.6: Effect of formulation of the problem in solving Example 3, with initial starting point $x_i = 0.1$, $i = 1, \cdots, n$

series as used by Seader *et al.* [26] and Gupta [7]. The system of equations can be written in terms of seven simple equalities followed by two difficult equality constraints as

$$x_9 = x_1 + x_2 \tag{1.120}$$

$$x_3 = \frac{x_9(6.875x_9 + x_2)}{5.797x_9 + x_2} \tag{1.121}$$

$$x_4 = 1.25x_1 + 2.25x_2 - 1.054x_3 \tag{1.122}$$

$$x_7 = 0.025714x_3 - 0.007346189x_2 - 0.2720539x_4 \tag{1.123}$$

$$x_5 = 12.128256x_3 - x_2 - 124x_4 - 1.75x_7 - 75.85949 \tag{1.124}$$

$$x_6 = 61.177 - 8.7614x_3 - x_2 + 91x_4 - x_5 + x_7 \tag{1.125}$$

$$x_8 = 1.0986x_3 - x_2 - 9x_4 - x_5 - x_6 + x_7 \tag{1.126}$$

$$f_1 = x_4(x_3 - x_2 - x_1)^2 - 2056.4(0.0986x_3 - x_4)^2 = 0 \tag{1.127}$$

$$f_2 = 5.5(0.0986x_3 - x_2 - x_5 - 0.01x_8)(x_5 + x_6) - x_5x_8 = 0. \tag{1.128}$$

Here x_1 represents the amount of BuOH in off-gas and x_2 represents the amount of water in off-gas from the first reactor expressed in lb mol/h.

By picking 100 random starting points in the range $0 \leq x_1 \leq 100, 0 \leq x_2 \leq 100$, convergence to $\mathbf{x^T} = [$ 61.15373 6.97629 80.57946 7.20807 0.54718 3.65884 0.05979

12.52943 68.13003] was obtained 66 times. The other 34 starting points gave nonfeasible solutions by having some of the variables negative. The total computation time for the 100 starting conditions was 0.11 s. The success rate was decreased to 28 out of 100 starting points by taking a smaller range for the initial values for x_2, namely $0 \leq x_2 \leq 10$, as was used by Gupta [7], but the computation time was reduced to 0.06 s. By using the range $0 \leq x_1 \leq 1000, 0 \leq x_2 \leq 1000$, the success rate was 57 times out of 100, obtained also in a total computation time of 0.06 s.

Example 5

Sometimes it is important to capture all the possible solutions to a set of nonlinear algebraic equations. For example, in the problem of finding the maximum distance from the origin to the intersection of an ellipsoid with a hyperboloid, as considered by Luus [12], and by Wang and Luus [30], when Lagrange multipliers are used, the optimum lies at one of the 8 stationary points. Therefore, it is important to locate all of the stationary points, and the stationary point that gives the maximum value for the square of the distance $P = {x_1}^2 + {x_2}^2 + {x_3}^2$ provides the solution to the problem. The system of algebraic equations is

$$x_7 = \frac{x_3}{2x_3 + 0.2x_2} \tag{1.129}$$

$$x_6 = 4(x_2 - 0.2) + 0.1x_1 + 0.2x_3 \tag{1.130}$$

$$x_5 = \frac{(x_6 x_7 - x_2)}{x_2 + 2x_6 x_7} \tag{1.131}$$

$$x_4 = (4x_5 - 2)x_7 \tag{1.132}$$

$$f_1 = 2x_1 + x_4[8(x_1 - 0.5) + 0.1x_2] + 4x_1 x_5 \tag{1.133}$$

$$f_2 = 2({x_1}^2 - {x_3}^2 - 1) + {x_2}^2 \tag{1.134}$$

$$f_3 = 4(x_1 - 0.5)^2 + 2(x_2 - 0.2)^2 + {x_3}^2 + 0.1x_1 x_2 + 0.2x_2 x_3 - 16, \tag{1.135}$$

where x_4 and x_5 are Lagrange multipliers, and we have introduced x_6 and x_7 to simplify the formulation of the problem. By choosing 200 random starting points for x_1, x_2, and x_3 in the range $-2 \leq x_i \leq 2, i = 1, 2, 3$, we were able to locate all the 8 stationary points with the frequencies given in Table 1.1. It is noted that one of the stationary points was obtained only twice and another one only three times, whereas the stationary point related to the maximum distance was obtained 21 times. To solve the algebraic equations from the 200 starting points took a minimum of 6 iterations and a maximum of 15 iterations, with a total computation time of 0.11 s on the Pentium/120 for the 200 starting points. Therefore, the computation time is quite negligible.

Table 1.1: Solutions to Example 5, showing the existence of 8 solutions

x_1	x_2	x_3	x_4	x_5	P	Frequency
0.98842	2.67366	-1.88446	-0.67341	0.21106	11.67664	21
1.03962	2.49153	1.78457	-0.59958	0.15835	10.47324	97
1.37627	-2.12617	1.77606	-0.59704	0.23721	9.56909	2
2.22372	-0.15747	1.98930	-0.48910	0.25739	8.92702	6
1.56816	-1.80074	-1.75513	-0.53053	0.20752	8.78227	41
2.04323	-0.95605	-1.90573	-0.49408	0.24057	8.72063	10
-1.41865	-0.16020	1.01262	-0.31252	0.34621	3.06362	3
-1.40504	-0.24481	-1.00205	-0.30976	0.34134	3.03818	20

Example 6

Another problem for which computational difficulties have been recently reported by Shacham *et al.* [29] is the calculation of the dew point temperature for a 20% isobutanol and 80% water mixture. This is based on the problem in Henley and Rosen [8]. The problem consists of 16 algebraic equations that can be arranged into a structure of 13 simple equations and 3 difficult equations:

$$x_4 = 1 - x_1 \tag{1.136}$$

$$x_5 = 1 - x_2 \tag{1.137}$$

$$\log_{10} x_6 = 7.62231 - \frac{1417.90}{191.15 + x_3} \tag{1.138}$$

$$\log_{10} x_7 = 8.10765 - \frac{1750.29}{235.00 + x_3} \tag{1.139}$$

$$\log_{10} x_8 = \frac{1.7 x_4^2}{(1.7 x_1 / 0.7 + x_4)^2} \tag{1.140}$$

$$\log_{10} x_9 = \frac{0.7 x_1^2}{(x_1 + 0.7 x_4 / 1.7)^2} \tag{1.141}$$

$$\log_{10} x_{10} = \frac{1.7 x_5^2}{(1.7 x_2 / 0.7 + x_5)^2} \tag{1.142}$$

$$\log_{10} x_{11} = \frac{0.7 x_2^2}{(x_2 + 0.7 x_5 / 1.7)^2} \tag{1.143}$$

$$x_{12} = x_6 x_8 / 760 \tag{1.144}$$

$$x_{13} = x_7 x_9 / 760 \tag{1.145}$$

$$x_{14} = x_{10}x_6/760 \tag{1.146}$$

$$x_{15} = x_{11}x_7/760 \tag{1.147}$$

$$x_{16} = \frac{0.2/x_1 - x_{12}/x_{14}}{1 - x_{12}/x_{14}} \tag{1.148}$$

$$f_1 = x_{12}x_1 + x_{13}x_4 - 1 \tag{1.149}$$

$$f_2 = x_2 - \frac{x_1 x_{12}}{x_{14}} \tag{1.150}$$

$$f_3 = x_4 - \frac{0.8}{x_{16} + (1 - x_{16})x_{13}/x_{15}}. \tag{1.151}$$

Table 1.2: Four solutions to Example 6, showing the frequency with which the solutions were obtained from 1000 randomly chosen starting points

Variable	Definition	Solution 1	Solution 2	Solution 3	Solution 4
x_1	$x_{1,1}$	0.022698	0.686748	0.200000	0.200000
x_2	$x_{1,2}$	0.686748	0.022698	0.031601	0.904122
x_3	Temperature	88.537830	88.537830	86.423947	86.423947
x_4	$x_{2,1}$	0.9773028	0.313252	0.800000	0.800000
x_5	$x_{2,2}$	0.313252	0.977302	0.968399	0.095878
x_6	P_1	357.050288	357.050288	326.679334	326.679334
x_7	P_2	498.658802	498.658802	459.435969	459.935969
x_8	$\gamma_{1,1}$	33.366490	1.102821	4.551692	4.551692
x_9	$\gamma_{2,1}$	1.004606	3.134222	1.258639	1.258639
x_{10}	$\gamma_{1,2}$	1.102821	33.366490	28.807440	1.006876
x_{11}	$\gamma_{2,2}$	3.134222	1.004606	1.008729	4.391885
x_{12}	$K_{1,1}$	15.675677	0.518108	1.956505	1.956505
x_{13}	$K_{2,1}$	0.659152	2.056457	0.760874	0.760874
x_{14}	$K_{1,2}$	0.518108	15.675677	12.382625	0.432797
x_{15}	$K_{2,2}$	2.056457	0.659152	0.609798	2.654987
x_{16}	β	0.732999	0.267001	1.000000	1.000000
Frequency		16	30	10	23

The 16 variables are defined in Table 1.2, where also the 4 solutions are given. These solutions were obtained by taking 1000 starting points in the range $0.01 \le x_1 \le 0.99$, $0.01 \le x_2 \le 0.99$, and $80 \le x_3 \le 90$. If during the course of calculations any of the variables became negative, if either x_1 or x_2 became less than 10^{-4} or greater than 0.9999, or if x_{16} exceeded 1.01, then the calculations for that initial point were stopped and a new starting point was taken. From the 1000 starting points only 79 led to solutions where the absolute value norm of the left hand sides of the difficult

equality constraints was less than 10^{-10}. The total computation time for the run was 0.38 s on a Pentium/120, showing a computation time of about 0.005 s per successful solution.

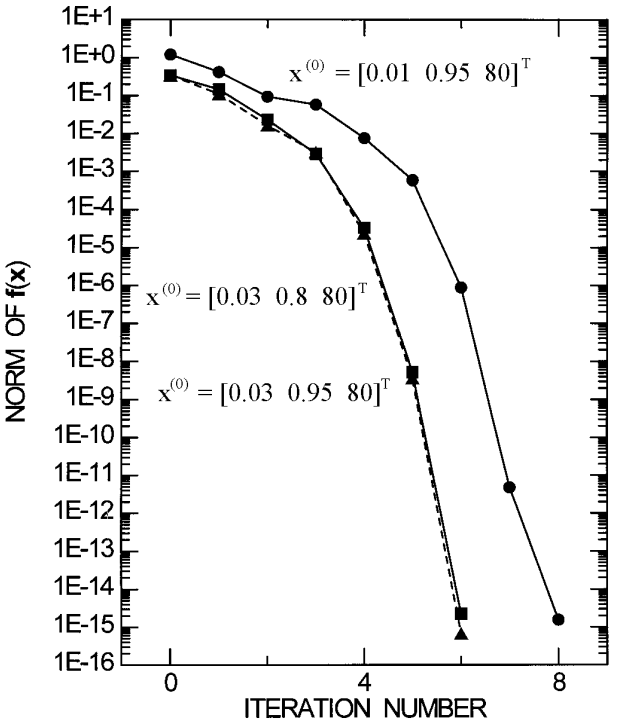

Figure 1.7: Convergence profile to Solution 1 for Example 6, showing the effect of starting conditions

The convergence profile from three different starting conditions is given in Figure 1.7, where no difficulties were encountered and convergence to Solution 1 was readily obtained. Since the convergence is so rapid, it is natural to take numerous starting points chosen at random over some allowable range, to capture *all* the possible solutions. The constraints restricting calculations for only meaningful solutions not only reduce the computational effort, but ensure stability during calculations.

Now that the personal computers are very fast, the computation time is negligible for all of these 6 examples that have been outlined here. Of greater concern is the robustness of the method. One of the greatest advantages of separating the set of algebraic equations into two groups is that starting conditions must be specified only for the set of difficult equalities. Also, by minimizing the number of equations included in that set, we minimize the computational effort. Although for these 6 problems outlined here, the computational effort is negligible, if a situation arises

where a set of algebraic equations must be solved repeatedly a large number of times, the way the equations are set up can have a profound effect.

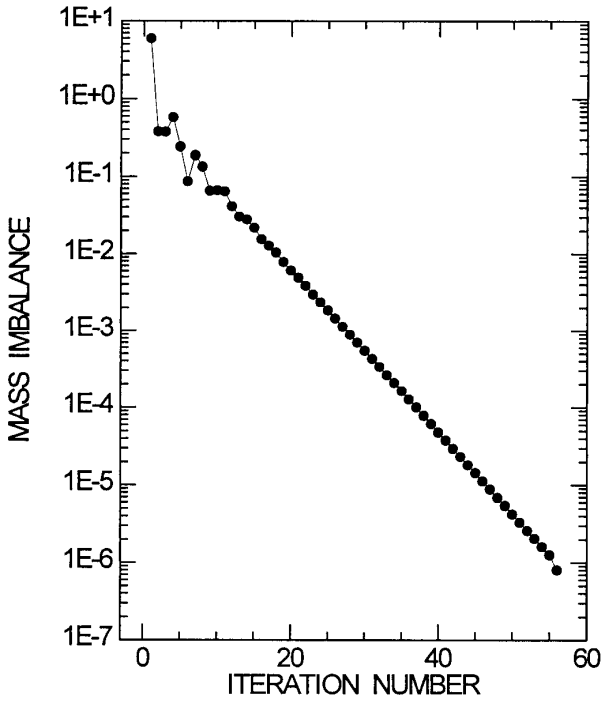

Figure 1.8: Convergence profile for multicomponent distillation column described by 750 algebraic equations

1.9.3 Application to multicomponent distillation

In simulation of multicomponent distillation at steady state, we encounter a very large number of nonlinear algebraic equations called the MESH equations that include the material balance (M), equilibrium equations (E), the summation of mole fractions (S) and the heat balance equations (H). For a distillation column consisting of n stages, in which a feed of m components is distilled, there are a total of $2mn + 3n$ equations to be solved simultaneously. A feed consisting of 11 components fed into a distillation column of 30 stages thus presents a problem of solving 750 algebraic equations, many of which are nonlinear.

By using the thermodynamic data given by Seppala and Luus [27] for 11 components, and using a distillation column of 30 stages with feed to the 11^{th} stage and a reflux ratio of 0.4, by grouping of equations, the 750 algebraic equations were solved in 0.11 s on a PentiumII/350 digital computer giving a mass imbalance of less than

10^{-6} in 56 iterations as is shown in Figure 1.8. The mass imbalance is defined by

$$M = \sum_{i=1}^{m} |Fz_{iF} - DK_{i1}x_{i1} - Bx_{in}|, \tag{1.152}$$

where F is the feed flow rate to stage F, here taken as 11, with composition z_{iF}, D is the distillate flow rate from the partial condenser with liquid composition x_{i1} and B is the bottom flow rate with liquid composition x_{in}.

The only starting points required to solve the algebraic equations are very crude estimates for the top and the bottom temperatures of the distillation column. For this particular run we chose the bottom temperature of 810 and top temperature of 510. The actual temperatures for the bottom and top stages are 764.75 and 589.84 respectively. When another run was made with initial estimates for the bottom and top temperatures of 765 and 590, 57 iterations were required for convergence. The same temperature profile was obtained and the computation time was also 0.11 s. Needless to say, such an approach is very easy to use.

1.10 Solving ordinary differential equations

There are numerous methods of numerically integrating a set of ordinary differential equations. For some common methods, the reader could refer to the excellent book by Carnahan *et al.* [5]. To balance the accuracy with speed, it has been found that the fourth order Runge-Kutta method is well suited for many problems. One has direct control over the integration time step, so that the most appropriate step size can be used for well behaved problems.

There are situations, however, when a constant step size is not appropriate, so one method which has been found very useful is an IMSL routine based on the Runge-Kutta method called DVERK, programmed by Hull *et al.* [9]. Here the user specifies the local error tolerance and the routine chooses the step-size required. We have found this method most useful and, with the permission of the authors, we have included the listing of DVERK in Appendix E. This routine is also available by connecting to their website, http://www.cs.toronto.edu/NA/index.html, moving to the topic **Mathematical Software** and clicking on DVERK.

1.11 References

[1] Aris, R. and Amundson, N.R.: "An analysis of chemical reactor stability and control", *Chem. Eng. Sci.* **7** (1958), 121-147.

[2] Bashein, G.: "A simplex algorithm for on-line computation of time optimal control problems", *IEEE Trans. Auto. Control* **AC-16** (1971), 479-482.

[3] Bennett, H.W. and Luus, R.: "Application of numerical hill-climbing in control of systems via Liapunov's direct method", *Can. J. Chem. Eng.* **49** (1971), 685-690.

[4] Bojkov, B. and Luus, R.: "Time optimal control of high dimensional systems by iterative dynamic programming", *Can. J. Chem. Eng.* **73** (1995), 380-390.

[5] Carnahan, B., Luther, H.A., and Wilkes, J.O.: *Applied Numerical Methods*, John Wiley & Sons, Inc., New York (1969).

[6] Fogler, H.S.: *Elements of Chemical Kinetics and Reactor Calculations*, Prentice Hall, Englewood Cliffs, NJ (1974), pp. 335-347.

[7] Gupta, Y.P.: "Bracketing method for on-line solution for low-dimensional nonlinear algebraic equations", *Ind. Eng. Chem. Res.* **34** (1995), 536-544.

[8] Henley, E.J. and Rosen, E.H.: *Material and Energy Balance Computations*, John Wiley, New York (1969), p. 354.

[9] Hull, T.E., Enright, W.D., and Jackson, K.R.: *User Guide to DVERK–a Subroutine for Solving Nonstiff ODE's*. Report 100 (1976), Department of Computer Science, University of Toronto, Canada.

[10] Jensen, T.: *Dynamic Control of Large Dimension Nonlinear Chemical Processes*, Ph.D. Dissertation, Princeton University (1964).

[11] Lapidus, L. and Luus, R.: *Optimal Control of Engineering Processes*, Blaisdell, Waltham, Mass. (1967).

[12] Luus, R.: "Two-pass method for handling difficult equality constraints in optimization", *AIChE J.* **20** (1974), 608-610.

[13] Luus, R.: "Application of dynamic programming to high-dimensional nonlinear optimal control problems", *Int. J. Control* **52** (1990), 239-250.

[14] Luus, R.: "Effect of the choice of final time in optimal control of nonlinear systems", *Can. J. Chem. Eng.* **69** (1991), 144-151.

[15] Luus, R.: "Sensitivity of control policy on yield of a fed-batch reactor", *Proc. IASTED Int. Conf. on Modelling and Simulation*, Pittsburgh, PA, April 27-29, 1995, pp. 224-226.

[16] Luus, R.: "Numerical convergence properties of iterative dynamic programming when applied to high dimensional systems", *Chem. Eng. Res. Des.* **74** (1996), 55-62.

[17] Luus, R. and Bojkov, B.: "Global optimization of the bifunctional catalyst problem", *Can. J. Chem. Eng.* **72** (1994), 160-163.

[18] Luus, R. and Cormack, D.E.: "Multiplicity of solutions resulting from the use of variational methods in optimal control problems", *Can. J. Chem. Eng.* **50** (1972), 309-312.

[19] Luus, R., Dittrich, J., and Keil, F.J.: "Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor", *Can. J. Chem. Eng.* **70** (1992), 780-785.

[20] Luus, R. and Hennessy, D.: "Optimization of fed-batch reactors by the Luus-Jaakola optimization procedure", *Ind. Eng. Chem. Res.* **38** (1999), 1948-1955.

[21] Luus, R. and Jaakola, T.H.I.: "Optimization of nonlinear functions subject to equality constraints. Judicious use of elementary calculus and random numbers", *Ind. Eng. Chem. Process Des. and Develop.* **12** (1973a), 380-383.

[22] Luus, R. and Jaakola, T.H.I.: "Optimization by direct search and systematic reduction of the size of search region", *AIChE J.* **19** (1973b), 760-766.

[23] Luus, R. and Smith, S.G.: "Application of dynamic programming to high-dimensional systems described by difference equations", *Chem. Eng. Tech.* **14** (1991), 122-126.

[24] Park, S. and Ramirez, W.F.: "Optimal production of secreted protein in fed-batch reactors", *AIChE J.* **34** (1988), 1550-1558.

[25] Rao, S.N. and Luus, R.: "Evaluation and improvement of control vector iteration procedures for optimal control", *Can. J. Chem. Eng.* **50** (1972), 777-784.

[26] Seader, J.D., Kuno, M., Lin, W.J., Johnson, S.A., Unsworth, K., and Wiskin, J.W.: "Mapped continuation methods for computing all solutions to general systems of nonlinear equations", *Computers & Chem. Eng.* **14** (1990), 71-75.

[27] Seppala, R.E. and Luus, R.: "Evaluation and improvement of simulation procedures for multicomponent distillation", *J. of Franklin Inst.* **293** (1972), 325-344.

[28] Shacham, M.: "Comparing software for the solution of systems of nonlinear algebraic equations arising in chemical engineering", *Computers & Chem. Eng.* **9** (1985), 103-112.

[29] Shacham, M., Brauner, N., and Pozin, M.: "Comparing software for interactive solution of systems of nonlinear algebraic equations", *Computers & Chem. Eng.* **22** (1998), 323-331.

[30] Wang, B.C. and Luus, R.: "Reliability of optimization procedures for obtaining global optimum", *AIChE J.* **19** (1978), 619-626.

[31] Zain, O.S., Kumar, S., and Shashi: "Application of modified nonlinear ORTHOMIN to chemical process simulation", *Hung. J. Ind. Chem.* **26** (1998), 13-18.

Chapter 2

Steady-state optimization

2.1 Introduction

We consider the general problem of minimizing or maximizing the real-valued performance index

$$I = f(x_1, x_2, ..., x_n) \tag{2.1}$$

subject to the set of equality constraints

$$\phi_i(x_1, x_2, ..., x_n) = 0, \qquad i = 1, 2, ..., m \tag{2.2}$$

with $m \leq n$, and a set of inequality constraints

$$g_j(x_1, x_2, ..., x_n) \leq 0, \qquad j = 1, 2, ..., s, \tag{2.3}$$

through the appropriate choice of $x_1, x_2, ..., x_n$.

There are many excellent books written to deal with steady-state optimization as described by Equations (2.1)-(2.3). For example, one may refer to the books by Himmelblau [7] and Edgar and Himmelblau [4]. Here we wish to outline only two optimization procedures: linear programming and the direct search optimization procedure of Luus and Jaakola [29], referred to as the LJ optimization procedure. Linear programming is important to give the reader an understanding of shadow prices that provide useful sensitivity information. The shadow prices are related to Lagrange multipliers, which in turn are similar to the adjoint variables encountered in optimal control. The LJ optimization procedure provides the basis for iterative dynamic programming where we use a systematic reduction in the size of the regions allowed for control candidates.

2.2 Linear programming

One of the most useful and powerful means of solving multi-dimensional steady-state optimization problems is *linear programming*. There are three requirements for its

use. First, the performance index must be a linear function of the variables. Second, the model, as expressed by the constraining equations, must be linear, and, third, the variables must be positive semi-definite. Specifically, the problem for linear programming can be posed as follows.

Problem:

Determine the values of the variables $x_1, x_2, \cdots, x_n$ such that the performance index

$$I = p_1 x_1 + p_2 x_2 + \cdots + p_n x_n \tag{2.4}$$

is maximized, subject to the constraints

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.5}$$

and

$$x_i \geq 0, \quad i = 1, 2, \cdots, n \tag{2.6}$$

where $\mathbf{b}$ is an $(m \times 1)$ vector with $b_j \geq 0$, $j = 1, 2, \cdots, m$. Also, the $(m \times n)$ matrix $\mathbf{A}$ has the special form

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & a_{1,m+1} & \cdots & a_{1j} & \cdots & a_{1n} \\ 0 & 1 & 0 & \cdots & 0 & 0 & a_{2,m+1} & \cdots & a_{2j} & \cdots & a_{2n} \\ . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & \cdots & 1 & 0 & a_{m-1,m+1} & \cdots & a_{m-1,j} & \cdots & a_{m-1,n} \\ 0 & 0 & 0 & \cdots & 0 & 1 & a_{m,m+1} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix} \tag{2.7}$$

where the first m columns of $\mathbf{A}$ consisting of elementary $(m \times 1)$ unit vectors make up an $(m \times m)$ identity matrix.

Solution:

Picture $\mathbf{A}$ as being partitioned into n column vectors, so that Eq. (2.5) may be written as

$$\mathbf{b} = x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \cdots + x_m \mathbf{a}_m + x_{m+1} \mathbf{a}_{m+1} + \cdots + x_n \mathbf{a}_n. \tag{2.8}$$

Since $\mathbf{b}$ is a vector of m elements, only m of the vectors $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n$ can be linearly independent. Let us choose these to be the first m vectors, since these vectors constitute the natural basis for the m-dimensional space. Then a feasible solution to Eq. (2.8) is

$$\mathbf{x}^T = [b_1 \quad b_2 \quad \cdots \quad b_m \quad 0 \quad 0 \quad \cdots \quad 0], \tag{2.9}$$

and the corresponding value of the performance index as calculated from Eq. (2.4) is

$$I = p_1 b_1 + p_2 b_2 + \cdots + p_m b_m + p_{m+1} 0 + \cdots + p_n 0. \tag{2.10}$$

The next step is to improve the performance index by changing the basis of the m-dimensional vector space. We accomplish this by bringing into the basis one of the $\mathbf{a}_j$ where $j > m$, and removing one of the vectors from the basis. Since the basis $\{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m\}$ consists of unit vectors, we can expand $\mathbf{a}_j$ immediately in terms of its elements as

$$\mathbf{a}_j = a_{1j}\mathbf{a}_1 + a_{2j}\mathbf{a}_2 + \cdots + a_{mj}\mathbf{a}_m. \tag{2.11}$$

We can now write Eq. (2.8), by assigning the values of x_i from Eq. (2.9), as

$$\mathbf{b} = \sum_{i=1}^{m} b_i\mathbf{a}_i + \alpha\mathbf{a}_j - \alpha\mathbf{a}_j, \tag{2.12}$$

where we have given the flexibility of including $\mathbf{a}_j$ through the positive constant α. By using Eq. (2.11), Eq. (2.12) can be rearranged into the form

$$\mathbf{b} = \sum_{i=1}^{m}(b_i - \alpha a_{ij})\mathbf{a}_i + \alpha\mathbf{a}_j. \tag{2.13}$$

The new value of the performance index now becomes

$$I_{new} = \sum_{i=1}^{m}(b_i - \alpha a_{ij})p_i + \alpha p_j, \tag{2.14}$$

which can be rearranged into

$$I_{new} = \sum_{i=1}^{m} b_i p_i + \alpha[p_j - \sum_{i=1}^{m} a_{ij}p_i]. \tag{2.15}$$

Since the first term of Eq. (2.15) is simply the old value of the performance index given by Eq. (2.10), we can rewrite this equation as

$$I_{new} = I_{old} + \alpha(p_j - z_j) \tag{2.16}$$

where

$$z_j = \sum_{i=1}^{m} a_{ij}p_i. \tag{2.17}$$

To ensure that the performance index I is increased in value, the quantity $(p_j - z_j)$ should be positive. Therefore, to choose which vector $\mathbf{a}_j, j = m+1, \cdots, n$ should come into the basis, we choose the $\mathbf{a}_j$ for which $(p_j - z_j)$ is maximum. We would like to choose α as large as possible, but constraint (2.6) must be satisfied. Thus, from Eq. (2.13) with the new value $x_i = (b_i - \alpha a_{ij})$, the largest value of α that can be chosen is

$$\alpha = \min_k[\frac{b_k}{a_{kj}}] > 0, \quad k = 1, 2, \cdots, m. \tag{2.18}$$

If for all replacement j we find that $(p_j - z_j) \leq 0$, then we have arrived at the solution if it exists. The quantity $(z_j - p_j)$ is called the *shadow price* of the variable j and

indicates the amount by which p_j needs to be increased so that the variable j would be in the solution.

After the vector $\mathbf{a}_j$ enters the basis, elementary row operations are carried out on Eq. (2.5) so that the basis again consists of elementary unit vectors. The procedure is repeated until the shadow prices of all the vectors outside the basis are positive or zero.

The above procedure is easy to program and is computationally fast. Usually the original problem, however, is formulated in terms of inequalities rather than in the form of equality constraints. This introduces no difficulty, since inequalities can be easily converted into equalities by the use of *slack* variables or *surplus* variables. We can also introduce *artificial* variables to construct the $\mathbf{A}$ matrix to include extra columns if required to make up the identity matrix. The method of introducing these auxiliary variables is illustrated in the example.

For convenience, in keeping track of the variables in a particular problem, we rearrange the columns of $\mathbf{A}$ so that the last, rather than the first, m columns constitute the identity matrix. Such rearrangement of columns makes no difference, since only the subscripts have to be rearranged. In using linear programming, one does not usually input the matrix $\mathbf{A}$, but provides sufficient information so that the matrix can be generated by the computer. These ideas are illustrated by the following example.

2.2.1 Example – diet problem with 5 foods

Let us consider the problem of selecting, from five foods, a snack which gives the maximum amount of satisfaction, and which also provides some nutritional value. Let us suppose that while attempting to gain the maximum satisfaction, we are concerned about cost, calories, protein, and iron. This information, along with constraints, is given in Table 2.1.

Each food is rated on a satisfaction scale from 1 to 100, called utility, where, for example beer is given a utility value of 100 and hot dog 20. These utility values will change from one individual to another, and from day to day. By choosing these values, we are saying that we prefer beer 5 times more than a hot dog, and we prefer a hamburger slightly more than a hot dog on that particular day. The nutritional values for the foods have been taken from the report of the Health and Welfare Canada [6] and the costs are estimated. The goal is to maximize total utility, obtained by summing the amount of utility obtained from each food, and to meet the constraints imposed in Table 2.1. It is assumed that the satisfaction gained from each food is proportional to the amount of the food consumed.

The total cost should not exceed \$4, and we do not want to include more than 800 calories (food calories which are really kcal). However, we would like to gain at least 25 g protein and 3.5 mg of iron. As indicated in the last column of Table 2.1, there is also an upper limit placed upon the amount of each food.

If we denote by x_i the amount of food i, then the performance index to maximize

Table 2.1: Diet problem with 5 foods

Food	Utility	Cost dollars	Calories	Protein gm	Iron mg	Maximum allowed
Beer, 12-oz bottle	100	1.25	150	1.0	0.0	2
Pizza, sector	50	0.90	185	7.0	0.7	3
Hamburger	25	0.99	245	21.0	2.7	2
Hot dog	20	1.10	270	8.6	1.3	1
Grapefruit	35	0.79	90	2.0	1.0	1
Constraints		4.00	800	25	3.50	

is

$$I = 100x_1 + 50x_2 + 25x_3 + 20x_4 + 35x_5 \tag{2.19}$$

subject to the constraints

$$1.25x_1 + 0.9x_2 + 0.99x_3 + 1.1x_4 + 0.79x_5 \leq 4 \quad \text{(cost)} \tag{2.20}$$

$$150x_1 + 185x_2 + 245x_3 + 270x_4 + 90x_5 \leq 800 \quad \text{(calories)} \tag{2.21}$$

$$x_1 + 7x_2 + 21x_3 + 8.6x_4 + 2x_5 \geq 25 \quad \text{(protein)} \tag{2.22}$$

$$0.7x_2 + 2.7x_3 + 1.3x_4 + x_5 \geq 3.5 \quad \text{(iron)} \tag{2.23}$$

with $x_i \geq 0$. We also have upper bounds on each of the foods. Therefore, there are the additional inequality constraints as determined from the last column of Table 2.1:

$$x_1 \leq 2 \tag{2.24}$$

$$x_2 \leq 3 \tag{2.25}$$

$$x_3 \leq 2 \tag{2.26}$$

$$x_4 \leq 1 \tag{2.27}$$

$$x_5 \leq 1. \tag{2.28}$$

To put this problem into standard form, we start with the inequalities (2.22) and (2.23) and convert them into equalities by subtracting positive semi-definite *surplus* variables x_6 and x_7 from the left hand sides to yield

$$x_1 + 7x_2 + 21x_3 + 8.6x_4 + 2x_5 - x_6 = 25 \tag{2.29}$$

$$0.7x_2 + 2.7x_3 + 1.3x_4 + x_5 - x_7 = 3.5. \tag{2.30}$$

To convert inequalities (2.24)-(2.28) and (2.20)-(2.21) into equalities we add positive semi-definite *slack* variables x_8 to x_{14} to give

$$x_1 + x_8 = 2 \tag{2.31}$$

$$x_2 + x_9 = 3 \tag{2.32}$$

$$x_3 + x_{10} = 2 \tag{2.33}$$

$$x_4 + x_{11} = 1 \tag{2.34}$$

$$x_5 + x_{12} = 1 \tag{2.35}$$

$$1.25x_1 + 0.9x_2 + 0.99x_3 + 1.1x_4 + 0.79x_5 + x_{13} = 4 \tag{2.36}$$

$$150x_1 + 185x_2 + 245x_3 + 270x_4 + 90x_5 + x_{14} = 800. \tag{2.37}$$

Now all the constraints have been converted into equalities. However, to start the iterations in linear programming we need an identity matrix in the partitioned part of $\mathbf{A}$. Thus we introduce non-negative *artificial* variables x_{15} and x_{16} into Eq. (2.29)-(2.30) to give

$$x_1 + 7x_2 + 21x_3 + 8.6x_4 + 2x_5 - x_6 + x_{15} = 25 \tag{2.38}$$

$$0.7x_2 + 2.7x_3 + 1.3x_4 + x_5 - x_7 + x_{16} = 3.5. \tag{2.39}$$

These nine equations can now be written in the form

$$\mathbf{Ax} = \mathbf{b} \tag{2.40}$$

where the (9×16) matrix $\mathbf{A}$ is

$$\mathbf{A} = \left[\begin{array}{ccccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{I} \\ 1.25 & 0.9 & 0.99 & 1.1 & 0.79 & 0 & 0 & \\ 150 & 185 & 245 & 270 & 90 & 0 & 0 & \\ 1 & 7 & 21 & 8.6 & 2 & -1 & 0 & \\ 0 & 0.7 & 2.7 & 1.3 & 1 & 0 & -1 & \end{array}\right] \tag{2.41}$$

where $\mathbf{I}$ is a (9×9) identity matrix, and

$$\mathbf{b}^T = [2 \quad 3 \quad 2 \quad 1 \quad 1 \quad 4 \quad 800 \quad 25 \quad 3.5]. \tag{2.42}$$

As mentioned earlier, in the formulation of the problem, the identity matrix usually appears as the last m columns of $\mathbf{A}$, but this makes no difference in computer

programming, since proper subscripts can easily be assigned. The performance index must now be changed to take into account the extra variables that have been introduced. Obviously, there is no cost associated with surplus variables and slack variables. However, the artificial variables x_{15} and x_{16} have been introduced only to provide the necessary entries for $\mathbf{A}$ to contain the identity matrix. Since the artificial variables must not appear in the final solution, then to achieve such a goal computationally, we assign to these artificial variables in the performance index arbitrarily large negative cost factors (such as -1000). Therefore, the performance index to be maximized is given by

$$\begin{aligned} I &= 100x_1 + 50x_2 + 25x_3 + 20x_4 + 35x_5 + 0x_6 + 0x_7 + 0x_8 + 0x_9 + 0x_{10} \\ &\quad +0x_{11} + 0x_{12} + 0x_{13} + 0x_{14} - 1000x_{15} - 1000x_{16}. \end{aligned} \tag{2.43}$$

Although there are now 16 variables, the computation time on a personal computer is very short, yielding results almost instantaneously. It must be realized that the programming effort is very small. The algorithm itself is easily programmed with only 64 lines of FORTRAN code, as is shown in Appendix B. More effort is required in programming the input and output subroutines to handle the input data than to program the linear programming simplex algorithm. It is important to have the input as simple as possible, and to have the output meaningfully presented, where the foods are properly identified and numbered.

```
      DIET PROBLEM WITH 5 FOODS
   5    2    2    5
 4.0 800.0 25.0 3.5
BEER, 12-OZ BOTTLE  100. 1.25 150. 1.0  0.0  2.0
PIZZA,SECTOR         50. 0.90 185. 7.0  0.7  3.0
HAMBURGER            25. 0.99 245. 21.0 2.7  2.0
HOT DOG              20. 1.10 270. 8.6  1.3  1.0
GRAPEFRUIT           35. 0.79  90. 2.0  1.0  1.0
```

Figure 2.1: Input data file for the 5 food problem

The input data file is given in Figure 2.1. The first line is a general title. The second line of input gives the number of foods, the number of slack variables required in general inequality constraints, the number of surplus variables required, and the number of slack variables required for simple upper bound constraints. The third line provides the inequality constraints in the proper order. The rest of the numbers are self-explanatory by reference to Table 2.1. It should be noted that the foods in the input file are not numbered, so that the input data can be easily changed if additional foods or different foods are to be considered.

The output from the computer program is given in Figure 2.2. It is noted that in the output, the foods are numbered so that the shadow prices can be properly identified. In this example, the computations were so fast that the printed CPU time of 0.02 s is not meaningful.

```
                DIET PROBLEM WITH 5 FOODS
            FOOD          UTILITY      COST      CAL    PROTEIN      IRON      MAX
1 BEER, 12-OZ BOTTLE       100.00      1.25    150.00      1.00      0.00      2.00
2 PIZZA,SECTOR              50.00      0.90    185.00      7.00      0.70      3.00
3 HAMBURGER                 25.00      0.99    245.00     21.00      2.70      2.00
4 HOT DOG                   20.00      1.10    270.00      8.60      1.30      1.00
5 GRAPEFRUIT                35.00      0.79     90.00      2.00      1.00      1.00
        * * * CONSTRAINTS * * *          4.00    800.00     25.00      3.50
                  SOLUTION TO THE FOOD PROBLEM:
        X(1) =  2.00      BEER, 12-OZ BOTTLE
        X(3) =  1.21      HAMBURGER
        X(2) =  0.34      PIZZA,SECTOR
         MAXIMUM UTILITY =   247.06390
        SHADOW PRICE OF X( 4) =     34.203
        SHADOW PRICE OF X( 5) =      2.896
        SHADOW PRICE OF X( 7) =     15.544
        SHADOW PRICE OF X( 8) =     15.443
        SHADOW PRICE OF X(13) =     67.645
        SHADOW PRICE OF X(15) =   1000.000
        SHADOW PRICE OF X(16) =    984.456
   CPU TIME =     0.02000 SECONDS
```

Figure 2.2: Output file for the 5 food problem

The output provides the solution and, more importantly, the shadow prices for the foods that are not in the solution. We see that maximum utility is achieved by a snack of 1.21 hamburgers, one third of a pizza sector and 2 beers. We may wonder why the other two foods were not part of the snack and here is where the shadow prices play the most useful role.

2.2.2 Interpretation of shadow prices

In Figure 2.2 we see that the shadow price for the hot dog is 34.203 and for the grapefruit the shadow price is 2.896. The shadow price is the amount by which these commodities are "underpriced" in value. This means that if the utility for each of these foods is increased (individually) by slightly more than these values, then they will be included in the solution. Let us illustrate this by changing the utility of grapefruit from 35 to 38, and rerunning the problem. The output given in Figure 2.3 shows that now grapefruit is, indeed, included in the snack. Also it is noted that the

performance index is marginally increased from 247.06 to 247.12. Also it is observed that the pizza sector is no longer part of the snack, having the shadow price 0.158.

```
                DIET PROBLEM WITH 5 FOODS
          FOOD          UTILITY      COST      CAL    PROTEIN      IRON      MAX
1 BEER, 12-OZ BOTTLE     100.00      1.25    150.00      1.00      0.00      2.00
2 PIZZA,SECTOR            50.00      0.90    185.00      7.00      0.70      3.00
3 HAMBURGER               25.00      0.99    245.00     21.00      2.70      2.00
4 HOT DOG                 20.00      1.10    270.00      8.60      1.30      1.00
5 GRAPEFRUIT              38.00      0.79     90.00      2.00      1.00      1.00
        * * * CONSTRAINTS * * *       4.00    800.00     25.00      3.50
                  SOLUTION TO THE FOOD PROBLEM:
        X(1) =  2.00     BEER, 12-OZ BOTTLE
        X(3) =  1.11     HAMBURGER
        X(5) =  0.51     GRAPEFRUIT
         MAXIMUM UTILITY =   247.11723
        SHADOW PRICE OF X( 2) =      0.158
        SHADOW PRICE OF X( 4) =     34.356
        SHADOW PRICE OF X( 7) =     15.634
        SHADOW PRICE OF X( 8) =     15.136
        SHADOW PRICE OF X(13) =     67.892
        SHADOW PRICE OF X(15) =   1000.000
        SHADOW PRICE OF X(16) =    984.366
   CPU TIME =     0.00000 SECONDS
```

Figure 2.3: Output file for the 5 food problem showing the effect of increasing the utility for grapefruit from 35 to 38

Suppose now we increase the utility for pizza from 50 to 51. The output in Figure 2.4 shows that, as expected, the pizza is back in the solution. We also note that the grapefruit is out, having now a shadow price of 0.554. In linear programming, the shadow prices provide very useful insight into the problem. Such information is not available, however, for the variables that appear in the solution. When a more realistic diet problem is considered, involving all the important nutritional values required for daily intake and a larger number of foods is used, this same computer program can be used. All that is required is to allow larger arrays by changing the dimension statements. Even with 60 foods, and including all the important nutritional requirements, such a diet problem can be solved in less than a second on a personal computer.

However, for solving problems where nonlinearities are present, linear programming can not be used directly. Linearization may be used to linearize the nonlinearities, as was done by Sauer *et al.* [36] in solving the alkylation plant optimization problem. A preferred method, however, is to use an optimization procedure that does not require any changes to the original problem. One possibility is to use the direct search optimization of Luus and Jaakola [29], which is presented in the next section.

```
                DIET PROBLEM WITH 5 FOODS
            FOOD          UTILITY      COST      CAL    PROTEIN      IRON      MAX
1 BEER, 12-OZ BOTTLE       100.00      1.25    150.00      1.00      0.00      2.00
2 PIZZA,SECTOR              51.00      0.90    185.00      7.00      0.70      3.00
3 HAMBURGER                 25.00      0.99    245.00     21.00      2.70      2.00
4 HOT DOG                   20.00      1.10    270.00      8.60      1.30      1.00
5 GRAPEFRUIT                38.00      0.79     90.00      2.00      1.00      1.00
        * * * CONSTRAINTS * * *          4.00    800.00     25.00      3.50
                  SOLUTION TO THE FOOD PROBLEM:
        X(1) =  2.00      BEER, 12-OZ BOTTLE
        X(3) =  1.21      HAMBURGER
        X(2) =  0.34      PIZZA,SECTOR
         MAXIMUM UTILITY =    247.40070
        SHADOW PRICE OF X( 4) =      35.172
        SHADOW PRICE OF X( 5) =       0.554
        SHADOW PRICE OF X( 7) =      16.114
        SHADOW PRICE OF X( 8) =      13.500
        SHADOW PRICE OF X(13) =      69.200
        SHADOW PRICE OF X(15) =    1000.000
        SHADOW PRICE OF X(16) =     983.886
   CPU TIME =      0.05000 SECONDS
```

Figure 2.4: Output file for the 5 food problem showing the effect of increasing the utility for pizza from 50 to 51

2.3 LJ optimization procedure

Let us now consider a totally different type of optimization procedure to solve steady-state optimization problems as given by Eqs. (2.1) − (2.3). For now, suppose the equality constraints in Eq. (2.2) are not present, or that they are simple in nature so that they are removed by proper rearrangement of the equations. We will deal with difficult equality constraints later. The direct search optimization procedure suggested by Luus and Jaakola [29] may then be used. Conceptually, the optimization procedure is very simple, involving only three steps:

1. Given some initial point $\mathbf{x}^*$, choose a number of random points in the n-dimensional space around this point through the equation

$$\mathbf{x} = \mathbf{x}^* + \mathbf{Dr} \tag{2.44}$$

where $\mathbf{D}$ is a diagonal matrix, where randomly chosen diagonal elements lie in the interval $[-1, +1]$, and $\mathbf{r}$ is the region size vector.

2. Check the feasibility of each such randomly chosen point with respect to the inequality constraint (2.3). For each feasible point evaluate the performance index I in Eq. (2.1), and keep the best $\mathbf{x}$-value.

3. An iteration is defined by Steps 1 and 2. At the end of each iteration, $\mathbf{x}^*$ is replaced by the best feasible $\mathbf{x}$-value obtained in step 2, and the region size vector $\mathbf{r}$

is reduced by γ through

$$\mathbf{r}^{j+1} = \gamma \mathbf{r}^{j} \tag{2.45}$$

where γ is a region contraction factor such as 0.95, and j is the iteration index. This procedure is continued for a number of iterations and the results are examined.

To increase the efficiency of this optimization procedure and to make it applicable to high-dimensional optimization problems, it was found by Luus *et al.* [27] in solving an 84-dimensional cancer chemotherapy optimization problem that the use of a multi-pass procedure (in which a relatively small number of randomly chosen points is used in each iteration) improved the computational efficiency. In the multi-pass method the three-step procedure is repeated after a given number of iterations, usually with a smaller initial region size than that in the previous pass. Recently, Luus [22] showed that the strategy of choosing the region size at the beginning of a pass to be the range over which the variables have changed during the previous pass is an effective way of increasing the efficiency.

The procedure is easy to program and, with the availability of very fast personal computers, computational inefficiency can be tolerated. One of the great advantages of the method is that no auxiliary variables are required, so that the user is closer to the problem at hand.

This optimization procedure enabled several difficult optimization problems to be solved in the original paper [29], and provided a means to solve a wide variety of problems in optimal control, such as time suboptimal control [10,11] and gave good approximation to optimal control by providing a means of obtaining the elements for the feedback gain matrix [12]. The possibility of using LJ optimization procedure for general optimal control problems was demonstrated by Bojkov *et al.* [2]. LJ optimization procedure was found very useful for stabilizing systems through shifting of poles [30] and testing stabilizability of linear systems [14]. Research was done to improve the likelihood of getting the global optimum for non-unimodal systems [38], but even without any modification, the reliability of the LJ procedure was found to be very good [39], even for the difficult bifunctional catalyst blend problem [26]. Therefore, the LJ optimization procedure could be used effectively for optimization of complex systems such as heat exchanger networks [19], a transformer design problem [37], design of structural columns in such a way that the amount of material would be minimized [3], study of plant expansion possibilities [17], and problems dealing with metallurgical processes [35]. The simplicity of the method was illustrated by the computer program given in its entirety in reference [19].

When the variables are restricted to be integers, special procedures may be necessary [16], since we cannot simply search on integer values to obtain the global optimum. Thus the scope of problems where LJ optimization procedure has been successfully applied is quite wide. In parameter estimation, Kalogerakis and Luus [8] found that by LJ optimization reliable estimates could be obtained for parameters in very few iterations, so that these estimates could then be used as starting values for the quadratically convergent Gauss-Newton method, without having to worry about

nonconvergence. In model reduction the LJ method has been found useful to match the reduced system's Nyquist plot to that of the original system [18], or used directly in time domain [40]. LJ optimization procedure has also been used successfully in model reduction in sampled-data systems [41].

2.3.1 Determination of region size

One of the aspects of LJ optimization procedure that was not considered until very recently is how to choose the region size vector **r** effectively at the beginning of the iterations, especially when a multi-pass procedure was used. This problem was recently addressed by Luus [22], by suggesting that the initial region size be determined by the extent of the variation of the variable during the previous pass, and, using this approach, rapid convergence resulted in parameter estimation problems [23,25]. With the use of reliable values for the region size at the beginning of each pass in a multi-pass run, the computational efficiency is increased quite substantially. For example, let us consider the nonseparable optimization problem introduced by Luus and Tassone [32] where we have a system described by three difference equations:

$$x_1(k+1) = \frac{x_1(k)}{1 + 0.01u_1(k)(3 + u_2(k))} \tag{2.46}$$

$$x_2(k+1) = \frac{x_2(k) + u_1(k)x_1(k+1)}{1 + u_1(k)(1 + u_2(k))} \tag{2.47}$$

$$x_3(k+1) = \frac{x_3(k)}{1 + 0.01u_2(k)(1 + u_3(k))} \tag{2.48}$$

with the initial condition

$$\mathbf{x}(0) = [2 \quad 5 \quad 7]^T. \tag{2.49}$$

The control variables are constrained by

$$0 \leq u_1(k) \leq 4 \tag{2.50}$$

$$0 \leq u_2(k) \leq 4 \tag{2.51}$$

$$0 \leq u_3(k) \leq 0.5. \tag{2.52}$$

The performance index to be minimized is

$$\begin{aligned} I \;=\; & x_1^2(P) + x_2^2(P) + x_3^2(P) + [(\sum_{k=1}^{P} x_1^2(k-1) + x_2^2(k-1) + 2u_3^2(k-1)) \\ & (\sum_{k=1}^{P} x_3^2(k-1) + 2u_1^2(k-1) + 2u_2^2(k-1))]^{\frac{1}{2}} \end{aligned} \tag{2.53}$$

where P is the number of stages.

When P is taken as 100, then we have a 300 variable optimization problem, because at each stage there are three control variables to be determined. Without the use of a reliable way of determining the region sizes over which to take the control variables, the problem is very difficult, but with the method suggested in [22], the problem was solved quite readily by the LJ optimization procedure by using 100 random points per iteration and 60 passes, each consisting of 201 iterations, to yield $I = 258.3393$. The problem is difficult because of the lack of smoothness in the optimal control policy. There is a sudden change in controls u_1 and u_2 at stage 69, as is shown in Figure 2.5.

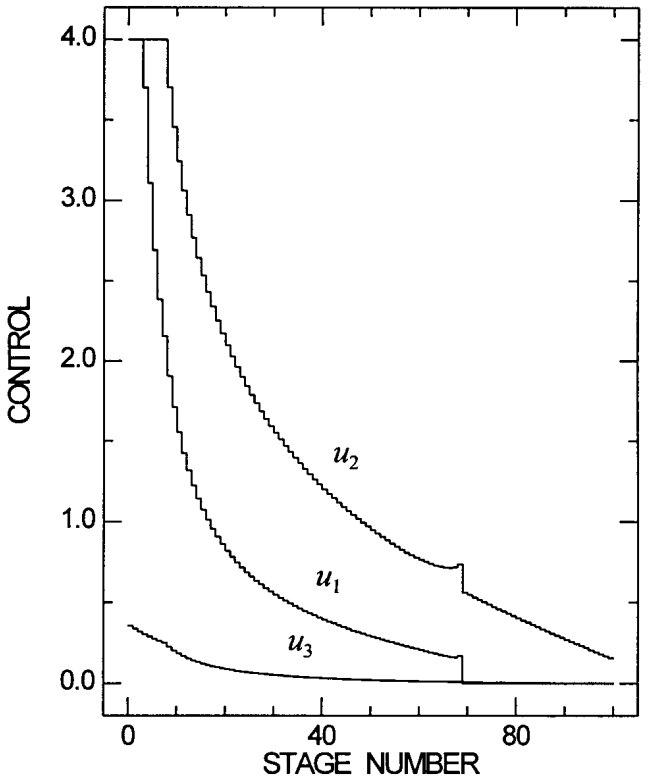

Figure 2.5: Optimal control policy, yielding I = 258.339218 obtained with IDP

Although the computational requirements appear enormous, the actual computation time was less than 20 min on a Pentium/120 personal computer [22], which corresponds to less than 6 min on the PentiumII/350. This value of the performance index is very close to the value $I = 258.33922$ obtained by use of iterative dynamic programming [21]. To solve this problem, IDP is much more efficient in spite of the nonseparability of the problem, because in IDP the problem is solved as a 3 variable problem over 100 stages, rather than a 300 variable optimization problem. We will discuss IDP and nonseparable problems in Chapter 13.

Here, the control policies obtained by IDP and LJ optimization procedure are almost identical, where a sudden change at stage 69 occurs in the control variables u_1 and u_2. Therefore, LJ optimization procedure is ideally suited for checking results obtained by other methods, especially when the optimal control policy differs from what is expected, as is the case with this particular example. There are some problems, for example, model reduction, where LJ optimization method is well suited [24]. To illustrate the procedure, we consider several examples.

2.3.2 Simple example – 5 food diet problem

Let us first consider the food problem as specified in Table 2.1 which we solved by linear programming earlier. To solve this with the LJ optimization procedure is straightforward, since the number of variables is small and no auxiliary variables are required.

By denoting the amount of each food by x_i, let us choose as starting conditions $x_i = 0.5, i = 1, \cdots, 5$, and the initial region size for each variable 2.0. Although the initial starting point is infeasible, that does not matter, because the initial starting point simply sets the centre point around which the randomly chosen points will be chosen. Let us choose the reduction factor for reducing the region sizes after every iteration $\gamma = 0.95$, 101 iterations per pass, and a maximum of 20 passes. Let us use the number of randomly chosen points per iteration as a parameter and use the 50, 100, 150, 200, and 300 for this parameter. After every pass we put the region sizes equal to the extent of the variation of that particular variable. If the variation is less than 10^{-6}, then the region size is put equal to 10^{-6}. This scheme of determining the initial region sizes for variables has been found effective in solving several optimization problems [22, 23, 25]. The listing of the computer program for this problem is given in Appendix C.

The results of the computations are given in Table 2.2. When more than 50 randomly chosen points were chosen, there was no problem in getting convergence to the maximum utility of $I = 247.06390$ with $x_1 = 2.0000, x_2 = 0.3368, x_3 = 1.2090, x_4 = 0.0000$ and $x_5 = 0.0000$. These values correspond very closely to the values obtained by linear programming and reported in Figure 2.2. It is interesting to note that with $R = 200$ randomly chosen points, convergence results already after 4 passes. The computation time for 20 passes with $R = 200$ was 5.83 s on a Pentium/120 digital computer, which is considerably greater than required by linear programming, but is still quite small.

2.3.3 Model reduction example

An important problem encountered in process control is how to reduce the complexity of a transfer function without losing essential information. Transfer functions of very high order are difficult to use, and if a transfer function of lower order gives essentially

Table 2.2: Convergence of the LJ optimization procedure showing the effect of the number of randomly chosen points per iteration R on the performance index as a function of the pass number

Pass Number	$R = 50$	$R = 100$	$R = 150$	$R = 200$	$R = 300$
1	246.91464	245.72210	246.97915	246.93550	247.03129
2	247.02599	245.83314	246.99476	247.01367	247.05779
3	247.06385	245.88798	247.06383	247.06376	247.06389
4	247.06385	245.93883	247.06388	247.06390	247.06390
5	247.06389	246.00646	247.06388	247.06390	247.06390
10	247.06389	246.61040	247.06389	247.06390	247.06390
20	247.06389	247.06390	247.06390	247.06390	247.06390

the same behavior, then it is natural to use the lower order transfer function. Many methods suggested for model reduction give unsatisfactory results [24]. The method of reducing the order of the transfer function as suggested by Luus [18] is to minimize the deviations between the reduced model and the original model in the Nyquist plot.

Let us consider the eighth-order system of Krishnamurthy and Seshadri [9]

$$F(s) = \frac{194480 + 482964s + 511812s^2 + 278376s^3 + 82402s^4 + 13285s^5 + 1086s^6 + 35s^7}{9600 + 28880s + 37492s^2 + 27470s^3 + 11870s^4 + 3017s^5 + 437s^6 + 33s^7 + s^8} \tag{2.54}$$

where we wish to get a reduced model of lower order. The zeroes (roots of the numerator) of the transfer function are $-1.0346 \pm 0.63102j$, -2.6369, -3.6345, -4.9021, -7.8014, and -9.7845. The poles (roots of the denominator) of the transfer function are $-1, -1 \pm j, -3$, $-4, -5$, -8, and -10.

Let us first choose the third order reduced model in the form

$$G(s) = \frac{20.2583(1 + x_1 s + x_2 s^2)}{(1 + x_3 s)(1 + x_4 s + x_5 s^2)} \tag{2.55}$$

where we have chosen the constant in the numerator to preserve the steady-state gain.

As was suggested in [18], we choose 100 values for the frequency by taking $\omega_1 = 0.01$ and $\omega_{i+1} = 1.1\omega_i$ to cover the frequency range from 0.01 to 125 and choose the parameters $x_i, i = 1, 2, \cdots, 5$ by minimizing the sum of squares of deviations expressed as

$$I = \sum_{i=1}^{100} [Re(F(j\omega_i)) - Re(G(j\omega_i))]^2 + [Im(F(j\omega_i)) - Im(G(j\omega_i))]^2. \tag{2.56}$$

The initial values for the five parameters are chosen from the dominant zeroes and poles of the original system to be $x_1 = 1.4090$, $x_2 = 0.680928$, $x_3 = 1.0$, $x_4 = 1.0$,

and $x_5 = 0.5$. For the region sizes, we choose for the beginning of the first pass 0.1 times the initial values of the parameters; for the second pass we choose 0.95 times this initial values for the parameters. For all subsequent passes the initial region sizes were calculated from the extent of the variation of each particular variable. The region sizes were reduced by $\gamma = 0.95$ after every iteration.

There is a trade-off between the number of random points R used per iteration and the number of passes, each consisting of 50 iterations, required for convergence to the minimum value of the sum of squares of deviations in the Nyquist plot, $I_{min} = 0.32782509$. This is shown in Table 2.3. The computation time is given for the total number of passes used for the run.

Table 2.3: Effect of the number of randomly chosen points per iteration for convergence to $I_{min} = 0.3278251$

Random Points per Iteration R	Number of Passes for Convergence	Number of Function Evaluations	CPU Time s|Passes
25	543	13575	176| 600
50	424	21200	299| 500
100	169	16900	348| 300
200	75	15000	231| 100
250	39	9750	434| 150
500	32	16000	577| 100
1000	18	18000	462| 40
2000	11	22000	471| 20
5000	9	45000	635| 10

The convergence profile tends to be quite smooth, as is shown in Figure 2.6. Although faster convergence in terms of number of passes is obtained when R is increased, the computational effort increases substantially when R exceeds 2000.

When we measure the computational effort in terms of the number of times the performance index has been calculated (number of function evaluations), we see that in the range between $R = 100$ to $R = 1000$ we need approximately 20000 function evaluations, requiring about 4 min of computation time. It is also clear that it is not advisable here to take R to be greater than 2000. In fact, it is actually most economical to take $R = 25$ and allow a large number of passes.

The resulting third order reduced model is

$$G(s) = \frac{20.2583(1 + 0.84076s + 0.55067s^2)}{(1 + 0.61769s)(1 + 0.73908s + 0.53529s^2)}. \tag{2.57}$$

Let us now consider a fourth order reduced model with the following structure

$$G(s) = \frac{20.2583(1 + x_1 s)(1 + x_2 s + x_3 s^2)}{(1 + x_4 s + x_5 s^2)(1 + x_6 s + x_7 s^2)} \tag{2.58}$$

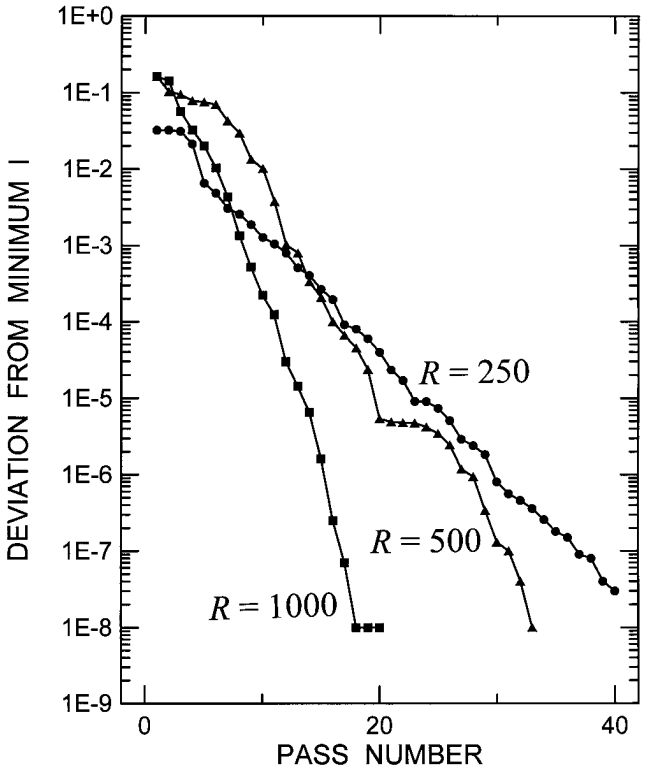

Figure 2.6: Convergence profile for the third order reduced model using $N = 50$ iterations per pass

where again we want to preserve the steady-state gain. There are now 7 parameters to be determined. The initial values are again chosen from the dominant zeroes and poles of the original transfer function: $x_1 = 0.37923$, $x_2 = 1.40900$, $x_3 = 0.68093$, $x_4 = 1.33333$, $x_5 = 0.33333$, $x_6 = 1.0$, and $x_7 = 0.5$. From the experience gained by the third order model reduction, we take a small number of random points per iteration ($R = 25$), but a large number of passes. The convergence rate is very much dependent on the tolerance ϵ used as the minimum value for the initial region size to be used to prevent collapse of the particular region. As can be seen in Figures 2.7 and 2.8, the trend is similar whether 60 iterations are used per pass, or 40 iterations are used per pass.

This leads us to use the scheme where the tolerance parameter $\epsilon = 10^{-5}$ is used for the first 2000 passes; it is then decreased to $\epsilon = 10^{-6}$ for the next 1000 passes, and decreased again for the last 1000 passes, as is shown in Figure 2.9. As is seen, convergence is obtained within 4000 passes with this adaptive choice of the tolerance parameter. The computation time on a Pentium/120 with $R = 25$ and $N = 50$ iterations per pass was 2077s, and very accurate results are obtained. The use of $R = 200$, with the use of adaptive method of specifying the tolerance parameter, yielded convergence to $I_{min} = 0.00073136556$ in 2929 passes. Computationally, however,

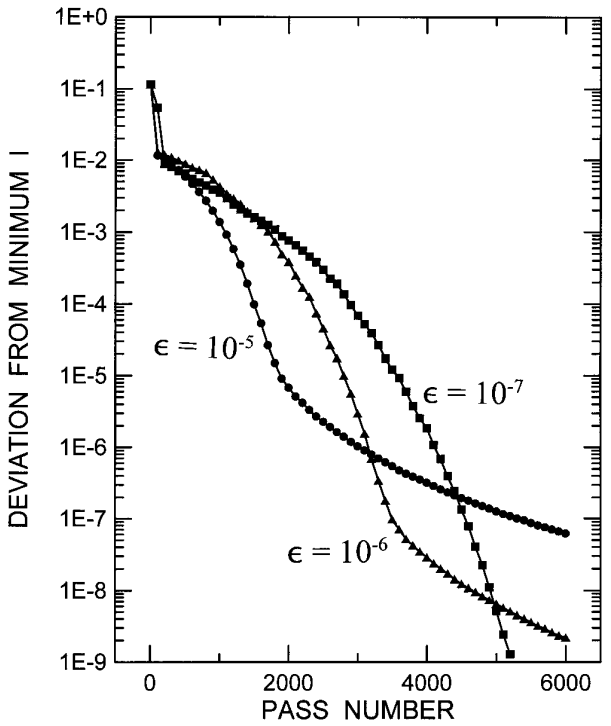

Figure 2.7: Convergence profile using $N = 60$ iterations per pass, and using the tolerance ϵ as a parameter

rather than trying to get the optimum in a small number of passes, it is better to take a relatively small number of random points R and allow a greater number of passes.

The optimum reduced fourth order model is

$$G(s) = \frac{20.2583(1 + 0.20781s)(1 + 1.39761s + 0.72546s^2)}{(1 + 1.09872s + 0.16412s^2)(1 + 1.03128s + 0.53312s^2)}. \tag{2.59}$$

The zeroes of this reduced model are $-4.811, -0.963 \pm 0.671j$, and the poles are $-5.607, -1.086, -0.967 \pm 0.970j$. It is noted that the three dominant poles of this reduced system are quite close to the three dominant poles of the original system, i.e., -1, $-1 \pm j$, but the fourth pole is closest to the sixth pole of the original system. Similar observation is made with respect to the two zeroes closest to the imaginary axis. The computer program for this problem is listed in Appendix C.

As was shown by Luus *et al.* [33], instead of carrying out the model reduction in frequency domain, we get equally good results when the optimization is done in the time domain. Suppose we use as input a unit step function. Then the output $f(t)$ of the system described by $F(s)$ is

$$f(t) = \sum_{i=1}^{n} f_i e^{\alpha_i t} \tag{2.60}$$

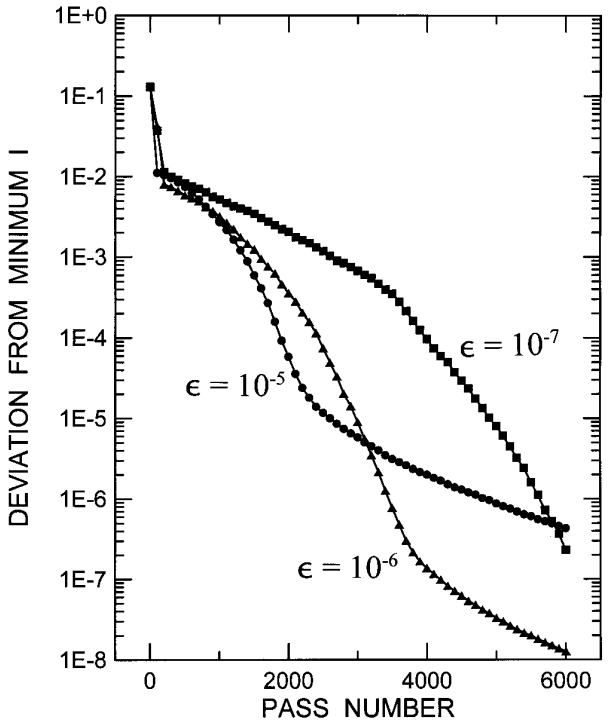

Figure 2.8: Convergence profile using $N = 40$ iterations per pass, and using ϵ as a parameter

where α_i is the i^{th} pole of the transfer function $F(s)$, and

$$f_i = \lim_{s \to \alpha_i} (s - \alpha_i) F(s). \tag{2.61}$$

The output $g(t)$ of the reduced model is

$$g(t) = \sum_{i=1}^{q} g_i e^{\beta_i t} \tag{2.62}$$

where β_i is the i^{th} pole of the transfer function $G(s)$, and

$$g_i = \lim_{s \to \beta_i} (s - \beta_i) G(s). \tag{2.63}$$

The performance index to be minimized is

$$I = \int_0^T (f(t) - g(t))^2 dt, \tag{2.64}$$

where T is sufficiently large.

By taking $T = 10$, we obtained the reduced model

$$G(s) = \frac{20.2583(1 + 0.22972s)(1 + 1.42268s + 0.71952s^2)}{(1 + 1.15241s + 0.18572s^2)(1 + 1.02491s + 0.51747s^2)}. \tag{2.65}$$

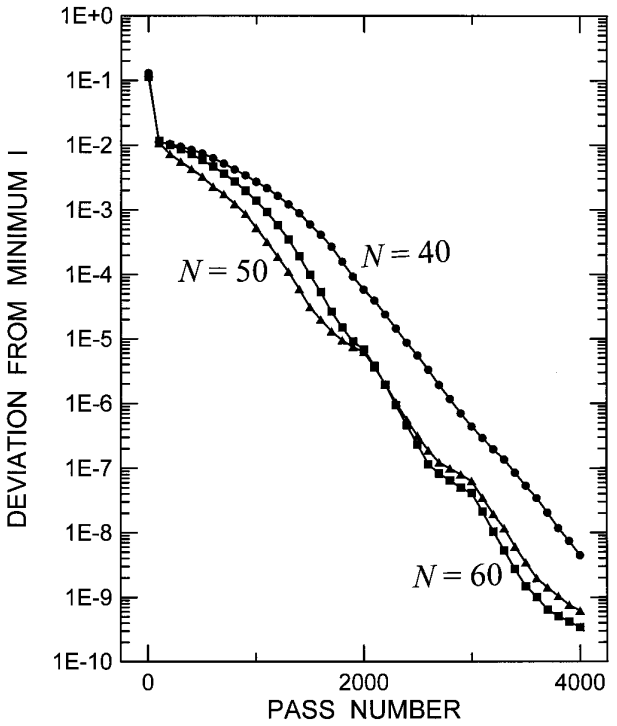

Figure 2.9: Convergence profile with $R = 25$ to $I = 0.00073136556$, showing the effect of the number of iterations per pass. For the first 2000 passes $\epsilon = 10^{-5}$, for next 1000 passes $\epsilon = 10^{-6}$, and for the last 1000 passes $\epsilon = 10^{-7}$.

The zeroes of this reduced model are -4.3531, and $-0.9886 \pm 0.6422j$, while the poles are -5.1620,-1.0431, and $-0.9990 \pm 0.9756j$.

The step responses from these two reduced models are almost indistinguishable from the step response of the original system.

2.3.4 Parameter estimation

As a practical means of parameter estimation, we consider two parameter estimation problems from Belohlav *et al.* [1].

Example 1

Let us first consider toluene hydrogenation, where the catalytic reaction

$$A \overrightarrow{\leftarrow} B \rightarrow C$$

takes place. The changes in concentration are described by three differential equations

$$\frac{dx_1}{dt} = -r_1 + r_{-1} \tag{2.66}$$

$$\frac{dx_2}{dt} = -r_{-1} + r_1 - r_2 \tag{2.67}$$

$$\frac{dx_3}{dt} = r_2 \tag{2.68}$$

where $r_1 = k_1\theta_A$, $\quad r_{-1} = k_2\theta_B$, and $r_2 = k_3\theta_B$, and the surface coverages are given by

$$\theta_A = \frac{k_4 x_1}{k_4 x_1 + x_2 + k_5 x_3}, \quad \theta_B = \frac{x_2}{k_4 x_1 + x_2 + k_5 x_3}. \tag{2.69}$$

The concentration of toluene (A) is given by x_1, of 1-methylcyclohexene (B) by x_2 and of methylcyclohexane (C) by x_3. Starting with pure toluene, the experimental measurements of the concentrations reported by Belohlav *et al.* [1] are given as the 13 data points in Table 2.4.

Table 2.4: Experimental data of Example 1 of Belohlav *et al.*[1]

Time,	Measured concentrations		
min	$\hat{x_1}$	$\hat{x_2}$	$\hat{x_3}$
15	0.695	0.312	0.001
30	0.492	0.430	0.080
45	0.276	0.575	0.151
60	0.225	0.570	0.195
75	0.163	0.575	0.224
90	0.134	0.533	0.330
120	0.064	0.462	0.471
180	0.056	0.362	0.580
240	0.041	0.211	0.747
320	0.031	0.146	0.822
360	0.022	0.080	0.898
380	0.021	0.070	0.909
400	0.019	0.073	0.908

The problem is to obtain the best estimate for the $\mathbf{k}$ in the sense of minimizing the sum of squares of deviations of the model values of the concentrations and the measured values of the concentrations at the 13 points where the data have been collected. Therefore the performance index to be minimized is

$$I = \sum_{i=1}^{13} [\mathbf{x}(t_i) - \hat{\mathbf{x}}(t_i)]^T [\mathbf{x}(t_i) - \hat{\mathbf{x}}(t_i)]. \tag{2.70}$$

By taking as an initial value for the parameter vector $\mathbf{k} = [0.25 \quad 0.25 \quad 0.25 \quad 0.25 \quad 0.25 \ ]^T$, giving $I = 6.90489$, and initial region size for the first pass for each

variable to be 0.125, convergence to the minimum performance index $I = 0.01590003$ was readily obtained, giving $\mathbf{k} = [0.02511 \quad 0.00326 \quad 0.00873 \quad 1.27258 \quad 1.24230]^T$. These parameter values differ quite substantially from those given by Belohlav *et al.* [1], namely, $\mathbf{k} = [0.023 \quad 0.005 \quad 0.011 \quad 1.9 \quad 1.8]^T$, giving $I = 0.01789155$, but the differences in the responses are difficult to detect visually. The computational details are given by Luus [25].

Example 2

Now let us consider a more demanding problem,where we have four chemical species. The experimental data, as obtained by Belohlav *et al.* [1], for methyl ester hydrogenation

$$A \xrightarrow{r_1} B \xrightarrow{r_2} C \xrightarrow{r_3} D,$$

where x_1 denotes the concentration of linolinic acid (A), x_2 denotes the concentration of linoleic acid (B), x_3 denotes the concentration of oleic acid (C), and x_4 denotes the concentration of stearic acid (D), are given in Table 2.5. The rate of change of the

Table 2.5: Experimental data of Example 2 of Belohlav *et al.*[1]

Time,	Measured concentrations			
min	$\hat{x_1}$	$\hat{x_2}$	$\hat{x_3}$	$\hat{x_4}$
0	0.1012	0.2210	0.6570	0.0208
10	0.0150	0.1064	0.6941	0.1977
14	0.0044	0.0488	0.6386	0.3058
19	0.0028	0.0242	0.5361	0.4444
24	0.0029	0.0015	0.3956	0.6055
34	0.0017	0.0005	0.2188	0.7808
69	0.0003	0.0004	0.0299	0.9680
124	0.0001	0.0002	0.0001	0.9982

concentrations of the esters is given by

$$\frac{dx_1}{dt} = -r_1 \tag{2.71}$$

$$\frac{dx_2}{dt} = r_1 - r_2 \tag{2.72}$$

$$\frac{dx_3}{dt} = r_2 - r_3 \tag{2.73}$$

$$\frac{dx_4}{dt} = r_3 \tag{2.74}$$

where

$$r_1 = k_1\theta_A, \;\; r_2 = k_2\theta_B, \;\; r_3 = k_3\theta_C.$$

Belohlav *et al.*[1] gave the following expressions for the surface coverage:

$$\theta_A = \frac{x_1}{x_1 + k_4x_2 + k_5x_3 + k_6x_4}, \tag{2.75}$$

$$\theta_B = \frac{k_4x_2}{x_1 + k_4x_2 + k_5x_3 + k_6x_4}, \tag{2.76}$$

$$\theta_C = \frac{k_5x_3}{x_1 + k_4x_2 + k_5x_3 + k_6x_4}. \tag{2.77}$$

The parameter estimation problem is to determine the values of the six parameters $k_i, i = 1, 2, \cdots, 6$ from the measurements of the concentrations of the four chemical species given in Table 2.5. Although Belohlav *et al.* [1] used a different criterion for optimization, we choose the sum of squares of deviations

$$I = \sum_{k=1}^{7}\sum_{i=1}^{4}[x_i(t_k) - \hat{x}_i(t_k)]^2. \tag{2.78}$$

The first data point is used as the initial condition for the integration of the equations; so, there are only 7 data points to consider for the parameter estimation.

For parameter estimation we used $\gamma = 0.95$ and allowed 101 iterations per pass and used widely different initial values for the parameters. For region sizes for the parameter search, for the beginning of the first pass we chose $\mathbf{r}_{in} = 0.5\mathbf{k}^{(0)}$, for the second pass we chose $\mathbf{r}_{in} = 0.25\mathbf{k}^{(1)}$, and for subsequent passes, we chose the initial region size according to the extent of the variation of the variables during the most recent pass, namely

$$r_{in,i}^{q+1} = |k_i^{q-1} - k_i^q|, \quad i = 1, 2, \cdots, 6, \tag{2.79}$$

where q denotes the pass number, as suggested by Luus [22].

To avoid region collapse, if a particular region becomes zero, because there was no change in the parameter during the pass, then we assigned arbitrarily a value 10^{-6} to the region size at the beginning of the subsequent pass.

When we applied the LJ optimization procedure, we found that convergence to widely different sets of parameter values resulted, as is shown in Table 2.6. It is clear that the relationship among the parameters k_1, k_4, k_5, and k_6 is retained, and if these are increased proportionately, the performance index is improved to the minimum value of $I = 4.5777 \times 10^{-4}$. This value of the performance index is almost 10 times lower than 4.3433×10^{-3} resulting from the parameter values reported by Belohlav *et al.* [1]. It is clear, therefore, that x_1 should not appear in the denominator of the surface coverage terms. Based on the given experimental data, therefore, the surface coverage terms can be simplified to

$$\theta_A = \frac{x_1}{k_4x_2 + k_5x_3 + x_4} \tag{2.80}$$

$$\theta_B = \frac{k_4 x_2}{k_4 x_2 + k_5 x_3 + x_4} \tag{2.81}$$

$$\theta_C = \frac{k_5 x_3}{k_4 x_2 + k_5 x_3 + x_4}, \tag{2.82}$$

where we have arbitrarily set k_6 to 1. Therefore, for this problem there are only 5 parameters to be determined.

By using this 5-parameter model, convergence to a unique set of parameters was obtained from widely different starting points. Figure 2.10 shows the convergence profile to $I = 4.5777 \times 10^{-4}$ from the starting values $k_i^{(0)} = 0.05, i = 1, 2, \cdots, 5$. This minimum value for the performance index was obtained in only 6 passes with the use of $R = 100$ randomly chosen points per iteration and in 11 passes with the use of $R = 200$. It is observed that the use of a larger number of randomly chosen points does not necessarily increase the convergence rate. In Figure 2.11, however, from the starting point $k_i^{(0)} = 1.0, i = 1, 2, \cdots, 5$, the convergence rate with $R = 200$ is faster than with $R = 100$, where in the former case 15 passes are required to get convergence to $I = 4.5777 \times 10^{-4}$. In each case the resulting values of the parameters are: $k_1 = 0.42786$, $k_2 = 0.026877$, $k_3 = 0.094579$, $k_4 = 10.0760$, and $k_5 = 0.57546$.

The close agreement of the model with the experimental data is shown in Figure 2.12, where the solid lines are obtained from integrating the equations with the optimally determined five parameters and the symbols indicate the data in Table 2.5.

Since it is puzzling why x_1 should not appear in the denominator of the surface coverage terms, it may be postulated that the presence of A does not hinder the catalytic activity of the other species and is not affected by the catalyst at all. Thus the structure of the model can be simplified by eliminating the θ_A entirely, and writing $r_1 = k_1 x_1$. With this change, only a marginally higher performance index is obtained, namely $I = 4.7506 \times 10^{-4}$ with $k_1 = 0.21093$, $k_2 = 0.027179$, $k_3 = 0.089990$, $k_4 = 9.9521$, and $k_5 = 0.61614$. With these values of the parameters with this different model structure, the relationship between the model predictions and the data is shown in Figure 2.13. As can be seen, there is a very close correspondence between this model prediction and the data as well.

2.3.5 Handling equality constraints

In many of the problems if equality constraints are present, they are simple and can be eliminated by proper substitution. For example in the optimization of the alkylation plant, there are 7 equality constraints. However, these 7 equality constraints were rearranged to reduce the dimensionality of the optimization problem from 10 to 3. This then became a very simple optimization problem for which the solution could be obtained in a fraction of a second. The question arises about what to do in cases where we have *difficult* equality constraints. Suppose that in addition to the inequality constraints in Eq. (2.3), we also have m equality constraints

$$h_i(x_1, x_2, ..., x_n) = 0, \quad i = 1, 2, ..., m \tag{2.83}$$

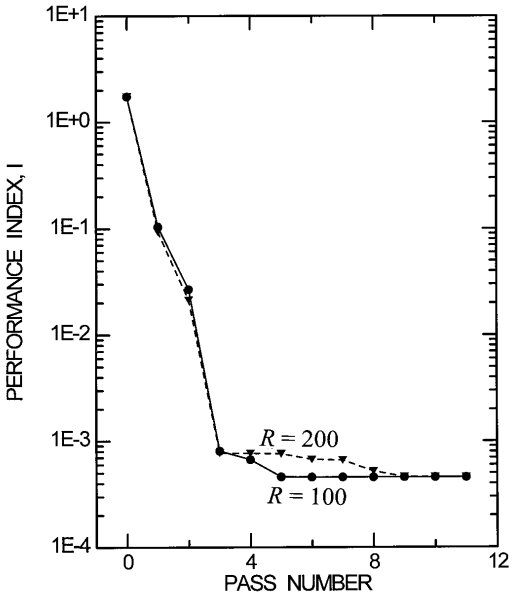

Figure 2.10: Convergence profile from initial values $k_i = 0.05, i = 1, 2, \cdots, 5$ to $I = 4.5777 \times 10^{-4}$ for Example 2

where these equality constraints are "difficult" in the sense that they can not be used to solve for some particular variable.

Although a two-pass method to deal with equality constraints [13] was effective to solve optimization problems involving recycle streams [15], the general approach for handling equality constraints with the LJ optimization procedure was not solved satisfactorily, until quite recently when Hartig *et al.* [5] showed that penalty functions can be used very effectively in direct search optimization. The work was extended by Luus and Wyrwicz [34] , who investigated the use of different types of penalty functions for steady state optimization problems. This work was extended further by Luus [20], who showed that shifting terms inside a penalty function can lead to very accurate values of the optimum. Now it appears that the use of a quadratic penalty function incorporating a shifting term is the best way of dealing with difficult equality constraints [20]. We consider the augmented performance index

$$J = I + \theta \sum_{i=1}^{m} (h_i - s_i)^2 \tag{2.84}$$

where a shifting term s_i is introduced for each equality constraint. To solve this optimization problem, the LJ optimization procedure is used in a multi-pass fashion, where, at the beginning of each pass consisting of a number of iterations, the region sizes are restored to a fraction of the sizes used at the beginning of the previous pass. The shifting terms s_i are updated after every pass simply by adjusting the values at the beginning of pass $(q + 1)$ based on the deviation of the left hand side in Eq. (2.83)

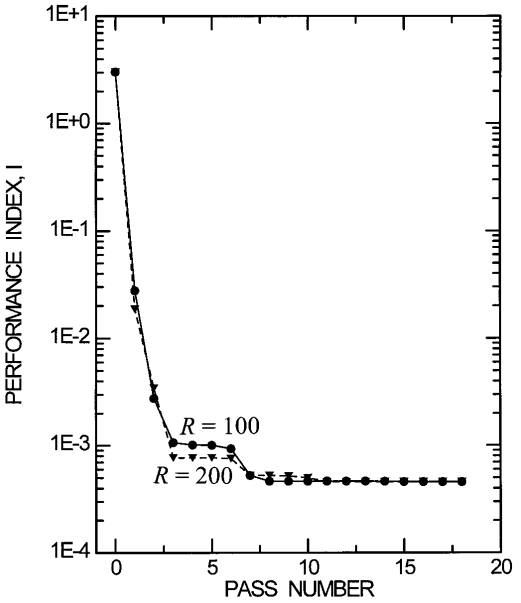

Figure 2.11: Convergence profile from initial values $k_i = 1.0, i = 1, 2, \cdots, 5$ to $I = 4.5777 \times 10^{-4}$ for Example 2

from zero, i.e.,

$$s_i^{q+1} = s_i^q - h_i, \quad i = 1, 2, ..., m \tag{2.85}$$

where q is the pass number. Upon optimization the product $-2\theta s_i$ gives the Lagrange multiplier associated with the i^{th} equality constraint, yielding useful sensitivity information. We will use this approach in Chapter 10 in dealing with equality constraints in using iterative dynamic programming (IDP) to solve an optimal control problem where the final state is specified [31]. Let us illustrate this procedure with a simple example.

Example: Geometric problem

Let us consider the geometric problem used by Luus [13] to show how to handle equality constraints in the LJ optimization procedure and then again [20] to show the use of shifting terms in optimization. The problem is to find the maximum distance from the origin to the intersection of an ellipsoid with a hyperboloid. For mathematical convenience we consider the square of the distance. Therefore it is required to maximize

$$I = x_1^2 + x_2^2 + x_3^2 \tag{2.86}$$

subject to the two equality constraints

$$h_1 = 4(x_1 - 0.5)^2 + 2(x_2 - 0.2)^2 + x_3^2 + 0.1x_1x_2 + 0.2x_2x_3 - 16 = 0 \tag{2.87}$$

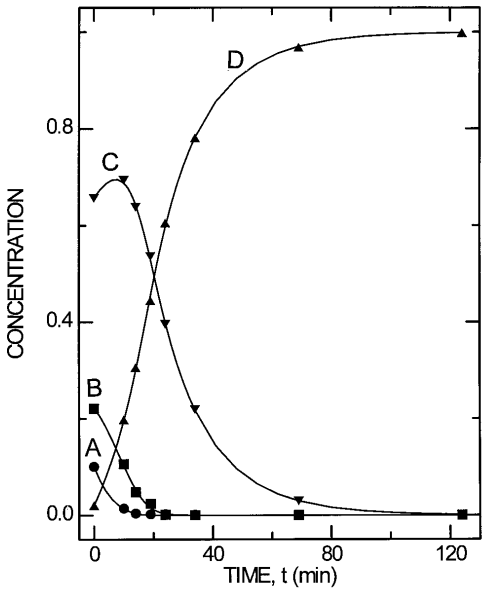

Figure 2.12: Comparison of model prediction (solid lines) to the experimental data (symbols) with $I = 4.5777 \times 10^{-4}$

and

$$h_2 = 2x_1^2 + x_2^2 - 2x_3^2 - 2 = 0. \tag{2.88}$$

We construct the augmented performance index

$$J = I - \theta[(h_1 - s_1)^2 + (h_2 - s_2)^2], \tag{2.89}$$

where θ is a positive penalty function factor and the shifting terms are initially put to zero and then, after ever pass q, are updated by

$$s_i^{q+1} = s_i^q - h_i, \qquad i = 1, 2. \tag{2.90}$$

To observe the convergence of the iterative scheme, we look at the performance index I and the norm of the constraint violation defined by

$$S = |h_1| + |h_2|. \tag{2.91}$$

To solve this problem with the LJ optimization procedure, we took the initial value for each x_i to be zero, and the region size for each variable to be 10. We used the region contraction factor of $\gamma = 0.95$, and allowed 5 iterations per pass. For the first five passes the region size was restored to 0.80 of its value at the beginning of the previous pass ($\eta = 0.80$), and for the rest of the passes the region size was determined by the extent of the variation of the variables, with 10^{-6} being the minimum value. We chose $R = 25$ random points per iteration. The listing of the computer program is given in Appendix C.

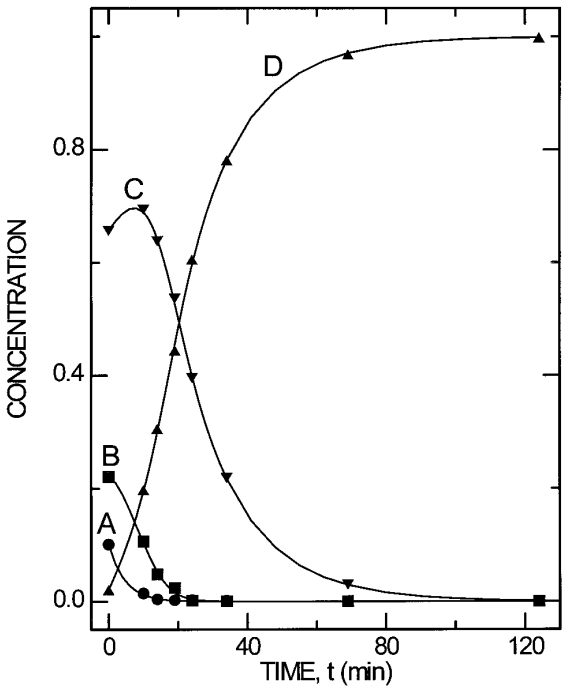

Figure 2.13: Comparison of model prediction (solid lines) to the experimental data (symbols) with $I = 4.7506 \times 10^{-4}$

The use of three different values for θ gives widely different convergence profiles, as is seen in Figure 2.14, so the convergence profile is dependent on θ, but the end result after 100 passes was the same. The value of the performance index in each case was $I = 11.67664$ and the norm of the constraint violation was less than 10^{-6}. As is shown in Table 2.7 the converged values of the shifting terms times the penalty function factor yield the same result for each case. In fact, $2\theta s_i$ gives the Lagrange multiplier for the constraint i. Thus we conclude that the Lagrange multiplier associated with the first constraint is -0.67341 and with the second constraint, 0.21106. These values are exactly the same as the Lagrange multipliers x_4 and x_5 reported in Table 1.1. As was shown by Luus [20], by choosing different starting points or different parameters in the optimization procedure, it is possible to get the other three local maxima with their associated Lagrange multipliers.

Due to its simple nature, the LJ optimization procedure can be programmed very easily. Computational experience with numerous optimization problems has shown that the method has high reliability of obtaining the global optimum; so, the LJ optimization procedure provides a very good means of obtaining the optimum for very complex problems, and can be used successfully for determining the optimal control policy for fed-batch reactors [28].

Table 2.6: Results of parameter estimation with the use of 6 parameters for Example 2, showing nonuniqueness of the optimal parameter values

Performance	Optimal values of the parameters					
index, I	k_1	k_2	k_3	k_4	k_5	k_6
4.5902×10^{-4}	2.42939	0.027024	0.094121	56.9225	3.28448	5.67431
4.5883×10^{-4}	5.43020	0.026942	0.094379	127.614	7.32090	12.6985
4.5782×10^{-4}	56.503	0.026883	0.094562	1330.377	76.0110	132.0595
4.5777×10^{-4}	5662.85	0.026877	0.094578	133360	7616.5	13235
4.5777×10^{-4}	5662800	0.026877	0.094580	133360000	7616500	13235000

Table 2.7: Results after 100 passes with different values of θ for the geometric problem

Penalty	Performance	Shifting terms		Variables		
factor, θ	index, I	s_1	s_2	x_1	x_2	x_3
0.005	11.67664	-67.34144	21.10645	0.98842	2.67366	-1.88446
0.050	11.67664	-6.73414	2.110645	0.98842	2.67366	-1.88446
0.500	11.67664	-0.673415	0.211065	0.98842	2.67366	-1.88446

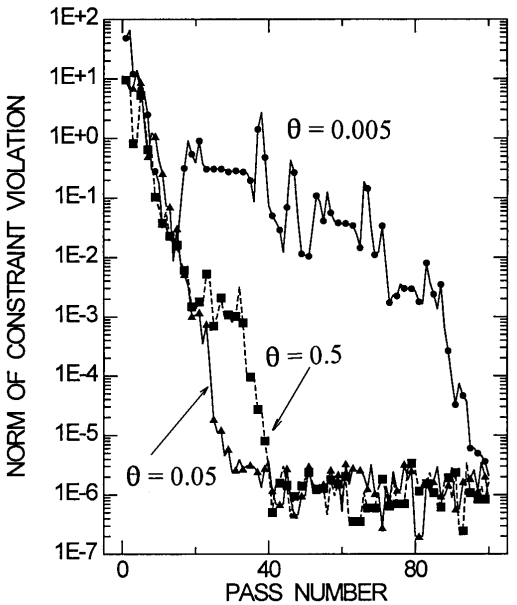

Figure 2.14: Convergence profile of the constraint violation norm with different values of the penalty function factor θ

2.4 References

[1] Belohlav, Z., Zamostny, P., Kluson, P., and Volf, J.: "Application of random-search algorithm for regression analysis of catalytic hydrogenations", *Can. J. Chem. Eng.* **75** (1997), 735-742.

[2] Bojkov, B., Hansel, R., and Luus, R.: "Application of direct search optimization to optimal control problems", *Hung. J. Ind. Chem.* **21** (1993), 177-185.

[3] Dinkoff, B., Levine, M., and Luus, R.: "Optimum linear tapering in the design of columns", *Trans. ASME* **46** (1979), 956-958.

[4] Edgar, T. and Himmelblau, D.M.: *Optimization of chemical processes*, McGraw-Hill, New York, 1988.

[5] Hartig, F., Keil, F.J., and Luus, R: "Comparison of optimization methods for a fed-batch reactor", *Hung. J. Ind. Chem.* **23** (1995), 141-148.

[6] Health and Welfare Canada: *Nutrient Value of Some Common Foods*, Supply and Services Canada, Cat. No. H58-28/1979E, 1979.

[7] Himmelblau, D.M.: *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.

[8] Kalogerakis, N. and Luus, R.: "Increasing the size of region of convergence for parameter estimation", *Proc. 1982 American Control Conf.* (1982), 358-362.

[9] Krishnamurthy, V. and Seshadri, V.: "Model reduction using the Routh stability criterion", *IEEE Trans. Automat. Control* **23** (1978), 729-731.

[10] Luus, R.: "Time-optimal control of linear systems", *Can. J. Chem. Eng.* **52** (1974), 98-102.

[11] Luus, R.: "A practical approach to time-optimal control of nonlinear systems", *Ind. Eng. Chem. Proc. Des. Develop.* **13** (1974), 405-408.

[12] Luus, R.: "Optimal control by direct search on feedback gain matrix", *Chem. Eng. Sci.* **29** (1974), 1013-1017.

[13] Luus, R.: "Two-pass method for handling difficult equality constraints in optimization", *AIChE J.* **20** (1974), 608-610.

[14] Luus, R.: "Solution of output feedback stabilization and related problems by stochastic optimization", *IEEE Trans. Auto. Contr.* **20** (1975), 820-821.

[15] Luus, R.: "Optimization of multistage recycle systems by direct search", *Can. J. Chem. Eng.* **53** (1975), 217-220.

[16] Luus, R.: "Optimization of system reliability by a new nonlinear integer programming procedure", *IEEE Trans. on Reliab.* **24** (1975), 14-16.

[17] Luus, R.: "A discussion on optimization of an alkylation process", *Int. J. Num. Methods in Eng.* **10** (1976), 1187-1190.

[18] Luus, R.: "Optimization in model reduction", *Int. J. Contr.* **32** (1980), 741-747.

[19] Luus, R.: "Optimization of heat exchanger networks", *Ind. Eng. Chem. Res.* **32** (1993), 2633-2635.

[20] Luus, R.: "Handling difficult equality constraints in direct search optimization", *Hung. J. Ind. Chem.* **24** (1996), 285-290.

[21] Luus, R.: "Application of iterative dynamic programming to optimal control of nonseparable problems", *Hung. J. Ind. Chem.* **25** (1996), 293-297.

[22] Luus, R.: "Determination of the region sizes for LJ optimization procedure", *Hung. J. Ind. Chem.* **26** (1998), 281-286.

[23] Luus, R.: "Parameter estimation of Lotka-Volterra problem by direct search optimization", *Hung. J. Ind. Chem.* **26** (1998), 287-292.

[24] Luus, R.: "Optimal reduction of linear systems", *J. of Franklin Inst.* **336** (1999), 523-532.

[25] Luus, R.: "Application of direct search optimization for parameter estimation", in *Scientific Computing in Chemical Engineering II*, Keil, Mackens, Voss, Werther (Eds.); Springer, New York, 1999, 346-353.

[26] Luus, R., Dittrich, J., and Keil, F.J.: "Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor", *Can. J. Chem. Eng.* **70** (1992), 780-785.

[27] Luus, R., Hartig, F., and Keil, F.J.: "Optimal drug scheduling of cancer chemotherapy by direct search optimization", *Hung. J. Ind. Chem.* **23** (1995), 55-58.

[28] Luus, R. and Hennessy, D.: "Optimization of fed-batch reactors by the Luus-Jaakola optimization procedure", *Ind. Eng. Chem. Res.* **38** (1999), 1948-1955.

[29] Luus, R. and Jaakola, T.H.I.: "Optimization by direct search and systematic reduction in the size of search region", *AIChE J.* **19** (1973), 760-766.

[30] Luus, R. and Mutharasan, R. : "Stabilization of linear system behavior by pole shifting", *Int. J. Control* **20** (1974), 395-405.

[31] Luus, R. and Storey, C. : "Optimal control of final state constrained systems", *Proc. IASTED Intl. Conf. Modelling, Simulation and Optimization*, Singapore, August 11-13, 1997, pp. 245-249. [32] Luus, R. and Tassone, V.: "Optimal control of nonseparable problems by iterative dynamic programming", *Proc. 42nd Canadian Chemical Engineering Conference*, Toronto, Canada, October 18-21, 1992, pp. 81-82.

[33] Luus, R., Wong, C.W., and Kostelnik, D.: "Importance of structure of the reduced model in model reduction", *Proc. IASTED Intl. Conf. Intelligent Systems and Control*, Santa Barbara, CA, October 28-30, 1999, pp. 322-326.

[34] Luus, R. and Wyrwicz, R.: "Use of penalty functions in direct search optimization", *Hung. J. Ind. Chem.* **24** (1996), 273-278.

[35] Papangelakis, V. G. and Luus, R.: "Reactor optimization in the pressure oxidation process", *Proc. Intl. Symp. Modelling, Simulation and Control of Metall. Processes* (1993), 159-171.

[36] Sauer, R.N., Colville, A.R., and Burwick, C.W.: "Computer points in the way to more profits", *Hydrocarbon Processing Petrol. Refiner* **43** (1964), 84-90.

[37] Spaans, R. and Luus, R.: "Importance of search-domain reduction in random optimization", *JOTA* **75** (1992), 635-638.

[38] Wang, B.C. and Luus, R.: "Optimization of non-unimodal systems", *Int. J. Num. Methods in Eng.* **11** (1977), 1235-1250.

[39] Wang, B.C. and Luus, R.: "Reliability of optimization procedures for obtaining global optimum", *AIChE J.* **19** (1978), 619-626.

[40] Yang, S.M. and Luus, R.: "Optimization in linear system reduction", *Electronics Letters* **19** (1983), 635-637.

[41] Yang, S.M. and Luus, R.: "A note on model reduction of digital control systems", *Int. J. Contr.* **37** (1983), 437-439.

Chapter 3

Dynamic programming

3.1 Introduction

A very powerful method for optimization of a system that can be separated into stages is *dynamic programming* developed by Bellman [2]. The main concept of this technique lies in the *principle of optimality* which can be stated as follows:

An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Many engineering systems are in the form of individual stages, or can be broken into stages, so the idea of breaking up a complex problem into simpler subproblems, so that optimization could be carried out systematically by optimizing the subproblems, was received enthusiastically. Numerous applications of the principle of optimality in dynamic programming were given by Bellman and Dreyfus [3], and there was a great deal of interest in applying dynamic programming to optimal control problems. In the 1960's many books and numerous papers were written to explore the use of dynamic programming as a means of optimization for optimal control problems. A very readable book was written by Aris [1]. Since an optimal control problem, involving optimization over a trajectory, can be broken into a sequence of time stages, it appeared that dynamic programming would be ideally suited for such problems.

To understand the principle of optimality and to see the beauty of dynamic programming we wish to present a few examples.

3.2 Examples

In order to understand and appreciate dynamic programming, we present several examples.

3.2.1 A simple optimal path problem

Let us consider a situation depicted in Figure 3.1, where there are different paths that could be taken in going from A to B. The costs for sections of the paths are shown by the numbers. Suppose we wish to find the path, always moving to the right, such that the total cost in going from A to B is minimized. For ease of discussion, the nodes are labelled by the letters $a, b, \cdots, l$.

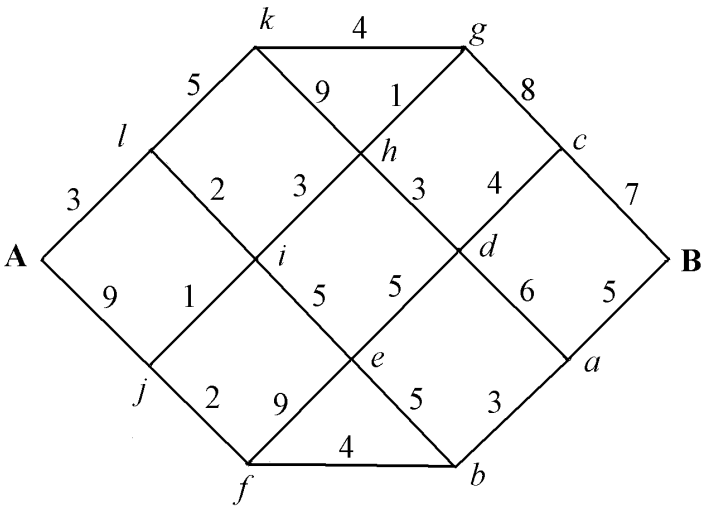

Figure 3.1: Optimal path problem

Starting at node a, we find that there is only one way to go to B and the associated cost is 5, so we associate the minimum cost 5 with node a. Similarly, there is only one way to proceed from node b and the cost is 3 plus the cost from going from node a to B. Thus we associate the minimum cost 8 with the node b. Proceeding to node c, since we are always going from left to right, we see that there is only one way to get to B and the associated cost is 7. At the node d, there are two options: one is to proceed to node c and the other is to proceed to node a. It is noted that here it does not make any difference, since the cost with each alternative is 11. Therefore we assign the minimum cost 11 to node d.

At node e there are also two ways of proceeding. We can go to either node d or to node b. In the former, the cost is $5 + 11 = 16$, and for the latter the cost is $5 + 8 = 13$. We compare these two numbers and choose the path that gives the smaller number, namely the path e to b, and associate with node e the minimum cost 13. Similarly at node f we compare the two costs $9 + 13 = 22$ and $4 + 8 = 12$, and associate the minimum cost 12 with the node f and the path f to b rather than f to e. At node g there is only one way of proceeding, so the minimum cost associated with node g

is $8 + 7 = 15$. At node h we compare the two costs $1 + 15 = 16$ and $3 + 11 = 14$, and choose the path h to d. At node i comparison of two costs gives us the minimum cost of 17 along the path i to h. At node j the minimum cost 14 is achieved by going from j to f. At the node k we compare the two costs $4 + 15 = 19$ and $9 + 14 = 23$, and choose the path k to g. At the node l we compare the two costs $5 + 19 = 24$ and $2 + 17 = 19$ and choose the path l to i which gives the minimum cost of 19.

Now we proceed to the starting point A. Here we have two ways of proceeding. For the path A to l the cost is $3 + 19 = 22$, and for the path A to j the cost is $9 + 14 = 23$. Therefore we conclude that the minimum cost in going from A to B is 22 and that there are two paths that yield this cost: A*lihdc*B and A*lihda*B.

We note here that instead of doing the minimization as a whole problem, we solve a sequence of very simple minimization problems which involve comparing two numbers at the nodes. Some of the nodes were actually "free" where no optimization was necessary. Also it is observed that it is not possible to miss the global minimum, and if there is more than one path that yields the minimum, the method captures all of them, as was illustrated here. The computational advantage of this approach over exhaustive enumeration becomes more apparent as the size of the problem increases. For example, if instead of the 3×3 grid, we would have a 10×10 grid, the approach with dynamic programming could still be easily done by hand, but exhaustive enumeration of all the possible paths would be unrealistic.

We now turn to a very similar type of a problem involving the scheduling of jobs that are to be done on a single facility in such a way that the cost would be minimized [4].

3.2.2 Job allocation problem

Let us consider a set of P jobs, called $J_1, J_2, \cdots, J_P$ to be executed on a single facility. Associated with each job J_n is a time of execution τ_n and a cost function $c_n(t)$ giving the cost associated with completing the job at time t. For simplicity, we assume that only one job is executed at a time, that there is no interruption in a job once it is started, and the facility is constantly in use. The problem is to choose the ordering of the jobs such that the total cost is minimized. The performance index is thus chosen to be

$$I = \sum_{k=1}^{P} c_k(t_k), \tag{3.1}$$

and we wish to determine the order in which the jobs are to be executed so that the performance index is minimized. For all possible orderings we need to examine $P!$ schedules, which becomes unmanageable when P is large. For this type of problem dynamic programming is well suited. We illustrate this with the case where $P = 5$ jobs. This problem is very similar to the optimal path problem discussed earlier. We picture the problem as involving 5 stages. At stage 1 we have 5 jobs and at stage 5 we have a single job to consider. We start at the last stage (stage 5) involving a

single job. There is no optimization possible at this stage. The cost of doing job i incurs a cost $c_i(\tau_i)$.

Now let us consider stage 4 where two jobs are to be considered. There are 10 possible sets of two jobs to consider. Each set has two permutations. For example, if we consider jobs J_1 and J_2, we ask whether the arrangement should be J_1 followed by J_2 denoted by J_1J_2, or whether it should be J_2J_1. For the order J_1J_2 the cost is $c_1(\tau_1) + c_2(\tau_2 + \tau_1)$ and for the order J_2J_1 the cost is $c_2(\tau_2) + c_1(\tau_1 + \tau_2)$. These costs are different since the cost of a particular job is dependent on the time when that job is finished. The two numbers are compared, and suppose that it is found that the latter arrangement gives a smaller cost. Then we know that in doing jobs J_1 and J_2, job J_2 should be done first followed by J_1. We carry out this comparison with the other 9 sets and then construct a table showing the arrangement of two jobs to yield the minimum cost, as is shown in Table 3.1.

Table 3.1: Order and minimum cost for 2 jobs

Ordered Jobs	Minimum cost
J_2J_1	$c_2(\tau_2) + c_1(\tau_2 + \tau_1)$
J_3J_1	$c_3(\tau_3) + c_1(\tau_3 + \tau_1)$
J_1J_4	$c_1(\tau_1) + c_4(\tau_4 + \tau_1)$
J_1J_5	$c_1(\tau_1) + c_5(\tau_5 + \tau_1)$
J_2J_3	$c_2(\tau_2) + c_3(\tau_2 + \tau_3)$
J_4J_2	$c_4(\tau_4) + c_2(\tau_4 + \tau_2)$
J_2J_5	$c_2(\tau_2) + c_5(\tau_2 + \tau_5)$
J_4J_3	$c_4(\tau_4) + c_3(\tau_4 + \tau_3)$
J_3J_5	$c_3(\tau_3) + c_5(\tau_3 + \tau_5)$
J_5J_4	$c_5(\tau_5) + c_4(\tau_5 + \tau_4)$

Now we proceed to stage 3 where three jobs are to be considered. There are 10 possible sets of 3 jobs taken from 5 jobs. Let us choose the set consisting of jobs J_1, J_2 and J_3. We use the information in Table 3.1 to reduce the number of calculations required to determine the order. If J_1 is last, then we know the order in which jobs J_2 and J_3 are to be done and the cost associated with these two jobs from the fifth line of Table 3.1. All that is necessary is to add the additional cost $c_1(\tau_2 + \tau_3 + \tau_1)$. Similarly, if J_2 is done last, the cost is the corresponding entry for jobs 1 and 3 from the second line of Table 3.1 plus $c_2(\tau_1 + \tau_3 + \tau_2)$; and if J_3 is done last, the cost is the corresponding entry for jobs 1 and 2 from the first line of Table 3.1 plus $c_3(\tau_1 + \tau_2 + \tau_3)$. These three costs are compared, and the arrangement that gives the minimum cost is kept. Let us suppose that of these 3 arrangements, the order $J_2J_1J_3$ gives the least cost. This entry appears on the first line of Table 3.2. This procedure is repeated

for the other 9 sets and the table is completed, showing the minimum costs for three jobs.

Table 3.2: Order and minimum cost for 3 jobs

Ordered Jobs	Minimum cost
$J_2J_1J_3$	$c_2(\tau_2) + c_1(\tau_2 + \tau_1) + c_3(\tau_1 + \tau_2 + \tau_3)$
$J_3J_1J_4$	$c_3(\tau_3) + c_1(\tau_3 + \tau_1) + c_4(\tau_1 + \tau_3 + \tau_4)$
$J_1J_4J_2$	$c_1(\tau_1) + c_4(\tau_4 + \tau_1) + c_2(\tau_1 + \tau_4 + \tau_2)$
$J_1J_5J_3$	$c_1(\tau_1) + c_5(\tau_5 + \tau_1) + c_3(\tau_1 + \tau_5 + \tau_3)$
$J_2J_3J_4$	$c_2(\tau_2) + c_3(\tau_2 + \tau_3) + c_4(\tau_2 + \tau_3 + \tau_4)$
$J_4J_2J_5$	$c_4(\tau_4) + c_2(\tau_4 + \tau_2) + c_5(\tau_4 + \tau_2 + \tau_5)$
$J_2J_5J_1$	$c_2(\tau_2) + c_5(\tau_2 + \tau_5) + c_1(\tau_2 + \tau_5 + \tau_1)$
$J_4J_3J_5$	$c_4(\tau_4) + c_3(\tau_4 + \tau_3) + c_5(\tau_4 + \tau_3 + \tau_5)$
$J_3J_5J_2$	$c_3(\tau_3) + c_5(\tau_3 + \tau_5) + c_2(\tau_3 + \tau_5 + \tau_2)$
$J_5J_4J_1$	$c_5(\tau_5) + c_4(\tau_5 + \tau_4) + c_1(\tau_5 + \tau_4 + \tau_1)$

Now we proceed to stage 2 where 4 jobs are to be considered. There are 5 possible sets of 4 jobs that can be chosen from 5 jobs. Let us choose the set consisting of jobs J_1, J_2, J_3, and J_5. We use the information in Table 3.2 to determine the order and the minimum cost for the first three jobs, and we simply add the cost of doing the last job. If J_1 is done last, then we know the order in which jobs J_2, J_3 and J_5 are to be done and the cost associated with these three jobs from Table 3.2. All that is necessary is to add the additional cost $c_1(\tau_2 + \tau_3 + \tau_5 + \tau_1)$. We proceed for the other three cases, where J_2, J_3, or J_5 is done last as before and compare the 4 cost values to give the arrangement yielding the smallest cost. Repeating the procedure with the other 4 sets provides the information that can be displayed in Table 3.3.

Table 3.3: Order and minimum cost for 4 jobs

Ordered Jobs	Minimum cost
$J_1J_4J_2J_3$	$c_1(\tau_1) + c_4(\tau_4 + \tau_1) + c_2(\tau_1 + \tau_4 + \tau_2) + c_3(\tau_1 + \tau_4 + \tau_2 + \tau_3)$
$J_1J_5J_3J_4$	$c_1(\tau_1) + c_5(\tau_5 + \tau_1) + c_3(\tau_1 + \tau_5 + \tau_3) + c_4(\tau_1 + \tau_5 + \tau_3 + \tau_4)$
$J_4J_2J_5J_1$	$c_4(\tau_4) + c_2(\tau_4 + \tau_2) + c_5(\tau_4 + \tau_2 + \tau_5) + c_1(\tau_4 + \tau_2 + \tau_5 + \tau_1)$
$J_4J_3J_5J_2$	$c_4(\tau_4) + c_3(\tau_4 + \tau_3) + c_5(\tau_4 + \tau_3 + \tau_5) + c_2(\tau_4 + \tau_3 + \tau_5 + \tau_2)$
$J_3J_5J_2J_1$	$c_3(\tau_3) + c_5(\tau_3 + \tau_5) + c_2(\tau_3 + \tau_5 + \tau_2) + c_1(\tau_3 + \tau_5 + \tau_2 + \tau_1)$

Let us now proceed to stage 1 where 5 jobs are to be considered. There is only one set to consider. We ask ourselves which job is to be done last. If J_4 is done last, then the minimum cost of doing the other 4 jobs is obtained from the last line of Table 3.3 and we simply add the cost of doing J_4, namely, $c_4(\tau_1 + \tau_2 + \tau_3 + \tau_4 + \tau_5)$. Similarly, we find from the second line of Table 3.3 the order and the minimum cost of doing the first four jobs if J_2 is done last and simply add the cost $c_2(\tau_1 + \tau_2 + \tau_3 + \tau_4 + \tau_5)$. We continue the process for when J_3 is done last, when J_4 is done last, and when J_1 is done last. These 5 values of cost are compared and the minimum of these gives us the order of the 5 jobs such that the cost is minimized.

We see in this example that the calculations proceed in a systematic way, so that at each stage the optimization involves only the addition of a single job to those that have been already optimally calculated. The procedure is ideally suited for the use of a digital computer where the same calculational procedure is used repeatedly numerous times. Instead of simply providing a numerical approach to optimization problems, dynamic programming also provides functional relationships as can be seen in the next example.

3.2.3 The stone problem

Let us illustrate the use of dynamic programming in solving the problem of how high a stone rises when tossed upwards with a velocity of v_0. Let us assume that air resistance cannot be ignored, so the equation of motion for the stone can be written as

$$\frac{dv}{dt} = -g - kv^2 \tag{3.2}$$

with the initial condition $v = v_0$ at $t = 0$. It is clear that the height is a function of the initial velocity. Let us break the trajectory into two parts: the first part is the short distance the stone travels in a short time Δt and the second part is the remainder of the trajectory

$$h(v_0) = v_0 \Delta t + h(v_0 + \frac{dv}{dt}\Delta t) + O(\Delta t^2). \tag{3.3}$$

Now let us expand the second term on the right hand side by Taylor series, retaining only the first two terms, and using Eq.(3.2) to give

$$h(v_0) = v_0 \Delta t + h(v_0) - [(g + kv_0^2)\Delta t]\frac{dh(v_0)}{dv_0} + O(\Delta t^2). \tag{3.4}$$

When we rearrange Eq. (3.4) and divide by Δt, we get

$$\frac{dh(v_0)}{dv_0} = \frac{v_0}{g + kv_0^2} + O(\Delta t). \tag{3.5}$$

Now we let $\Delta t \to 0$ to yield a first order ordinary differential equation

$$\frac{dh(v_0)}{dv_0} = \frac{v_0}{g + kv_0^2}, \tag{3.6}$$

which can be immediately integrated with the given initial condition to give

$$h(v_0) = \frac{1}{2k}\ln(1 + \frac{kv_0^2}{g}). \tag{3.7}$$

When k is negligibly small then we see that $h(v_0) \to v_0^2/2g$, a result which can be obtained by equating the kinetic energy at the initial time and the potential energy at the maximum height.

3.2.4 Simple optimal control problem

Let us consider a rather simple optimal control problem where a linear discrete scalar system is given in the form

$$x(k+1) = ax(k) + bu(k), \qquad k = 0, 1, 2, \cdots, P-1, \tag{3.8}$$

where the initial state $x(0)$ is given, and the final state $x(P) = 0$. We assume there are no constraints placed on the control $u(k)$. The optimal control problem is to choose $u(k), k = 0, 1, 2, \cdots, P-1$ so that the performance index

$$I[x(0), P] = \sum_{k=1}^{P}[x^2(k) + u^2(k-1)] \tag{3.9}$$

is minimized. Here we have shown the explicit dependence of the performance index on the initial state and the number of steps P. Naturally the performance index is also a function of the control policy which has to be chosen for minimization. To solve this problem by dynamic programming we first consider a single stage problem and then progressively increase the number of stages until the posed problem is solved.

P = 1

When we have a single stage problem, the performance index is

$$I[x(0), 1] = x^2(1) + u^2(0) \tag{3.10}$$

and no minimization is possible, since the specification of the final condition requires that we use

$$u(0) = \frac{x(1) - ax(0)}{b} = -\frac{a}{b}x(0). \tag{3.11}$$

The value of the performance index is

$$I[x(0), 1] = \frac{a^2}{b^2}x^2(0). \tag{3.12}$$

Next we consider $P = 2$ stages.

P = 2

With two stages we start at $x(0)$ proceed to $x(1)$ and then go to $x(2) = 0$. The performance index to be minimized is

$$I[x(0), 2] = x^2(1) + u^2(0) + x^2(2) + u^2(1) = x^2(1) + u^2(0) + u^2(1). \tag{3.13}$$

Once we get to $x(1)$, we already know how to get to $x(2) = 0$, because that is a 1-stage problem that was solved above. We therefore have

$$u(1) = -\frac{a}{b}x(1). \tag{3.14}$$

The performance index becomes

$$I[x(0), 2] = (1 + \frac{a^2}{b^2})x^2(1) + u^2(0). \tag{3.15}$$

By using Eq. (3.8) with $k = 0$, we can rewrite the performance index as an explicit function of $u(0)$ only, namely,

$$I[x(0), 2] = (1 + \frac{a^2}{b^2})[ax(0) + bu(0)]^2 + u^2(0). \tag{3.16}$$

Differentiating Eq. (3.16) with respect to u(0) gives

$$\frac{dI[x(0), 2]}{du(0)} = 2b(1 + \frac{a^2}{b^2})[ax(0) + bu(0)] + 2u(0). \tag{3.17}$$

Since the second derivative of the performance index is positive, the choice of $u(0)$ to minimize the performance index is obtained from the stationary condition that yields

$$u(0) = -(\frac{ab + (a^3/b)}{1 + a^2 + b^2})x(0). \tag{3.18}$$

We are now ready to consider $P = 3$ stages.

P = 3

The pattern for calculations becomes clear with the addition of one more stage. With $P = 3$, the performance index is

$$I[x(0), 3] = x^2(1) + u^2(0) + x^2(2) + u^2(1) + u^2(2), \tag{3.19}$$

since $x(3) = 0$. Here we know that the optimal choices for $u(2)$ and $u(1)$, as determined before, are

$$u(2) = -\frac{a}{b}x(2) \tag{3.20}$$

and

$$u(1) = -(\frac{ab + (a^3/b)}{1 + a^2 + b^2})x(1). \tag{3.21}$$

The performance index can now be expressed as an explicit function of $u(0)$, and the process can be repeated.

Although analytically this process is tedious after a while, numerically such optimization can be done very easily. By using dynamic programming, the solution of P simultaneous equations is replaced by a procedure which handles one single equation, but does it P times. Thus parallel computation is replaced by serial computation.

3.2.5 Linear optimal control problem

As a further example we consider a linear optimal control problem where the system is described by

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Psi}\mathbf{u}(k), \qquad k = 0, 1, \cdots, P-1, \tag{3.22}$$

where $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are constant matrices, and the initial condition $\mathbf{x}(0)$ is given. The performance index is assumed to have the quadratic form

$$I[\mathbf{x}(0), P] = \sum_{k=1}^{P} [\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k-1)\mathbf{R}\mathbf{u}(k-1)] \tag{3.23}$$

where $\mathbf{R}$ is a positive definite symmetric matrix and $\mathbf{Q}$ is a positive semi-definite symmetric matrix. Let us define

$$I^0[\mathbf{x}(0), P] = \min I[\mathbf{x}(0), P] \tag{3.24}$$

such that $I^0[\mathbf{x}(0), P]$ is the optimal value of the performance index over P stages of control. Thus

$$I^0[\mathbf{x}(0), P] = \min_{\mathbf{u}(0)} \min_{\mathbf{u}(1)} \cdots \min_{\mathbf{u}(P-1)} \sum_{k=1}^{P} [\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k-1)\mathbf{R}\mathbf{u}(k-1)]. \tag{3.25}$$

We can use the principle of optimality in dynamic programming to rewrite the expression as

$$I^0[\mathbf{x}(0), P] = \min_{\mathbf{u}(0)} [\mathbf{x}^T(1)\mathbf{Q}\mathbf{x}(1) + \mathbf{u}^T(0)\mathbf{R}\mathbf{u}(0) + I^0[\mathbf{x}(1), P-1]], \tag{3.26}$$

which provides a recurrence relationship of dynamic programming.

Let us assume that the optimum performance index can be written as

$$I^0[\mathbf{x}(0), P] = \mathbf{x}(0)^T \mathbf{J}_P \mathbf{x}(0) \tag{3.27}$$

where $\mathbf{J}_P$ is a positive semi-definite symmetric matrix. Then

$$I^0[\mathbf{x}(1), P-1] = \mathbf{x}(1)^T \mathbf{J}_{P-1} \mathbf{x}(1) \tag{3.28}$$

and Eq. (3.26) becomes

$$I^0[\mathbf{x}(0), P] = \min_{\mathbf{u}(0)} [\mathbf{x}^T(1)(\mathbf{Q} + \mathbf{J}_{P-1})\mathbf{x}(1) + \mathbf{u}^T(0)\mathbf{R}\mathbf{u}(0)], \tag{3.29}$$

where $\mathbf{x}(1)$ is obtained from Eq. (3.22) as an explicit function of $\mathbf{x}(0)$ and $\mathbf{u}(0)$. As is shown in [4], after some mathematical manipulations we get the optimal control policy

$$\mathbf{u}^0(k) = -\mathbf{K}_{P-k}\mathbf{x}(k), \tag{3.30}$$

where the feedback gain matrix $\mathbf{K}_{P-k}$ is obtained by solving recursively

$$\mathbf{K}_{P-k} = [\mathbf{\Psi}^T(\mathbf{Q} + \mathbf{J}_{P-k-1})\mathbf{\Psi} + \mathbf{R}]^{-1}\mathbf{\Psi}^T(\mathbf{Q} + \mathbf{J}_{P-k-1})\mathbf{\Phi} \tag{3.31}$$

and

$$\mathbf{J}_{P-k} = \mathbf{\Phi}^T(\mathbf{Q} + \mathbf{J}_{P-k-1})(\mathbf{\Phi} - \mathbf{\Psi}\mathbf{K}_{P-k}) \tag{3.32}$$

with the initial condition $\mathbf{J}_0 = \mathbf{0}$. Therefore, dynamic programming can be used for optimal control of high dimensional systems if the state equation is linear, the performance index is quadratic, and the control is unbounded. For general nonlinear optimal control problems, where the solution can be obtained only numerically, there are some inherent difficulties in the use of dynamic programming, as will be pointed out in Section 3.3.

3.2.6 Cross-current extraction system

Let us consider the liquid-liquid cross-current extraction system with immiscible solvents of a single solute, as used by Lapidus and Luus [4] to illustrate optimization techniques. The system consists of three stages, as is shown in Figure 3.2. The solvent flow rate in the raffinate stream is denoted by q, and the solvent flow rate in the extract streams entering and leaving stage k is denoted by $w(k)$. The solute concentration leaving stage k in raffinate is $x(k)$, and the solute concentration leaving stage k in extract is $y(k)$.

Mass balance around stage k yields

$$x(k) = x(k-1) - u(k)y(k), \qquad k = 1, 2, 3, \tag{3.33}$$

where $u(k) = w(k)/q$. We assume equilibrium between the raffinate and extract phases at each stage, and that the solute concentrations in the two phases can be expressed through the linear relationship

$$y(k) = \alpha x(k). \tag{3.34}$$

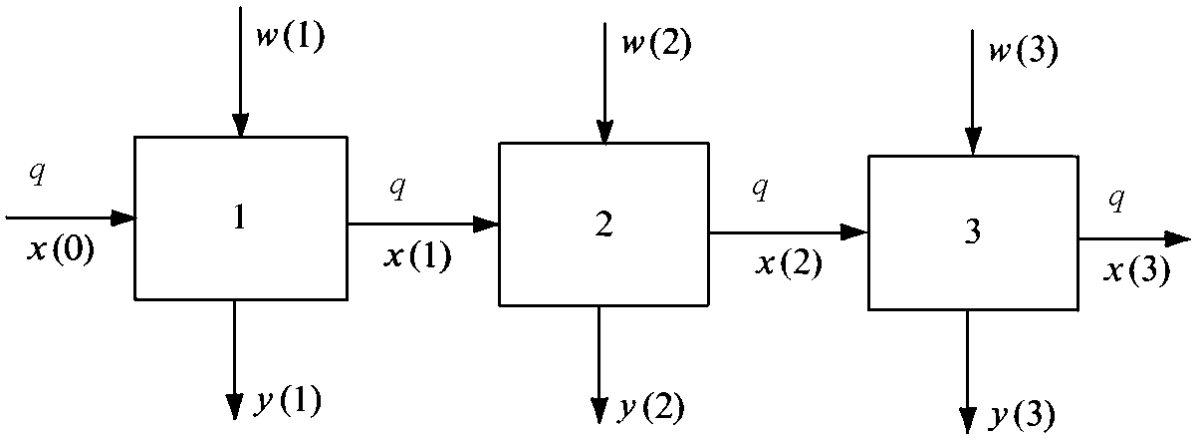

Figure 3.2: Cross-current extraction system

The state equation can then be written as

$$x(k+1) = \frac{x(k)}{1+\alpha u(k+1)}, \qquad k = 0, 1, 2. \tag{3.35}$$

We would like to choose the flow rates $w(1), w(2)$, and $w(3)$, or in normalized form $u(1), u(2)$, and $u(3)$, so that we remove the maximum amount of solute from the raffinate stream. However, there is a cost associated with the use of the extract solvent. Therefore, the performance index to be maximized is chosen as

$$I[x(0), 3] = [x(0) - x(3)] - \theta[u(1) + u(2) + u(3)], \tag{3.36}$$

where θ is a cost factor associated with the use of extract solvent. This optimization problem is well suited for dynamic programming. When there are P stages, then the performance index becomes

$$I[x(0), P] = [x(0) - x(P)] - \theta \sum_{i=1}^{P} u(i). \tag{3.37}$$

As we did with the simple optimal control problem earlier, we solve this optimization problem by first considering $P = 1$ and then increasing P until we reach $P = 3$.

P = 1

When we have a single stage, the performance index is

$$I[x(0), 1] = [x(0) - x(1)] - \theta u(1) \tag{3.38}$$

where

$$x(1) = \frac{x(0)}{1+\alpha u(1)}. \tag{3.39}$$

The maximum value of the performance index is thus

$$I^0[x(0),1] = \max_{u(1)}[u(1)(\frac{\alpha x(0)}{1+\alpha u(1)} - \theta)]. \tag{3.40}$$

By taking the derivative of the expression with respect to $u(1)$ and setting it to zero, we get

$$\frac{\alpha x(0) - \theta(1+\alpha u(1))^2}{(1+\alpha u(1))^2} = 0, \tag{3.41}$$

so that the optimal value for $u(1)$ is

$$u^0(1) = [\frac{x(0)}{\alpha\theta}]^{\frac{1}{2}} - \frac{1}{\alpha}. \tag{3.42}$$

From Eq. (3.39) it follows that

$$x(1) = [\frac{\theta x(0)}{\alpha}]^{\frac{1}{2}} \tag{3.43}$$

and the maximum performance index with the single stage is

$$I^0[x(0),1] = x(0) - 2[\frac{\theta x(0)}{\alpha}]^{\frac{1}{2}} + \frac{\theta}{\alpha}. \tag{3.44}$$

P = 2

From the principle of optimality in dynamic programming, we can write the maximum value of the performance index as

$$I^0[x(0),2] = \max_{u(1)}[u(1)(\frac{\alpha x(0)}{1+\alpha u(1)} - \theta) + I^0[x(1),1]]. \tag{3.45}$$

However, we have already calculated the optimum value of the performance index for the single stage. All we have to do is to change $x(0)$ to $x(1)$ in Eq. (3.44). Thus

$$I^0[x(1),1] = x(1) - 2[\frac{\theta x(1)}{\alpha}]^{\frac{1}{2}} + \frac{\theta}{\alpha}. \tag{3.46}$$

Therefore,

$$I^0[x(0),2] = \max_{u(1)}[u(1)(\frac{\alpha x(0)}{1+\alpha u(1)} - \theta) + \frac{x(0)}{1+\alpha u(1)} - 2[\frac{\theta x(0)}{\alpha(1+\alpha u(1))}]^{\frac{1}{2}} + \frac{\theta}{\alpha}]. \tag{3.47}$$

Differentiating the expression with respect to $u(1)$ and putting the result equal to zero gives

$$-\theta(1+\alpha u(1))^2 + \theta x(0)[\frac{\alpha(1+\alpha u(1))}{\theta x(0)}]^{\frac{1}{2}} = 0, \tag{3.48}$$

so that the optimal value for $u(1)$ is

$$u^0(1) = [\frac{x(0)}{\alpha^2\theta}]^{\frac{1}{3}} - \frac{1}{\alpha} \tag{3.49}$$

and

$$x(1) = [\frac{x^2(0)\theta}{\alpha}]^{\frac{1}{3}}. \tag{3.50}$$

The optimal performance index with $P = 2$ is

$$I^0[x(0), 2] = x(0) - 3[\frac{\theta^2 x(0)}{\alpha^2}]^{\frac{1}{3}} + \frac{2\theta}{\alpha}. \tag{3.51}$$

At this time it is interesting to note that

$$u^0(2) = [\frac{x(1)}{\alpha\theta}]^{\frac{1}{2}} - \frac{1}{\alpha} = [[\frac{x^2(0)\theta}{\alpha}]^{\frac{1}{3}} \frac{1}{\alpha\theta}]^{\frac{1}{2}} - \frac{1}{\alpha} = u^0(1). \tag{3.52}$$

Therefore, the optimal policy is to distribute the extract solvent in equal quantities in each stage.

P = 3

The optimal performance index with $P = 3$ is

$$I^0[x(0), 3] = \max_{u(1)}[u(1)(\frac{\alpha x(0)}{1 + \alpha u(1)} - \theta) + I^0[x(1), 2]], \tag{3.53}$$

where the last term that has been previously calculated is

$$I^0[x(1), 2] = x(1) - 3[\frac{\theta^2 x(1)}{\alpha^2}]^{\frac{1}{3}} + \frac{2\theta}{\alpha}. \tag{3.54}$$

After substituting Eq. (3.54) into Eq. (3.53) and differentiating the expression with respect to $u(1)$, we get for the stationary condition

$$\theta(1 + \alpha u(1))^2 = \frac{\theta^2 x(0)}{\alpha}[\frac{\theta^2 x(0)}{\alpha^2(1 + \alpha u(1))}]^{-\frac{2}{3}}, \tag{3.55}$$

so that

$$1 + \alpha u(1) = [\frac{x(0)\alpha}{\theta}]^{\frac{1}{4}}. \tag{3.56}$$

Therefore, the optimal value for control to the first stage is

$$u^0(1) = [\frac{x(0)}{\alpha^3\theta}]^{\frac{1}{4}} - \frac{1}{\alpha}, \tag{3.57}$$

and the solute concentration in the raffinate stream leaving this stage is

$$x(1) = [\frac{x^3(0)\theta}{\alpha}]^{\frac{1}{4}}. \tag{3.58}$$

The optimal performance index with $P = 3$ is

$$I^0[x(0), 3] = x(0) - 4[\frac{\theta^3 x(0)}{\alpha^3}]^{\frac{1}{4}} + \frac{3\theta}{\alpha}. \tag{3.59}$$

It is easy to show that the optimal control policy is

$$u^0(1) = u^0(2) = u^0(3), \tag{3.60}$$

showing that the extract solvent should be distributed in equal quantities in each stage.

3.3 Limitations of dynamic programming

Although dynamic programming has been successfully applied to some simple optimal control problems, one of the greatest problems in using dynamic programming for optimal control of nonlinear systems, however, is the interpolation problem encountered when the trajectory from a grid point does not reach exactly the grid point at the next stage [4]. This interpolation difficulty coupled with the dimensionality restriction and the requirement of a very large number of grid points limits the use of dynamic programming to only very simple optimal control problems. The limitations imposed by the "*curse of dimensionality*" and the "*menace of the expanding grid*" for solving optimal control problems kept dynamic programming from being used for practical types of optimal control problems.

3.4 References

[1] Aris, R.: *Discrete dynamic programming*, Blaisdell Publishing Co., New York, 1964.

[2] Bellman, R.E.: *Dynamic programming*, Princeton University Press, 1957.

[3] Bellman, R. and Dreyfus, S.: *Applied dynamic programming*, Princeton University Press, 1962.

[4] Lapidus, L. and Luus, R.: *Optimal control of engineering processes*, Blaisdell, Waltham, Mass., 1967.

Chapter 4

Iterative dynamic programming

4.1 Introduction

We consider the continuous dynamic system described by the vector differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{4.1}$$

with the initial state $\mathbf{x}(0)$ given, where $\mathbf{x}$ is an $(n \times 1)$ state vector and $\mathbf{u}$ is an $(m \times 1)$ control vector bounded by

$$\alpha_j \leq u_j(t) \leq \beta_j, \quad j = 1, 2, \cdots, m. \tag{4.2}$$

For the performance index we choose a scalar function of special form to allow the discussion of the optimization procedure:

$$I[\mathbf{x}(0), t_f] = \Psi(\mathbf{x}(t_f)) + \int_0^{t_f} \phi(\mathbf{x},\mathbf{u}, t)dt, \tag{4.3}$$

where the final time t_f is specified. The optimal control problem is to choose the control policy $\mathbf{u}(t)$ in the time interval $0 \leq t < t_f$ such that this performance index is minimized. We may have also state constraints, but for simplicity we shall leave these for later consideration in Chapter 11.

The initial ideas on iterative dynamic programming were developed and tested by Luus [5] and then refined [4] to make the computational procedure much more efficient. The fundamental ideas about using dynamic programming are based on the optimal path problem considered in Chapter 3, and are adapted to the solution of the optimal control problem.

4.2 Construction of time stages

To set up the problem into a sequence of stages, as required for dynamic programming, we may approximate the optimal control problem by requiring a piecewise constant control policy instead of a continuous control policy, over P stages, each of length L, so that

$$L = \frac{t_f}{P}. \tag{4.4}$$

Then the performance index is approximated by

$$I[\mathbf{x}(0), P] = \Psi(\mathbf{x}(t_f)) + \sum_{k=1}^{P} \int_{t_{k-1}}^{t_k} \phi(\mathbf{x}(t), \mathbf{u}(k-1), t)dt, \tag{4.5}$$

where t_{k-1} is the time at the beginning of the stage k, and $\mathbf{u}(k-1)$ is the constant control in the time interval $t_{k-1} \leq t < t_k$, and we can consider the system at the grid points set up at these P stages. We may also use a piecewise linear approximation and the stages do not necessarily have to be of equal length. These ideas are developed in later chapters. At present we wish to keep everything as simple as possible.

4.3 Construction of grid for x

Each component x_i of the state vector $\mathbf{x}$ is allowed to take on N values over a region s_i at each time stage, with the exception of the first stage, which is taken as the specified initial condition. Thus there are N^n grid points at stages $2, 3, \cdots, P$. The centre point of the grid is taken as the best value for $\mathbf{x}$ obtained from the previous iteration.

4.4 Allowable values for control

Each component u_j of the control vector $\mathbf{u}$ is allowed to take M values over a region r_j at each stage. The centre point is taken as the best value from the previous iteration. If the constraints in Equation (4.2) are violated for the component u_j, the clipping technique is used to substitute for that value the upper or lower bound. Thus there are M^m allowable values for the control vector at each of the grid points for $\mathbf{x}$.

4.5 First iteration

From the given initial condition $\mathbf{x}(0)$ and the constraints specified by Eq. (4.2) we can choose the centre point for the $\mathbf{x}$-grid and the allowable range for control, and also the regions s_i and r_j. The example will show how this choice may be made.

4.5.1 Stage P

Let us start the calculations at stage P, which corresponds to the time interval $t_f - L \leq t < t_f$. For each $\mathbf{x}$-grid point evaluate M^m values of the performance index:

$$I[\mathbf{x}(t_f - L), 1] = \Psi(\mathbf{x}(t_f)) + \int_{t_f - L}^{t_f} \phi(\mathbf{x}(t), \mathbf{u}(P-1), t)dt, \tag{4.6}$$

where each of the M^m allowable values of control are used for $\mathbf{u}(P-1)$ in turn. Compare these M^m values of the performance index and choose the particular value of $\mathbf{u}(P-1)$ that gives the minimum value. This is the best control to use at that particular $\mathbf{x}$-grid point. When we continue for the N^n grid points, we know the optimal control policy to use at the last stage for each of the grid points.

4.5.2 Stage $P-1$

Let us now step backward to stage $P-1$ corresponding to the time interval $t_f - 2L \leq t < t_f - L$. For each grid point, we again consider M^m allowable values for control. However, when we integrate Eq. (4.1) from $t_f - 2L$ to $t_f - L$, it is unlikely that the state $\mathbf{x}(t_f - L)$ will be exactly one of the grid points at stage P. This problem of not hitting a grid point exactly is illustrated in Figure 4.1, where for simplicity we have taken $n = 2, N = 5, m = 1$, and M = 4. Therefore the grid consists of a (5×5) matrix. At the grid point (2,3) of stage $P-1$ we have shown 4 trajectories to stage P, corresponding to the use of the four allowable values of control, namely $u = a, b, c$, and d. None of these trajectories hits a grid point at stage P.

To continue integration to the final time t_f, we take the optimal control policy corresponding to the grid point that is *closest* to the state $\mathbf{x}(t_f - L)$. As has been shown by DeTremblay and Luus [3], this gives a good approximation if a sufficiently large number of grid points and of allowable values for control are used. As is shown in Figure 4.1, to continue the integration for the first trajectory, we use the optimal control at stage P corresponding to the grid point (2,3); to continue the second trajectory corresponding to $u = b$, we use the best control policy at (3,3); for $u = c$ we use (5,5); and to continue the trajectory corresponding to $u = d$, we use the optimal control policy established for the grid point (4,2) at stage P. At time t_f, then, we have four values for the performance index to compare and we select the control policy that gives the minimum value. Therefore, the control policy for the grid point (2,3) at stage $P-1$ is established. This is continued for the remaining 24 grid points to finish the calculations for stage $P-1$.

4.5.3 Continuation in backward direction

We proceed in this manner with stages $P-2$, $P-3$, $\cdots$, etc., until stage 1 is reached. Stage 1 corresponds to the time interval $0 \leq t < L$ and the grid consists only of the initial condition $\mathbf{x}(0)$. At this stage we compare the M^m values of the

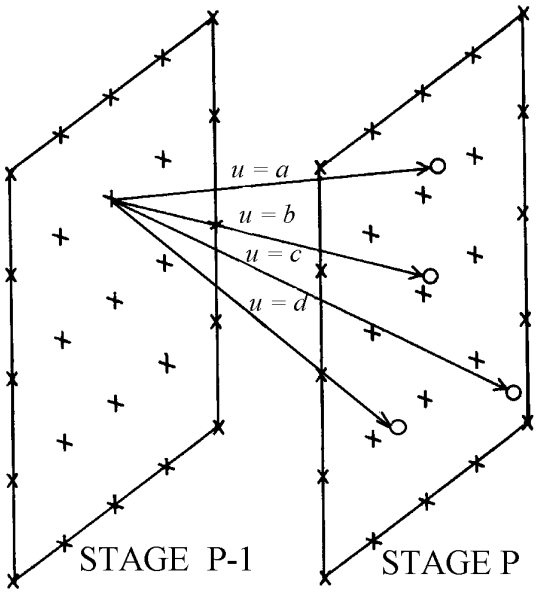

Figure 4.1: Illustration of the difficulty of reaching the grid points by assigning 4 values for control

performance index and pick the control policy that gives the minimum value. This finishes the first iteration. Even if a reasonably large number of grid points and allowable values for control are chosen, the optimal control policy obtained is quite far from the global optimal solution. Therefore, it is necessary to improve the control policy obtained after the first iteration, and we proceed to the main part of the optimization procedure.

4.6 Iterations with systematic reduction in region size

The optimal trajectory from the first iteration provides the centre point for the **x**-grid at each stage, and the optimal control policy from the first iteration gives the central value for the allowable values for control at each stage. The corresponding regions s_i and r_j are contracted by a small amount to provide a finer resolution, and the procedure is continued for a number of iterations. When this procedure is carried out for a sufficiently large number of iterations, it is expected that convergence to the optimal control policy is obtained with sufficient accuracy. To illustrate and test this procedure, we take a simple example.

4.7 Example

Let us consider the continuous stirred tank reactor (CSTR) as outlined in Chapter 1. The system is described by

$$\frac{dx_1}{dt} = -(2+u)(x_1+0.25) + (x_2+0.5)\exp[\frac{25x_1}{x_1+2}] \tag{4.7}$$

$$\frac{dx_2}{dt} = 0.5 - x_2 - (x_2+0.5)\exp[\frac{25x_1}{x_1+2}] \tag{4.8}$$

where x_1 represents deviation from dimensionless steady-state temperature and x_2 represents deviation from dimensionless steady-state concentration. In the present work, the initial conditions $x_1(0) = 0.09$ and $x_2(0) = 0.09$ are used. We consider the case where the control u is unbounded. The optimal control problem is to determine u in the time interval $0 \leq t < t_f$ that will minimize the performance index

$$I = \int_0^{t_f} (x_1^2 + x_2^2 + 0.1u^2)dt \tag{4.9}$$

where the dimensionless final time t_f is specified as 0.78. To evaluate the performance index conveniently and without introducing any unnecessary approximation, we introduce a new state variable x_3 by

$$\frac{dx_3}{dt} = x_1^2 + x_2^2 + 0.1u^2 \tag{4.10}$$

with the initial condition $x_3(0) = 0$, and integrate this equation along with the other two equations to yield the performance index. However, the **x**-grid pertains to x_1 and x_2 only. Suppose we divide the given time interval into 13 stages, each of length 0.06 ($P = 13$). To test the basic procedure, Luus [5] took the centre of the **x**-grid as (0,0) and for control 3.0. The initial region size for each x_i was taken as 0.09 and for the control 3.0. A reduction factor of 0.90 was used after every iteration. The convergence profile is given in Figure 4.2, where the specification of number of grid points and number of allowable values for control is given as $(N \times N, M)$. For adequate convergence, it is clear that more than 20 iterations are required even when a relatively large number of grid points are used.

4.8 Use of accessible states as grid points

Although easy to program, the method described above is not computationally attractive, since a very large number of calculations are required even for the 2-dimensional CSTR problem. The use of a coarse grid with region contraction to provide an accurate answer worked well with three test problems [5]. But even here the curse of dimensionality comes to the forefront. If we visualize an **x**-grid having only 5 values

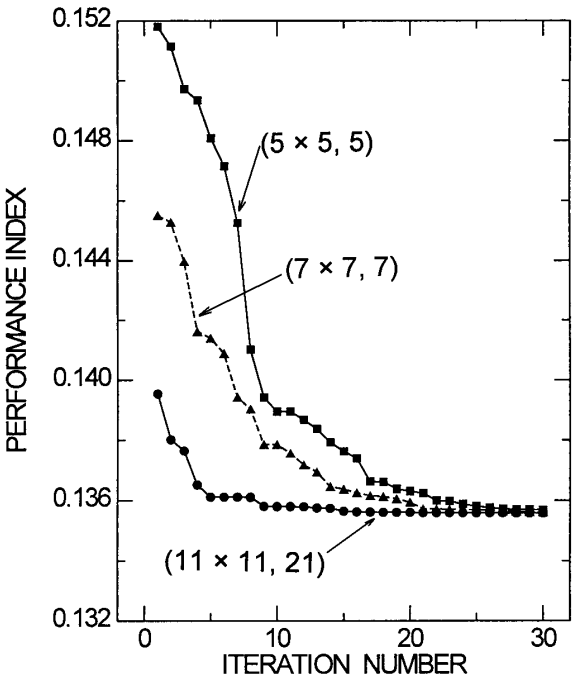

Figure 4.2: Convergence profile for the performance index, showing the effect of the number of grid points and the number of allowable values of control expressed by $(N \times N, M)$ for the CSTR

for each state variable, we see that for $n = 3$ there are 125 grid points to consider at every stage, which is still manageable; but when $n = 5$, the number of grid points would increase to 3,125, which is a very large number even for fast computers. We also observe that most of such grid points are inaccessible from the initial state specified; thus the computational procedure as outlined up to now is not very efficient.

This problem of very large number of grid points is overcome by using only the points that can actually be reached from the given state by applying the control. Therefore, instead of choosing the $\mathbf{x}$-grid points independently, it was suggested [4] to *generate* the grid points by applying different values for control. Then all the grid points are accessible from the specified initial state. Furthermore, the dimensionality of the state vector then does not matter, and a small number of grid points are required. The computational scheme can now be summarized as follows.

4.9 Algorithm for IDP

To illustrate the underlying logic in IDP, an algorithm is given to solve the optimal control problem as outlined in Eq.(4.1)-(4.3), where it is required to minimize the performance index in Eq. (4.3) with the use of piecewise constant control over P stages, each of the same length:

1. Divide the time interval $[0, t_f]$ into P time stages, each of length L.

2. Choose the number of test values for $\mathbf{u}$ denoted by R, an initial control policy and the initial region size r_{in}; also choose the region contraction factor γ used after every iteration and the number of grid points N.

3. Choose the total number of iterations to be used in every pass and set the iteration number index to $j = 1$.

4. Set the region size vector $\mathbf{r}^j = \mathbf{r}_{in}$.

5. By using the best control policy (the initial control policy for the first iteration) integrate Eq. (4.1) from $t = 0$ to t_f N times with different values for control to generate N $\mathbf{x}$-trajectories and store the values of $\mathbf{x}$ at the beginning of each time stage, so that $\mathbf{x}(k-1)$ corresponds to the value of $\mathbf{x}$ at beginning of stage k.

6. Starting at stage P, corresponding to time $t_f - L$, for each of the N stored values for $\mathbf{x}(P-1)$ from step 5 (grid points) integrate Eq. (4.1) from $t_f - L$ to t_f, with each of the R allowable values for the control vector calculated from

$$\mathbf{u}(P-1) = \mathbf{u}^{*j}(P-1) + \mathbf{D}\mathbf{r}^j \tag{4.11}$$

where $\mathbf{u}^{*j}(P-1)$ is the best value obtained in the previous iteration and $\mathbf{D}$ is a diagonal matrix of different random numbers between -1 and 1. Out of the R values for the performance index, choose the control values that give the minimum value, and store these values as $\mathbf{u}(P-1)$. We now have the best control for each of these N grid points.

7. Step back to stage $P-1$, corresponding to time $t_f - 2L$, and for each of the N grid points do the following calculations. Choose R values for $\mathbf{u}(P-2)$ as in the previous step, and by taking as the initial state $\mathbf{x}(P-2)$ integrate Eq. (4.1) over one stage length. Continue integration over the last time stage by using the stored value of $\mathbf{u}(P-1)$ from step 6 corresponding to the grid point that is closest to the value of the state vector that has been reached. Compare the R values of the performance index and store the $\mathbf{u}(P-2)$ that gives the minimum value for the performance index.

8. Continue the procedure until stage 1, corresponding to the initial time t = 0 and the given initial state, is reached. This stage has only a single grid point, since the initial state is specified. As before, integrate Eq. (4.1) and compare the R values of the performance index and store the control $\mathbf{u}(0)$ that gives the minimum performance index. Store also the corresponding $\mathbf{x}$-trajectory. This completes one iteration.

9. Reduce the region for allowable control

$$\mathbf{r}^{j+1} = \gamma \mathbf{r}^j \tag{4.12}$$

where j is the iteration number index. Use the best control policy from step 8 as the midpoint for the allowable values for the control denoted by the superscript *.

10. Increment the iteration index j by 1 and go to step 5 and continue the procedure for the specified number of iterations and interpret the results. In some problems it may be beneficial to use a multi-pass method, where, after a number of iterations

the region sizes are reset and the procedure is repeated.

In this algorithm the candidates for control are chosen at random. In the early development, the candidates for control were chosen from evenly spaced grids. For low-dimensional problems, very little difference is noted, but for higher dimensional control vectors the use of random values is better. The choice of allowable values for control is discussed in Chapter 5. The improvement in generating the grid points, rather than choosing the grid points as discussed earlier, is realized by considering the above CSTR example. By using a single grid point, convergence with 15 randomly chosen values for control was easily obtained. With $P = 13$, convergence in 3 passes yielded $I = 0.13558$ in computation time of 9.3 s on a Pentium/120 digital computer. The convergence profile is shown in Figure 4.3.

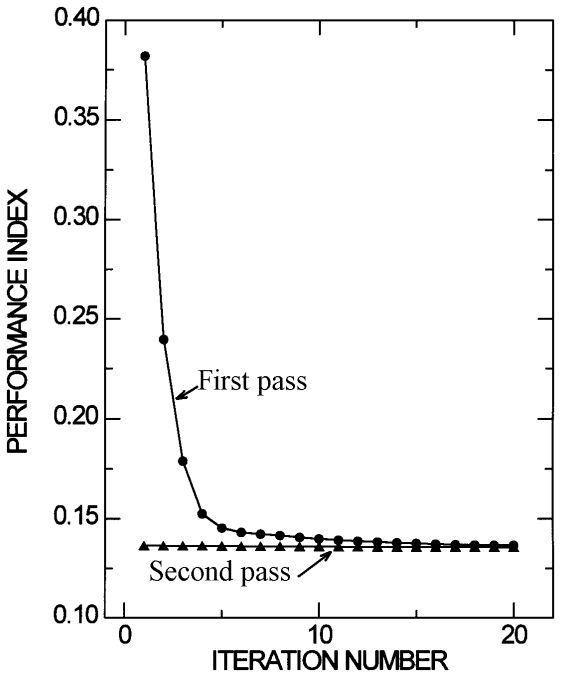

Figure 4.3: Convergence profile for the CSTR with the use of a single accessible grid point at each stage and $R = 15$ randomly chosen candidates for control

The computer program outlining the IDP algorithm and used for this problem is listed in its entirety in Appendix D. When $P = 20$ was used convergence to $I = 0.13416$ was obtained in 3 passes, requiring 14.1 s of CPU time on a Pentium/120 digital computer.

4.10 Early applications of IDP

Iterative dynamic programming provided a very convenient way of investigating the effect of the choice of the final time in optimal control problems [6]. However, by generating the grid points, it was no longer possible to guarantee a global optimum. This was illustrated by Luus and Galli [13]. Even the use of a very large number of grid points does not guarantee getting the global optimum. In fact, the number of grid points can be quite small in many cases and the global optimum is still obtained with good accuracy [1], [11].

A very challenging problem is the bifunctional catalyst problem, where it is necessary to determine the blend of the catalyst along a tubular reactor to maximize the yield of a desired component [12]. By using successive quadratic programming (SQP) and starting from 100 randomly chosen starting points, 26 local optima were located, but the global optimum was not obtained. With IDP, however, the global optimum was readily obtained with the use of a single grid point [11]. To avoid the numerous local optima, all that was required for this system was to take a sufficiently large initial region size for the control.

Although the optimal control of fed-batch reactors was very difficult to obtain by methods based on Pontryagin's maximum principle, iterative dynamic programming provided a reliable means of obtaining the global optimum, and the results were even marginally better than had been previously reported [8,9]. The additional advantage of IDP is that the computations are straightforward and the algorithm can be easily programmed to run on a personal computer.

Since derivatives are not required in the use of IDP, the method is applicable to more general types of optimal control problems. Also, since no auxiliary variables are necessary, except to handle state inequality constraints, the method is easier to use than variational methods based on Pontryagin's maximum principle for solving singular control problems [7]. As convergence properties of IDP are studied in greater detail, further improvements will inevitably be introduced, to make IDP even more useful. Recently Luus [10] showed that variable stage lengths can be incorporated into optimal control problems where state inequality constraints are also present, by combining the approach of Bojkov and Luus [2] along with that of Mekarapiruk and Luus [14]. Although the best choice for the penalty function to be used in IDP has not yet been established, good progress has been made in this field [15] and further research in this area is continuing. Furthermore, since no derivatives are required for IDP, the method should have important applications where nondifferentiable functions are encountered.

The goal for this book is to give a good understanding of IDP and also to realize the limitations. In order to accomplish that, the computational aspects are examined in depth using numerous examples. These are best presented in separate chapters.

4.11 References

[1] Bojkov, B. and Luus, R.: "Evaluation of the parameters used in iterative dynamic programming", *Can. J. Chem. Eng.* **71** (1993), 451-459.

[2] Bojkov, B. and Luus, R.: "Optimal control of nonlinear systems with unspecified final times", *Chem. Eng. Sci.* **51** (1996), 905-919.

[3] DeTremblay, M. and Luus, R.: "Optimization of non-steady state operation of reactors", *Can. J. Chem. Eng.* **67** (1989), 494-502.

[4] Luus, R.: "Optimal control by dynamic programming using accessible grid points and region reduction", *Hung. J. Ind. Chem.* **17** (1989), 523-543.

[5] Luus, R.: "Optimal control by dynamic programming using systematic reduction in grid size", *Int. J. Control* **19** (1990), 995-1013.

[6] Luus, R.: "Effect of the choice of final time in optimal control of nonlinear systems",*Can. J. Chem. Eng.* **69** (1991), 144-151.

[7] Luus, R.: "On the application of iterative dynamic programming to singular optimal control problems", *IEEE Trans. Auto. Contr.* **37** (1992), 1802-1806.

[8] Luus, R.: "Application of dynamic programming to differential-algebraic process systems", *Comp. Chem. Eng.* **17** (1993), 373-377.

[9] Luus, R.: "Optimization of fed-batch fermentors by iterative dynamic programming", *Biotech. Bioeng.* **41** (1993), 599-602.

[10] Luus, R.: "Use of variable stage-lengths for constrained optimal control problems", *Hung. J. Ind. Chem.* **25** (1997), 299-304.

[11] Luus, R. and Bojkov, B.: "Global optimization of the bifunctional catalyst problem", *Can. J. Chem. Eng.* **72** (1994), 160-163.

[12] Luus, R., Dittrich, J., and Keil, F.J.: "Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor", *Can. J. Chem. Eng.* **70** (1992), 780-785.

[13] Luus, R. and Galli, M.: "Multiplicity of solutions in using dynamic programming for optimal control", *Hung. J. Ind. Chem.* **19** (1991), 55-62.

[14] Mekarapiruk, W. and Luus, R.: "Optimal control of inequality state constrained systems", *Ind. Eng. Chem. Res.* **36** (1997), 1686-1694.

[15] Mekarapiruk, W. and Luus, R.: "Optimal control of final state constrained systems", *Can. J. Chem. Eng.* **25** (1997), 806-811.

Chapter 5

Allowable values for control

5.1 Introduction

There are several ways in which the candidates for control can be chosen. The simplest way, which was used in the early work on IDP, involves taking allowable values of control uniformly inside the given region. This means taking one point at the center of the region, one point at each end of the region, and then the rest of the points uniformly inside the interval. This scheme is easy to program and is also easy to visualize. For each control variable we therefore have a minimum of 3 values, namely $-r, 0,$ and r distance from the optimum value obtained at the previous iteration, where r is the region size. If there are two control variables, the minimum number of candidates is then 9. For m control variables we must examine then 3^m candidates at each grid point. This is fine if m is less than 5, but if m is large, this number becomes excessively large. Even with $m = 5$, the examination of 243 candidates for control at each grid point may be too much.

To reduce the minimum number of candidates for control, Tassone and Luus [7] suggested an alternative method of examining only the vertices of the region plus the center point. This reduces the minimum number of candidates for control from 3^m to $2^m + 1$, so that with $m = 4$, only 17 candidates for control need to be examined. Instead of vertices, the same reduction is obtained by using the center of the faces of the region. However, a better approach for high dimensional systems, as shown by Bojkov and Luus [1], is to choose control candidates at random inside the allowable region. This means that in theory there is no upper limit on m. Conceptually m could be greater than 100 and iterative dynamic programming could still be used. In fact, IDP was used successfully on a system with 130 differential equations and 130 control variables [4] and later with 250 differential equations with 250 control variables [5].

To gain insight into choosing control candidates and to see the relationship between choosing the candidates over a uniform distribution and choosing them at random, we consider an example involving 3 control variables.

5.2 Comparison of uniform distribution to random choice

Let us consider the CSTR outlined in Chapter 1, where the inlet flow rate for species A in dimensionless units is specified as $q_1 = 6$, and there are three control variables. The equations describing the system are:

$$\frac{dx_1}{dt} = q_1 - qx_1 - 17.6x_1x_2 - 23x_1x_6u_3 \tag{5.1}$$

$$\frac{dx_2}{dt} = u_1 - qx_2 - 17.6x_1x_2 - 146x_2x_3 \tag{5.2}$$

$$\frac{dx_3}{dt} = u_2 - qx_3 - 73x_2x_3 \tag{5.3}$$

$$\frac{dx_4}{dt} = -qx_4 + 35.2x_1x_2 - 51.3x_4x_5 \tag{5.4}$$

$$\frac{dx_5}{dt} = -qx_5 + 219x_2x_3 - 51.3x_4x_5 \tag{5.5}$$

$$\frac{dx_6}{dt} = -qx_6 + 102.6x_4x_5 - 23x_1x_6u_3 \tag{5.6}$$

$$\frac{dx_7}{dt} = -qx_7 + 46x_1x_6u_3 \tag{5.7}$$

$$\frac{dx_8}{dt} = 5.8(qx_1 - q_1) - 3.7u_1 - 4.1u_2 + q(23x_4 + 11x_5 + 28x_6 + 35x_7) - 5u_3^2 - 0.099 \tag{5.8}$$

where $q = q_1 + u_1 + u_2$ and $q_1 = 6$. Later we allow q_1 to be another control variable, but here we want to restrict the number of control variables to 3, as was done by Rao and Luus [6]. The last differential equation gives the performance index. The problem is to maximize

$$I = x_8(t_f) \tag{5.9}$$

where the final time t_f is specified as 0.2. The initial state is specified as $\mathbf{x}(0) = [0.1883 \quad 0.2507 \quad 0.0467 \quad 0.0899 \quad 0.1804 \quad 0.1394 \quad 0.1046 \quad 0]^T$. The control variables are constrained by

$$0 \le u_1 \le 20 \tag{5.10}$$

$$0 \le u_2 \le 6 \tag{5.11}$$

$$0 \le u_3 \le 4 \tag{5.12}$$

This example was used by Rao and Luus [6] to evaluate numerous algorithms employing control vector iteration based on Pontryagin's maximum principle. They found that the algorithms yielded results between $I = 20.05$ and $I = 20.09$ for the maximum performance index. This example was also used by Luus [2] in showing how iterative dynamic programming overcomes the curse of dimensionality, and by Luus [3] in

examining the effect of the choice of final time in optimal control. Therefore, this example should provide a good illustration of the different ways of choosing candidates for control in iterative dynamic programming.

Here we use $P = 10$ stages of equal length (each of length 0.02), and for integration of the differential equations we use the fourth order Runge-Kutta method with an integration time step of length 0.005 to provide good accuracy. For all of the runs we used $N = 3$ grid points for the state at stages 2 to 10. At stage 1 we simply have the given initial state as the grid point.

5.2.1 Uniform distribution

By using piecewise constant control over the 10 time stages, convergence to $I = 20.09$ was readily obtained. The optimal control policy is given in Table 5.1.

Table 5.1: Piecewise constant optimal control policy giving $I = 20.08950$

Stage	Time	u_1	u_2	u_3
1	0.00	5.96725	3.03015	1.03413
2	0.02	7.40361	1.15660	1.01755
3	0.04	6.61664	1.78472	1.00934
4	0.06	6.44364	1.73329	1.03118
5	0.08	6.57736	1.98866	1.08532
6	0.10	8.32409	3.67016	1.08425
7	0.12	10.99145	6.00000	0.94330
8	0.14	10.56186	6.00000	0.79619
9	0.16	20.00000	6.00000	0.54567
10	0.18	20.00000	6.00000	0.18901

The convergence profiles showing deviations from the optimum $I = 20.08950$ are shown in Figure 5.1 for region contraction factors of $\gamma = 0.85$ and in Figure 5.2 for $\gamma = 0.90$.

It is seen that the convergence rate is not very sensitive to the region reduction factor γ. Only 20 iterations are required to get the value of the performance index within 0.001 of the optimum. Increasing the number of allowable values for control increases the convergence rate at the expense of increased computational effort. The use of 3 values for each control ($M = 3$) required 5.5 s of computer time for 50 iterations, the use of 5 values for each control required 15.3 s, and the use of $M = 7$ required 37.0 s on a PentiumII/300 computer for 50 iterations.

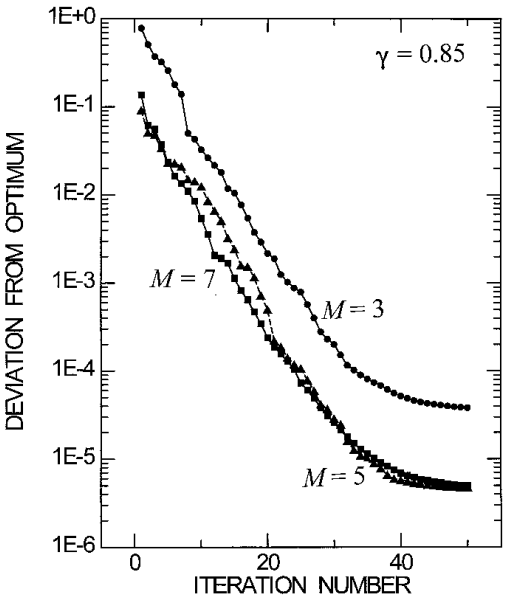

Figure 5.1: Convergence profile showing the effect of the number of uniformly chosen values for each control using a region contraction factor $\gamma = 0.85$

5.2.2 Random choice

To provide direct comparison, the numbers of candidates for control chosen at random were chosen to correspond to the three above cases, namely $R = 27, R = 125$, and $R = 343$ for the same two values for the region contraction factors. The convergence profiles are given in Figures 5.3 and 5.4. It is observed that the use of $R = 27$ does not provide as fast convergence as was obtained with $M = 3$, and deviation less than 0.001 could not be reached. However with $R = 125$ and $R = 343$ comparable convergence rates were obtained as with the choice of uniformly distributed values for control. It is interesting to note that with $R = 125$ a better convergence rate was obtained than with $R = 343$. The computation times for the three values of R were 2.9, 12.4, and 33.6 s.

We now consider the same example but allow q_1 to be another control variable, $q_1 = u_4$, as was done by Luus [2], using three uniformly chosen values for each control variable. This meant that 81 candidates for control had to be examined at each grid point. By using $P = 11$ stages and placing an upper limit of 20 on this additional control variable Luus [2] obtained a value of $I = 21.75722$, with the control policy given in Table 5.2. It is interesting to note that u_4 switches between the upper and lower limits.

The effect of the number of candidates for control chosen at random inside the region with the use of $N = 3$ grid points is given in Table 5.3 where a maximum of 150 iterations was allowed and the region reduction factor γ was varied between 0.90

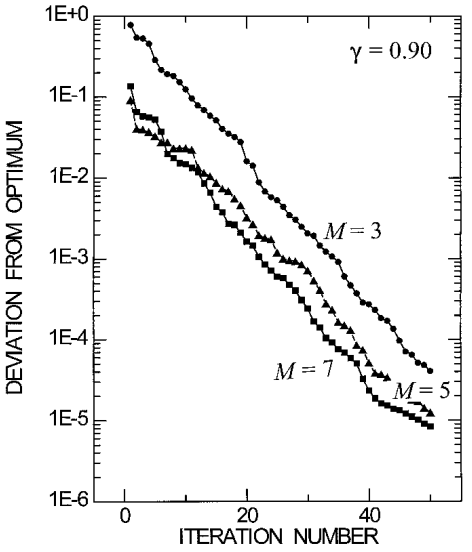

Figure 5.2: Convergence profile showing the effect of the number of uniformly chosen values for each control using a region contraction factor $\gamma = 0.90$

and 0.99. The global optimum is shown in boldface. As can be seen, there is some advantage in taking a larger number of candidates for control, but even with $R = 25$, the optimum can be accurately determined.

Let us now compare the two different methods of choosing candidates for control. By taking 3 uniformly distributed values for each control variable at $-r$, 0, and r from the best value means taking 81 candidates for control at each grid point at each time step. As is seen in Table 5.4, the region contraction factor γ can be taken smaller, and a smaller number of iterations yield convergence. In making the runs, the maximum number of iterations allowed was 150.

To reduce the number of candidates for uniform search, we may take the vertices of the region as candidates for control as suggested by Tassone and Luus [7]. Then instead of 81 values we may take 17 values for control. Although candidates chosen at random may appear to provide an inefficient means of getting the optimum, we shall see in Chapter 6, how the scheme is incorporated into a multi-pass method where the sensitivity of the parameters is reduced and the global optimum is more readily obtained.

If the number of control variables is small (less than 5), either method may be used for choosing the candidates for control. However, if the number of control variables is larger than 5, then choosing the candidates at random is a considerably better approach. This idea is illustrated with several examples by Bojkov and Luus [1] who also considered a problem with 20 control variables. For the latter, the only feasible approach was to use randomly chosen candidates for control. We consider the optimal control of high-dimensional systems later in Chapter 7.

Table 5.2: Piecewise constant optimal control policy with four control variables giving the maximum $I = 21.75722$

Stage	Time	u_1	u_2	u_3	u_4
1	0.0000	0.00000	0.00000	1.64594	20.00000
2	0.0182	14.17009	0.00000	1.49448	20.00000
3	0.0364	19.30869	0.00000	0.71145	20.00000
4	0.0546	12.21929	6.00000	0.27487	0.00000
5	0.0727	3.61857	3.99978	0.24286	0.00000
6	0.0909	0.00000	0.00000	0.49520	0.00000
7	0.1091	0.00000	0.00000	0.98515	0.00000
8	0.1273	0.00000	0.00000	1.89196	20.00000
9	0.1455	11.67268	0.00000	1.91146	20.00000
10	0.1636	20.00000	6.00000	0.95895	20.00000
11	0.1818	20.00000	6.00000	0.26054	20.00000

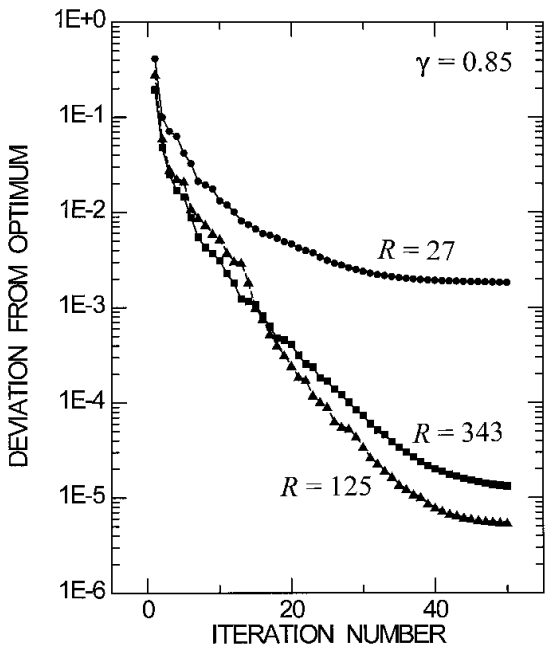

Figure 5.3: Convergence profile showing the effect of the number of randomly chosen candidates for control with $\gamma = 0.85$

Table 5.3: Effect of the number of randomly chosen candidates for control at each grid point with $N = 3$ grid points

	$R = 25$ randomly chosen control values		$R = 50$ randomly chosen control values	
Reduction factor γ	Performance index, I	Number of iterations	Performance index, I	Number of iterations
0.90	21.6564	150	21.6149	150
0.91	21.6555	150	21.5067	150
0.92	21.7044	150	**21.7572**	69
0.93	21.6770	150	**21.7572**	68
0.94	21.7375	150	21.5115	150
0.95	**21.7572**	93	21.5115	150
0.96	**21.7572**	118	**21.7572**	96
0.97	21.7555	150	**21.7572**	138
0.98	21.7565	150	21.7570	150
0.99	21.7535	150	21.7505	150

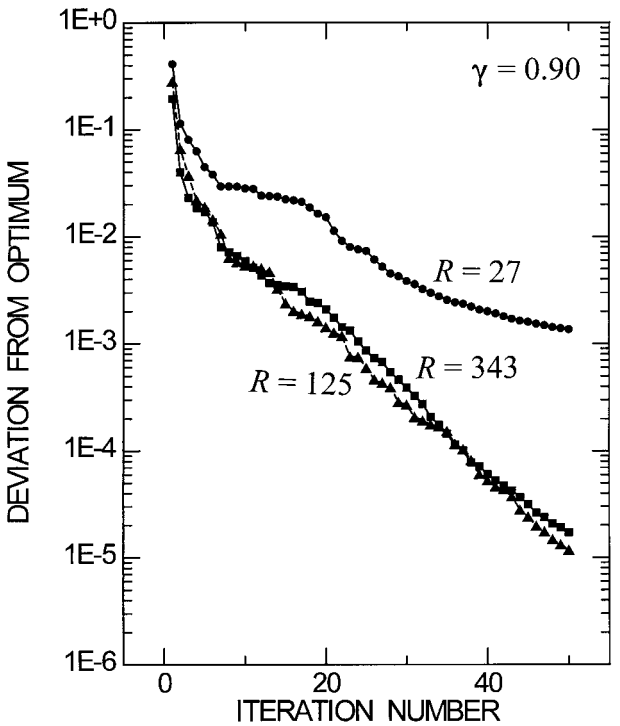

Figure 5.4: Convergence profile showing the effect of the number of randomly chosen candidates for control with $\gamma = 0.90$

Table 5.4: Comparison of the use of uniform means of taking control candidates to the use of randomly chosen values inside the region

	Using uniform means of choosing control		Using 81 randomly chosen control values inside the region	
Reduction factor γ	Performance index, I	Number of iterations	Performance index, I	Number of iterations
0.80	21.7030	150	21.6112	150
0.82	21.6547	150	21.6669	150
0.84	**21.7572**	29	21.6405	150
0.86	21.5115	150	21.7231	150
0.88	**21.7572**	39	21.7463	150
0.90	21.5115	150	21.7549	150
0.92	**21.7572**	60	**21.7572**	49
0.94	**21.7572**	80	21.6927	150
0.96	**21.7572**	118	**21.7572**	104
0.98	21.7540	150	21.7571	150

5.3 References

[1] Bojkov, B. and Luus, R. : "Use of random admissible values for control in iterative dynamic programming", *Ind. Eng. Chem. Res.* **31** (1992), 1308-1314.

[2] Luus, R.: "Application of dynamic programming to high-dimensional nonlinear optimal control problems", *Int. J. Control* **52** (1990), 239-250.

[3] Luus, R.: "Effect of the choice of final time in optimal control of nonlinear systems",*Can. J. Chem. Eng.* **69** (1991), 144-151.

[4] Luus, R.: "Application of iterative dynamic programming to very high-dimensional systems", *Hung. J. Ind. Chem.* **21** (1993), 243-250.

[5] Luus, R.: "Numerical convergence properties of iterative dynamic programming when applied to high dimensional systems", *Chem. Eng. Res. Des.* **74** (1996), 55-62.

[6] Rao, S.N. and Luus, R.: "Evaluation and improvement of control vector iteration procedures for optimal control", *Can. J. Chem. Eng.* **50** (1972), 777-784.

[7] Tassone, V. and Luus, R.: "Reduction of allowable values for control in iterative dynamic programming", *Chem. Eng. Sci.* **48** (1993), 3864-3867.

Chapter 6

Evaluation of parameters in IDP

6.1 Introduction

In iterative dynamic programming there are numerous parameters that can be varied. Already in Chapter 5 we saw the different ways of choosing candidates for control and the effects of the number of such values chosen at each grid point, and also the effect of the region contraction factor γ. If γ is too small, we get premature collapse of the region, and if γ is too large then either the convergence rate may be rather slow or convergence may not be obtained at all. Fortunately there is generally a wide range over which acceptable values for γ may be chosen to yield convergence to the optimum. There is a balance between the number of allowable values for control R and the region contraction factor γ. If R is chosen to be sufficiently large, then γ can be chosen to be quite small such as 0.7 without fear of premature region collapse. However, if a small value for R is used, then γ must be increased to a value closer to 0.9. Normally we would like to choose R as small as possible to reduce computational effort.

The choice for the number of grid points, however, is not quite as clear. In a very difficult nonlinear optimal control problem involving the determination of the optimal catalyst blend along a tubular reactor [7], convergence to the global optimum was readily obtained by using a single grid point at each time stage [6]. A straightforward fed-batch reactor optimization problem, on the other hand, required a minimum of 23 grid points for convergence to the global optimum [3]. Bojkov and Luus [2] studied the effects of the parameters in IDP and concluded that for many typical chemical engineering systems the region contraction factor γ should be in the range 0.75 to 0.95, and the use of more than 3 grid points does not increase the likelihood of getting the global optimum nor improve the convergence rate. Although surprising, with the exception of a few reactor problems such as the fed-batch reactor considered by Hartig *et al.* [3] and the two-stage CSTR we consider later in this chapter, the requirement of only a small number of grid points appears to be generally true for many optimal control problems. Therefore, we may use a single grid point at each time stage for

many problems and still obtain fast and accurate convergence to the global optimum.

The goal here is to illustrate the effects of the parameters used in IDP, such as the region reduction factor γ used after every iteration to reduce the size of the region, the region restoration factor η which is used to restore the region size after every pass, the initial region size, number of allowable values for control, and the number of grid points used at each time step. The use of several examples is the best way to show the relationship among the parameters used in IDP.

6.2 Number of grid points

In the early work with iterative dynamic programming, the number of grid points used was chosen to be quite large, usually greater than 20. The motivation for the use of large number of grid points was due to the vectorization capability of the CRAY supercomputer that was used, enabling the calculations to be done in parallel fashion. When calculations were switched from the CRAY to personal computers, the need to examine the effect of the number of grid points became necessary, since doubling the number of grid points more than doubled the computation time. The investigation of Bojkov and Luus [2] made it clear that a large number of grid points is not always necessary. Furthermore, it is obvious that if the number of grid points can be reduced without sacrificing convergence rate, then we should use the minimum number, namely $N = 1$, whenever possible.

The choice of the initial region size over which candidates for control are chosen is also very important, especially when numerous local optima are present. To illustrate the effects of grid points, the choice of the initial region size, and the relationship of other parameters, we choose two examples from chemical engineering. In the first example we use a single grid point, since convergence to the global optimum is so readily obtained that there is no need to examine the use of a larger number of grid points. In the second example, the effect of the use of a different number of grid points is illustrated. In Section 6.3 we examine the multi-pass approach and present another example for which we need several grid points to get accurate convergence.

6.2.1 Bifunctional catalyst blend optimization problem

In the optimization of the blend of a bifunctional catalyst in converting methylcyclopentane to benzene in a tubular reactor, the blend of two monofunctional catalyst components involving hydrogenation and isomerization is characterized by the mass fraction u given by

$$u = \frac{\text{mass of hydrogenation component}}{\text{total mass of the catalyst}}. \tag{6.1}$$

The characteristic time can be visualized as the distance along the tubular reactor defined by

$$t = \frac{\text{mass of catalyst up to a given section of reactor}}{\text{inlet molar flow rate of methylcyclopentane}}, \tag{6.2}$$

so that at the exit of the reactor, the final time is

$$t_f = \frac{\text{total mass of catalyst in the reactor}}{\text{molar flow rate of methylcyclopentane into the reactor}}. \tag{6.3}$$

The chemical reactions taking place are described by seven differential equations:

$$\frac{dx_1}{dt} = -k_1 x_1 \tag{6.4}$$

$$\frac{dx_2}{dt} = k_1 x_1 - (k_2 + k_3)x_2 + k_4 x_5 \tag{6.5}$$

$$\frac{dx_3}{dt} = k_2 x_2 \tag{6.6}$$

$$\frac{dx_4}{dt} = -k_6 x_4 + k_5 x_5 \tag{6.7}$$

$$\frac{dx_5}{dt} = k_3 x_2 + k_6 x_4 - (k_4 + k_5 + k_8 + k_9)x_5 + k_7 x_6 + k_{10} x_7 \tag{6.8}$$

$$\frac{dx_6}{dt} = k_8 x_5 - k_7 x_6 \tag{6.9}$$

$$\frac{dx_7}{dt} = k_9 x_5 - k_{10} x_7 \tag{6.10}$$

with the initial condition

$$\mathbf{x}(0) = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad]^T. \tag{6.11}$$

The rate constants are expressed as cubic functions of the catalyst blend u

$$k_i = c_{i0} + c_{i1}u + c_{i2}u^2 + c_{i3}u^3, \qquad i = 1, 2, \cdots, 10. \tag{6.12}$$

The coefficients c_{ij} are given in Table 6.1. It is required to find the catalyst blend u along the length of the reactor to maximize the concentration of benzene. Thus the performance index to be maximized is chosen as

$$I = x_7(t_f) \times 10^3, \tag{6.13}$$

where $t_f = 2000$g h/mol. We have scaled the final concentration of x_7 by the factor 10^3 to simplify the presentation of results. The catalyst blend is constrained by

$$0.60 \leq u \leq 0.9. \tag{6.14}$$

Table 6.1: Coefficients for the rate constants k_i used in bifunctional catalyst problem

i	c_{i0}	c_{i1}	c_{i2}	c_{i3}
1	0.2918487D-02	-0.8045787D-02	0.6749947D-02	-0.1416647D-02
2	0.9509977D+01	-0.3500994D+02	0.4283329D+02	-0.1733333D+02
3	0.2682093D+02	-0.9556079D+02	0.1130398D+03	-0.4429997D+02
4	0.2087241D+03	-0.7198052D+03	0.8277466D+03	-0.3166655D+03
5	0.1350005D+01	-0.6850027D+01	0.1216671D+02	-0.6666689D+01
6	0.1921995D-01	-0.7945320D-01	0.1105666D+00	-0.5033333D-01
7	0.1323596D+00	-0.4696255D+00	0.5539323D+00	-0.2166664D+00
8	0.7339981D+01	-0.2527328D+02	0.2993329D+02	-0.1199999D+02
9	-0.3950534D+00	0.1679353D+01	-0.1777829D+01	0.4974987D+00
10	-0.2504665D-04	0.1005854D-01	-0.1986696D-01	0.9833470D-02

As was done by Luus *et al.* [7], to solve this optimal control problem, we divide the given time interval into 10 equal sections, each of length 200 g h/mol, and determine the piecewise constant values for control in each section $u(0), u(1), \cdots, u(9)$ to maximize the performance index given in Eq. (6.13). For integration we used the integration subroutine DVERK of Hull *et al.* [4] with local error tolerance of 10^{-6}. There are numerous local optima. By using sequential quadratic programming as suggested by Rosen and Luus [12], from 100 initial points chosen at random 25 local optima were obtained, with the following values of the performance index: 10.053, 9.905, 9.881, 9.866, 9.815, 9.786, 9.722, 9.641, 9.628, 9.608, 9.582, 9.534, 9.501, 9.463, 9.439, 9.343, 9.293, 9.246, 9.148, 9.037, 8.673, 8.645, 8.490, 8.402, and 8.113. However, none of these is the global optimum.

The global optimal control policy and the next four best ones are given in Table 6.2. As was pointed out by Luus *et al.* [7], there are numerous local optima for this problem, many of which are not listed above. The aim is to get the global optimum. From the initial control policy $u^{(0)}(k) = 0.75, \quad k = 0, 1, \cdots, 9$ and initial region size $r^{(0)}(k) = 0.3, \quad k = 0, 1, \cdots, 9$ with the region contraction factor $\gamma = 0.70$, the number of grid points $N = 21$ and $M = 5$ uniformly chosen values for control, Luus *et al.* [7] obtained the global optimum with the use of IDP in 16 iterations. The M candidates for control were chosen from the uniform distribution around the best value, i.e., u^*, $u^* \pm [\frac{2j}{M-1}]r, \quad j = 1, 2, \cdots, (M-1)/2$, where u^* is the best value for the catalyst blend from the previous iteration.

Luus and Bojkov [6] showed that the global optimum could be obtained with a single grid point, $N = 1$, if the initial region is chosen to be sufficiently large. In Tables 6.3 and 6.4 we show the effect of the region contraction factor γ, the initial region size $r^{(0)}$, and the number of uniformly chosen candidates for control M. The number of iterations to reach the global optimum to six figures is shown in parentheses. A

Table 6.2: Control policies for the best 5 local optima

Local optimum	1 (global)	2	3	4	5
Performance index	10.0942	10.0527	10.0395	9.9047	9.8805
u(0)	0.6661	0.6651	0.6671	0.6637	0.9000
u(1)	0.6734	0.6721	0.6744	0.9000	0.6724
u(2)	0.6764	0.9000	0.6777	0.9000	0.6755
u(3)	0.9000	0.9000	0.6796	0.9000	0.9000
u(4)	0.9000	0.9000	0.9000	0.9000	0.9000
u(5)	0.9000	0.9000	0.9000	0.9000	0.9000
u(6)	0.9000	0.9000	0.9000	0.9000	0.9000
u(7)	0.9000	0.9000	0.9000	0.9000	0.9000
u(8)	0.9000	0.9000	0.9000	0.9000	0.9000
u(9)	0.9000	0.9000	0.9000	0.9000	0.9000

Table 6.3: Value of the performance index I with $N = 1$, and $M = 3$, showing the effects of the region contraction factor γ and the initial region size $r^{(0)}$; the number of iterations required to get the global optimum is shown in parentheses

Reduction factor	Initial region size $r^{(0)}$			
γ	0.27	0.30	0.40	0.50
0.70	10.0527	**10.0942** (18)	**10.0942** (19)	**10.0942** (20)
0.72	10.0527	**10.0942** (20)	**10.0942** (21)	10.0527
0.74	**10.0942** (22)	**10.0942** (22)	**10.0942** (23)	**10.0942** (24)
0.76	**10.0942** (23)	**10.0942** (24)	**10.0942** (25)	**10.0942** (26)
0.78	**10.0942** (26)	**10.0942** (26)	**10.0942** (28)	**10.0942** (29)
0.80	**10.0942** (29)	**10.0942** (29)	**10.0942** (31)	**10.0942** (32)
0.82	**10.0942** (32)	**10.0942** (32)	**10.0942** (34)	**10.0942** (36)
0.84	**10.0942** (36)	**10.0942** (38)	**10.0942** (39)	**10.0942** (40)
0.86	10.0941	10.0941	10.0941	10.0941
0.88	10.0940	10.0940	10.0939	10.0937
0.90	10.0933	10.0930	10.0925	10.0913

Table 6.4: Value of the performance index I with $N = 1$, and $M = 9$, showing the effects of the region contraction factor γ and the initial region size $r^{(0)}$; the number of iterations required to get the global optimum is shown in parentheses

Reduction factor	Initial region size $r^{(0)}$			
γ	0.27	0.30	0.40	0.50
0.70	10.0395	**10.0942** (15)	10.0527	**10.0942** (17)
0.72	10.0395	**10.0942** (16)	**10.0942** (17)	10.0527
0.74	10.0395	**10.0942** (17)	**10.0942** (18)	**10.0942** (19)
0.76	**10.0942** (19)	**10.0942** (19)	**10.0942** (20)	**10.0942** (21)
0.78	**10.0942** (21)	**10.0942** (21)	**10.0942** (22)	**10.0942** (23)
0.80	**10.0942** (22)	**10.0942** (23)	**10.0942** (24)	**10.0942** (25)
0.82	**10.0942** (25)	**10.0942** (26)	**10.0942** (27)	**10.0942** (28)
0.84	**10.0942** (28)	**10.0942** (29)	**10.0942** (30)	**10.0942** (32)
0.86	**10.0942** (33)	**10.0942** (34)	**10.0942** (36)	**10.0942** (37)
0.88	**10.0942** (39)	**10.0942** (39)	10.0941	10.0941
0.90	10.0941	10.0941	10.0940	10.0940

maximum of 40 iterations were allowed. As can be seen, the number of iterations to reach the global optimum is reduced when using $M = 9$, rather than $M = 3$. In each case, the global optimum was obtained in almost every case when the initial region size was chosen greater than 0.27. Of the 44 runs, the local optimum was obtained 3 times with the use of $M = 3$. With $M = 9$, local optima were obtained 5 times in the 44 runs. Therefore, the use of a larger number of allowable values for control does not necessarily improve the chances of avoiding local optima.

As expected, with $M = 9$ we can have a larger value for the region contraction factor γ and still get convergence to the global optimum to 6 figures within 40 iterations. However, it must be remembered that the use of $M = 9$ instead of $M = 3$ requires three times as much computational effort per iteration. Further results on this system are given by Luus and Bojkov [6], who showed, in addition, that instead of choosing the candidates for control in uniform fashion, the random choice gives essentially the same results. The computation time on a PentiumII/350 for 20 iterations with $M = 3$ was 6 s.

6.2.2 Photochemical CSTR

Let us consider the photochemical continuous stirred tank reactor (CSTR) used by Lapidus and Luus [5], which we considered in Chapter 5 to examine the effect of the number of grid points. Here, we consider the reactor with 4 control variables:

$$\frac{dx_1}{dt} = u_4 - qx_1 - 17.6x_1x_2 - 23x_1x_6u_3 \tag{6.15}$$

$$\frac{dx_2}{dt} = u_1 - qx_2 - 17.6x_1x_2 - 146x_2x_3 \tag{6.16}$$

$$\frac{dx_3}{dt} = u_2 - qx_3 - 73x_2x_3 \tag{6.17}$$

$$\frac{dx_4}{dt} = -qx_4 + 35.2x_1x_2 - 51.3x_4x_5 \tag{6.18}$$

$$\frac{dx_5}{dt} = -qx_5 + 219x_2x_3 - 51.3x_4x_5 \tag{6.19}$$

$$\frac{dx_6}{dt} = -qx_6 + 102.6x_4x_5 - 23x_1x_6u_3 \tag{6.20}$$

$$\frac{dx_7}{dt} = -qx_7 + 46x_1x_6u_3 \tag{6.21}$$

$$\frac{dx_8}{dt} = 5.8(qx_1 - q_1) - 3.7u_1 - 4.1u_2 + q(23x_4 + 11x_5 + 28x_6 + 35x_7) - 5u_3^2 - 0.099 \tag{6.22}$$

where the total inlet flow rate is given by $q = u_1 + u_2 + u_4$. The last differential equation is introduced to give the performance index. The problem is to maximize

$$I = x_8(t_f) \tag{6.23}$$

where the final time t_f is specified as 0.2. The initial state is specified as $\mathbf{x}(0) = [0.1883 \quad 0.2507 \quad 0.0467 \quad 0.0899 \quad 0.1804 \quad 0.1394 \quad 0.1046 \quad 0]^T$. The control variables are constrained by

$$0 \le u_1 \le 20 \tag{6.24}$$

$$0 \le u_2 \le 6 \tag{6.25}$$

$$0 \le u_3 \le 4 \tag{6.26}$$

$$0 \le u_4 \le 20. \tag{6.27}$$

In Chapter 5 we saw that with $P = 11$ time stages of equal length, over which piecewise constant control was used, the maximum value for the performance index was found to be 21.75722. Here we wish to illustrate how the number of grid points affects our capability of getting this optimum. As we did in Chapter 5, we choose as initial values for the control to be in the middle of the allowable region, i.e., $u_1^{(0)} = 10$, $u_2^{(0)} = 3$, $u_3^{(0)} = 2$, and $u_4^{(0)} = 10$, and let the initial region sizes be $r_1^{(0)} = 10$, $r_2^{(0)} = 3$, $r_3^{(0)} = 2$, and $r_4^{(0)} = 10$. By choosing $R = 81$ allowable values at random inside the allowable region that was contracted by the reduction factor γ after every iteration, the problem was solved for different values of γ and the number of grid points N. For each case 100 iterations were allowed. The results are given in Table 6.5. It is clear that there is a range for the region contraction factor $0.90 \le \gamma \le 0.96$

Table 6.5: Effect of the number of grid points on convergence with $R = 81$; the numbers in the parentheses give the number of iterations required for convergence

	Number of grid points at each time stage			
γ	$N = 1$	$N = 5$	$N = 11$	$N = 27$
0.80	21.4649	21.5476	21.6894	21.6575
0.81	21.6019	21.6621	21.5686	21.6263
0.82	21.6145	21.6624	21.6782	21.5847
0.83	21.6222	21.6729	21.7415	21.7272
0.84	21.6350	21.7276	21.6931	21.7547
0.85	21.6286	21.7370	21.6141	21.7360
0.86	21.6293	21.7085	21.4995	21.7353
0.87	21.6427	21.7522	21.7331	21.7329
0.88	21.6556	**21.7572** (32)	**21.7572** (34)	**21.7572** (30)
0.89	21.6603	**21.7572** (37)	21.7072	**21.7572** (32)
0.90	21.7506	**21.7572** (39)	**21.7572** (39)	**21.7572** (32)
0.91	**21.7572** (61)	**21.7572** (44)	**21.7572** (38)	**21.7572** (39)
0.92	**21.7572** (53)	**21.7572** (43)	**21.7572** (41)	**21.7572** (34)
0.93	**21.7572** (58)	**21.7572** (49)	**21.7572** (43)	**21.7572** (46)
0.94	**21.7572** (67)	**21.7572** (55)	**21.7572** (51)	**21.7572** (51)
0.95	**21.7572** (75)	**21.7572** (72)	**21.7572** (44)	**21.7572** (52)
0.96	**21.7572** (96)	**21.7572** (100)	**21.7572** (75)	**21.7572** (68)
0.97	21.7571	**21.7572** (70)	21.5007	**21.7572** (95)
0.98	21.7554	21.7554	21.7565	21.7569
0.99	21.7559	21.7554	21.7552	21.5105

over which the global optimum is obtained, regardless of the number of grid points. Where the optimum was obtained, the number of iterations required to reach the optimum is given in parentheses. The reduction in the number of iterations required for convergence is only marginally reduced when N is increased. However, the range is increased for larger values of N. For example, with $N = 27$ the range for convergence to the global optimum is $0.88 \leq \gamma \leq 0.97$. It is obvious that for this optimal control problem, a larger number of grid points than $N = 1$ is not justified.

6.3 Multi-pass approach

One method of preventing the collapse of the search region is to use the iterative dynamic programming in a multi-pass fashion, so that the region is restored to a fraction of its size at the beginning of the previous pass. The question that arises is, what should the restoration factor be and how sensitive are the results to that

choice? By taking a single grid point, i.e., $N = 1$, and only $R = 15$ random points, while allowing 20 iterations per pass, we solved the above optimal control problem for different values of the region contraction factor γ for different values of the region restoration factor η.

As is seen in Table 6.6, the success rate is almost 100% if the restoration factor is between 0.70 and 0.85. Of the 80 cases, only two yielded values different from the global optimum of 21.7572, where a maximum of 20 passes was allowed. When the restoration factor was 0.90, the region size did not shrink sufficiently fast to yield convergence to six figures in 20 passes. The use of the multi-pass method is a good means of keeping the number of candidates for control small and still providing accurate convergence. This is especially important when we consider systems of higher dimension.

6.3.1 Nonlinear two-stage CSTR system

Let us consider a very difficult optimal control problem that arose from trying to get a good initial control policy for a time-delay system [8]. The system consists of a series of two CSTRs, where there is a transportation delay $\tau = 0.1$ from the first tank to the second. This system was first investigated by Oh and Luus [11] and is described in greater detail in Chapter 8. Here we use a truncated Taylor series expansion for the time delay terms according to Mutharasan and Luus [9], to give a nondelay approximation

$$\frac{dx_1}{dt} = f_1 \tag{6.28}$$

$$\frac{dx_2}{dt} = f_2 \tag{6.29}$$

$$\frac{dx_3}{dt} = x_1 - x_3 - \tau f_1 - R_2 + 0.25 \tag{6.30}$$

$$\frac{dx_4}{dt} = x_2 - 2x_4 - u_2[x_4 + 0.25] - \tau f_2 + r_2 - 0.25 \tag{6.31}$$

$$\frac{dx_5}{dt} = x_1^2 + x_2^2 + x_3^2 + x_4^2 + 0.1(u_1^2 + u_2^2) \tag{6.32}$$

where

$$f_1 = 0.5 - x_1 - R_1 \tag{6.33}$$

$$f_2 = -2[x_2 + 0.25] - u_1[x_2 + 0.25] + R_1 \tag{6.34}$$

$$R_1 = [x_1 + 0.5]\exp(\frac{25x_2}{x_2 + 2}) \tag{6.35}$$

$$R_2 = [x_3 + 0.25]\exp(\frac{25x_4}{x_4 + 2}). \tag{6.36}$$

The state variables x_1 and x_3 are normalized concentration variables in tanks 1 and 2, respectively, and x_2 and x_4 are normalized temperature variables in tanks 1 and

Table 6.6: Effect of the restoration factor η on convergence with $R = 15$, showing the number of passes required for convergence in parentheses

	Restoration factor used after every pass of 20 iterations each				
γ	$\eta = 0.70$	$\eta = 0.75$	$\eta = 0.80$	$\eta = 0.85$	$\eta = 0.90$
0.80	**21.7572** (9)	**21.7572** (11)	**21.7572** (10)	**21.7572** (9)	**21.7572** (16)
0.81	**21.7572** (8)	**21.7572** (11)	**21.7572** (7)	**21.7572** (13)	**21.7572** (12)
0.82	**21.7572** (9)	**21.7572** (10)	**21.7572** (8)	**21.7572** (7)	**21.7572** (10)
0.83	**21.7572** (7)	**21.7572** (8)	**21.7572** (5)	**21.7572** (5)	**21.7572** (8)
0.84	**21.7572** (6)	**21.7572** (7)	**21.7572** (9)	**21.7572** (6)	**21.7572** (8)
0.85	**21.7572** (10)	**21.7572** (9)	**21.7572** (11)	**21.7572** (10)	**21.7572** (16)
0.86	**21.7572** (8)	**21.7572** (11)	**21.7572** (11)	**21.7572** (13)	**21.7572** (8)
0.87	**21.7572** (8)	**21.7572** (5)	**21.7572** (9)	**21.7572** (13)	**21.7572** (10)
0.88	**21.7572** (9)	**21.7572** (11)	**21.7572** (11)	**21.7572** (12)	21.7571
0.89	**21.7572** (6)	**21.7572** (8)	**21.7572** (12)	**21.7572** (15)	**21.7572** (16)
0.90	**21.7572** (11)	**21.7572** (12)	**21.7572** (14)	**21.7572** (19)	21.7570
0.91	**21.7572** (9)	**21.7572** (12)	**21.7572** (14)	**21.7572** (18)	21.7571
0.92	**21.7572** (11)	**21.7572** (12)	**21.7572** (14)	**21.7572** (19)	21.7570
0.93	**21.7572** (11)	**21.7572** (12)	**21.7572** (13)	**21.7572** (19)	21.7569
0.94	**21.7572** (8)	**21.7572** (12)	**21.7572** (12)	21.7571	21.7568
0.95	**21.7572** (11)	**21.7572** (13)	**21.7572** (15)	**21.7572** (18)	21.7570
0.96	**21.7572** (10)	**21.7572** (11)	**21.7572** (16)	**21.7572** (20)	21.7567
0.97	**21.7572** (13)	**21.7572** (12)	**21.7572** (17)	**21.7572** (17)	21.7571
0.98	**21.7572** (12)	**21.7572** (13)	**21.7572** (16)	**21.7572** (20)	21.7565
0.99	**21.7572** (10)	**21.7572** (14)	**21.7572** (19)	21.7571	21.7566

2, respectively. The variable x_5 is introduced to provide the performance index to be minimized:

$$I = x_5(t_f) \tag{6.37}$$

where the dimensionless final time $t_f = 2$. The initial state is

$$\mathbf{x}(0) = [\ 0.15\ \ -0.03\ \ 0.10\ \ 0\ \ 0\ ]^T. \tag{6.38}$$

Let us consider the problem of determining the piecewise constant control in the time interval $0 \le t < 2$ that will minimize the performance index in Eq. (6.37).

To solve this optimal control problem we chose $P = 30$ time stages of equal length, and an integration time step length of $1/45$, $\gamma = 0.95$, $\eta = 0.85$, with 30 iterations in each pass, and allowed 50 passes. The initial value for each control was taken as 0, and the initial region size for each control was 0.5. It was found that more than 7 grid points were necessary, even when 45 randomly chosen points were used. When $R = 5$

Table 6.7: Values of the performance index obtained after 50 passes, each consisting of 30 iterations

Number of grid points	Number of random points			
N	$R = 5$	$R = 15$	$R = 30$	$R = 45$
3	0.02640	0.02704	0.02701	0.02706
5	0.02560	0.02666	0.02622	0.02588
7	0.02564	0.02396	0.02358	0.02344
9	0.02408	0.02333	0.02328	**0.02327**
11	0.02347	**0.02327**	**0.02327**	**0.02327**

was chosen, then even 11 grid points were not enough. The relationship between the number of grid points and the number of randomly chosen points to yield the minimum performance index of $I = 0.02327$ is shown in Table 6.7. It is interesting to note that when a very small number of grid points was used, then better results were actually obtained with a smaller number of randomly chosen points R. However, the trend is quite clear that the larger the number of randomly chosen values for control, the smaller is the number of grid points required to obtain convergence to the minimum I.

The optimal piecewise constant control policy is given in Figure 6.1 and the state trajectories are given in Figure 6.2.

6.4 Further example

To show advantages of picking candidates for control variables at random, Bojkov and Luus [1] used the example presented by Nagurka *et al.* [10], where the nth-order linear time-invariant system is described by

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{u}(t) \tag{6.39}$$

with the initial state

$$\mathbf{x}(0) = [\quad 1 \quad 2 \quad \cdots \quad n \quad]^T \tag{6.40}$$

where the coefficient matrix is given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & -2 & 3 & \cdots & (-1)^{n+1}n \end{bmatrix}.$$

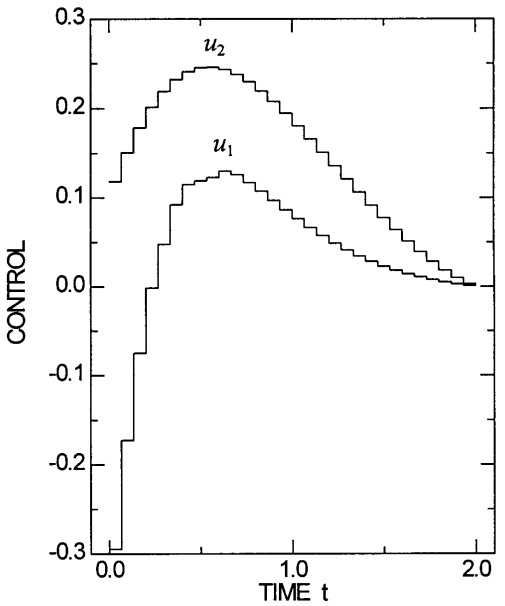

Figure 6.1: Optimal control policy for the two-stage CSTR system using 30 stages of piecewise constant control

The problem is to find the control vector $\mathbf{u}(t)$ in the time interval $0 \leq t < 1$ that minimizes the performance index

$$I = 10\mathbf{x}^T(1)\mathbf{x}(1) + \int_0^1 (\mathbf{x}^T\mathbf{x} + \mathbf{u}^T\mathbf{u})dt. \tag{6.41}$$

To put the problem into standard form, we introduce an additional variable by the differential equation

$$\frac{dx_{n+1}}{dt} = \mathbf{x}^T\mathbf{x} + \mathbf{u}^T\mathbf{u} \tag{6.42}$$

with the initial condition $x_{n+1}(0) = 0$, so the performance index to be minimized is in the standard form

$$I = 10\mathbf{x}^T(1)\mathbf{x}(1) + x_{n+1}(1) \tag{6.43}$$

where the final time $t_f = 1$ has been used. We are interested in establishing the optimal control for this system when the number of state variables and the number of control variables n is chosen to be 20. Here the goal is to show the effects of the region restoration factor η, the region contraction factor γ, and the number of random points chosen at each grid point R on the convergence of IDP, while using a single grid point ($N = 1$). The state equation is linear and the performance index is quadratic, so the problem is relatively simple.

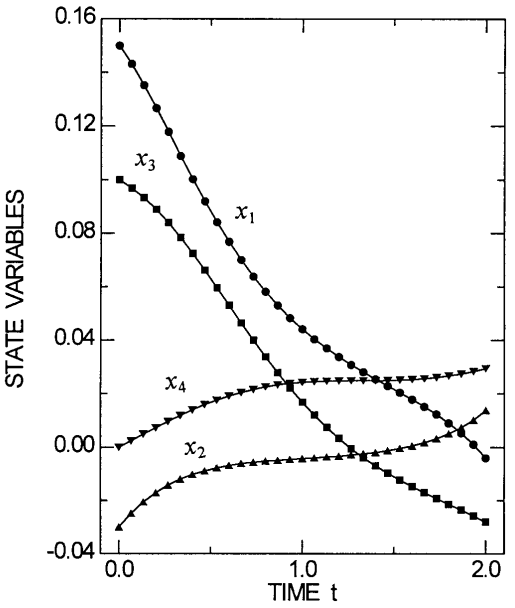

Figure 6.2: State trajectories for the two-stage CSTR system resulting from the application of the control policy in Figure 6.1

6.4.1 Effect of region restoration factor η

To keep the computational requirements reasonably small and yet have a control policy which approximates the continuous control policy, we choose $P = 40$ time stages of equal length and seek a piecewise constant control policy. A reasonable choice for the region restoration factor η is 0.90. Choosing the region contraction factor to be $\gamma = 0.95$ causes the region to be reduced quite gradually and after 30 iterations the region is 0.215 of the value at the beginning ($(0.95)^{30} = 0.215$). We choose 30 iterations in each pass. The initial region size for each control variable at the beginning of the first pass is chosen as 20, and the initial value for each control variable is chosen as 0. For integration we use the fourth-order Runge-Kutta method with an integration step-size of 0.0125, which gives accuracy to three decimal places for this problem.

As is seen in Figure 6.3, convergence to $I = 6226.047$ is readily obtained with the use of $\eta = 0.70$ and $\gamma = 0.95$. With the use of $R = 3$, convergence was obtained in 27 passes, each consisting of 30 iterations. The use of $R = 5$ reduced the number of passes required for convergence to 24, but the computation time for 30 passes was increased from 104 s to 158 s on a PentiumII/350 digital computer.

When η was reduced to 0.65, as is shown in Figure 6.4, premature collapse of the search region occurs when $R = 3$ is used. However, with the use of $R = 5$ there are no convergence difficulties and the use of $R = 11$ improves the convergence rate only marginally, but doubles the computation time to 343 s for 30 passes.

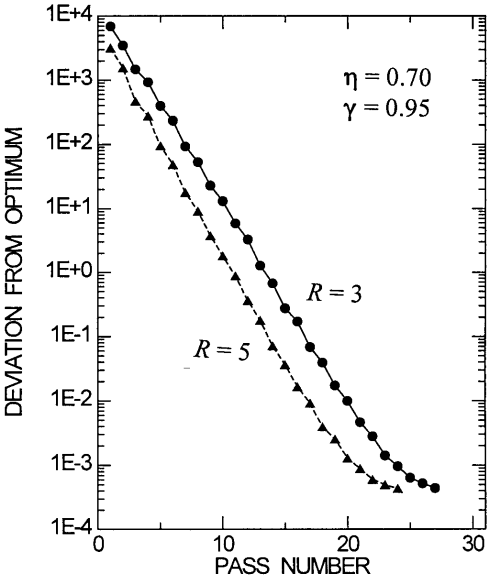

Figure 6.3: Convergence profile of IDP, showing the deviation from the optimum $I = 6226.047$ with $\eta = 0.70$ and $\gamma = 0.95$ using the number of randomly chosen points R as parameter

When η is increased to 0.75, the convergence rate is slightly slower than with $\eta = 0.70$, as is shown in Figure 6.5.

6.4.2 Effect of the region contraction factor γ

As is the case for the region restoration factor η, there is also a range for the region contraction factor γ for which reasonable rate of convergence is obtained with a small number of randomly chosen values for control. As is shown in Figure 6.6, when using $\eta = 0.70$, there is a range $0.93 < \gamma < 0.98$ for which convergence is obtained with the use of $R = 3$.

6.4.3 Effect of the number of time stages

The optimal control policy obtained for this example is shown for typical control variables in Figure 6.7 and the state trajectories for some typical state variables are given in Figure 6.8. It is seen that the control policy is very smooth. The smoothness is increased by using a larger number of time stages, yielding a better approximation to the continuous control policy.

By doubling the number of time stages to $P = 80$, we obtained the minimum performance index of $I = 6225.568$. This result was obtained with the use of $R = 3$ with $\eta = 0.70$ and $\gamma = 0.95$ in 24 passes, each consisting of 30 iterations as before.

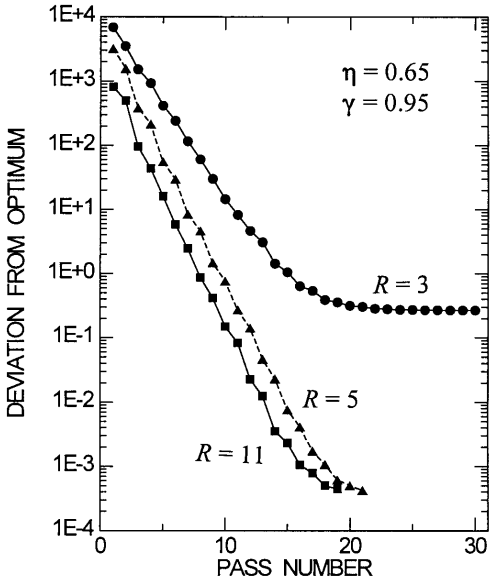

Figure 6.4: Convergence profile of IDP, showing the deviation from the optimum $I = 6226.047$ with $\eta = 0.65$ and $\gamma = 0.95$ using the number of randomly chosen points R as parameter

The computation time for 30 passes with 80 stages was 200.5 s. The doubling of the number of time stages did not slow the convergence rate at all, but the computation time was doubled.

The use of a smaller number of stages, namely, $P = 30$ stages, gave $I = 6226.543$. By using the same parameters, the computation time was 80.6 s for 30 passes.

Since the state equation is linear and the performance index is quadratic, the problem is ideally suited for solution by the use of Pontryagin's maximum principle, where the optimal control policy is obtained by minimizing the Hamiltonian. This leads to the $(n \times n)$ Riccati equation [6]

$$\frac{d\mathbf{J}}{dt} + \mathbf{JA} + \mathbf{A}^T\mathbf{J} - 0.5\mathbf{JJ} + 2\mathbf{I} = \mathbf{0} \tag{6.44}$$

with the final condition

$$\mathbf{J}(t_f) = 20\mathbf{I}. \tag{6.45}$$

The Riccati equation is solved backwards, yielding the symmetric matrix $\mathbf{J}$, which is stored at the end of each integration time step. Then the optimal control policy is obtained by integrating the state equation forward in time and using the stored values of $\mathbf{J}$

$$\mathbf{u} = -0.5\mathbf{J}\mathbf{x}(t). \tag{6.46}$$

To ensure sufficient accuracy and stability in the integration we used the integration subroutine DVERK [4] with local error tolerance of 10^{-4}. The accuracy of the

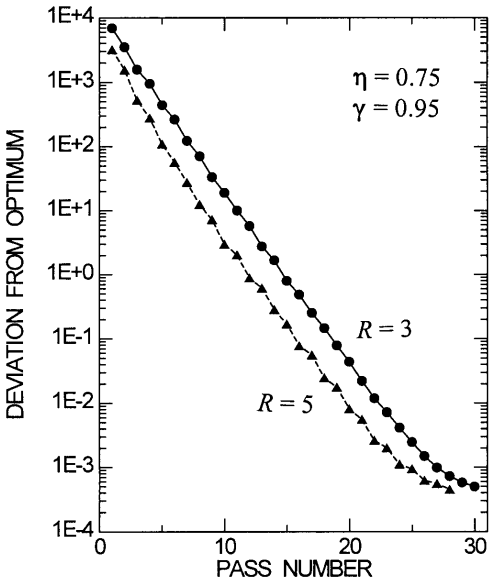

Figure 6.5: Convergence profile of IDP, showing the deviation from the optimum $I = 6226.047$ with $\eta = 0.75$ and $\gamma = 0.95$ using the number of randomly chosen points R as parameter

solution depends on the number of time stages at which the feedback matrices are stored, but generally at least 100 time stages are required for a good approximation. Here we will illustrate the effect of the number of stages by taking a different number of stages. This also enables us to see the relationship of this standard method to the results obtained by IDP. The computations are very fast, even when 200 stages are used, as is shown in Table 6.8.

It is obvious that the solution of the Riccati equation here with $n = 20$ is considerably faster than IDP, but for the same number of stages the use of IDP gives slightly better results. In the next chapter we will see that when piecewise linear

Table 6.8: Values for the performance index obtained by solving the Riccati equation

Number of stages P	Performance index I	Computation time on PentiumII/350, s
30	6228.42	1.60
40	6227.13	1.92
80	6225.85	3.24
150	6225.53	5.76
200	6225.48	7.69

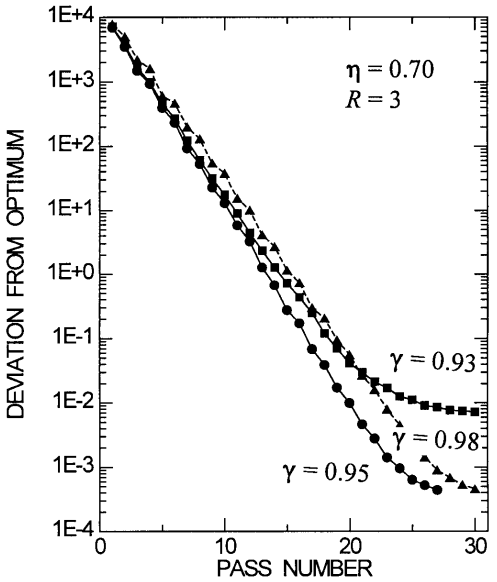

Figure 6.6: Convergence profile of IDP, showing the deviation from the optimum $I = 6226.047$ with $\eta = 0.70$ and 3 randomly chosen points, with region contraction factor γ used as a parameter

control is used in IDP and the dimensionality increases, then the computation time for IDP becomes quite comparable to the computation time required to solve the Riccati equation.

When these values of the performance index are plotted against $1/P^2$, as shown in Figure 6.9, the extrapolated value of the linear plots gives 6225.4, which is the same value as reported by Nagurka *et al.* [10]. It is quite clear that piecewise constant control is not the best choice when the control policy is very smooth. Rather than using a large number of stages with piecewise constant control, greater accuracy can be gained by using piecewise linear continuous control for such situations. The benefits in using piecewise linear continuous control for very high dimensional systems are illustrated in Chapter 7.

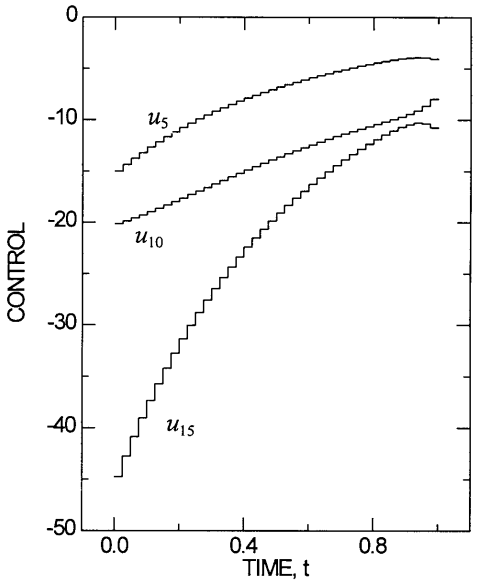

Figure 6.7: Optimal control policy, showing the piecewise constant control for typical control variables with $P = 40$ stages

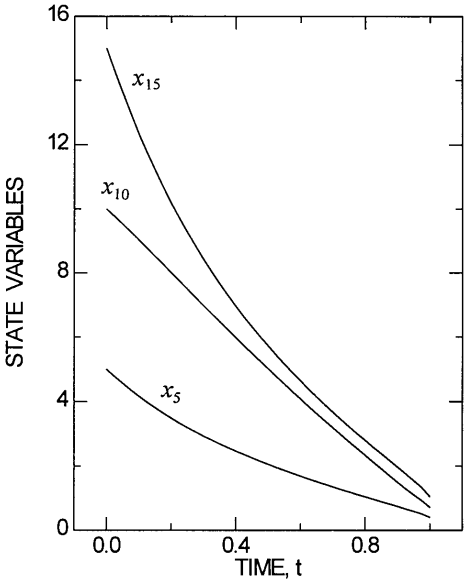

Figure 6.8: State trajectories for typical state variables resulting from the optimal control policy

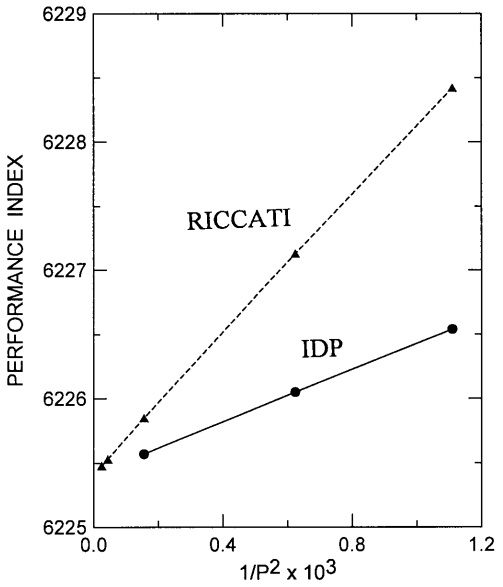

Figure 6.9: Comparison of results obtained by IDP and those obtained by solving the Riccati equation, showing a linear variation of the performance index with $1/P^2$ with the extrapolated value of $I = 6225.4$

6.5 References

[1] Bojkov, B. and Luus, R. : "Use of random admissible values for control in iterative dynamic programming", *Ind. Eng. Chem. Res.* **31** (1992), 1308-1314.

[2] Bojkov, B. and Luus, R.: "Evaluation of the parameters used in iterative dynamic programming", *Can. J. Chem. Eng.* **71** (1993), 451-459.

[3] Hartig, F., Keil, F.J., and Luus, R.: "Comparison of optimization methods for a fed-batch reactor", *Hung. J. Ind. Chem.* **23** (1995), 141-148.

[4] Hull, T.E., Enright, W.D., and Jackson, K.R.: *User Guide to DVERK– a Subroutine for Solving Nonstiff ODE's.* Report 100 (1976), Department of Computer Science, University of Toronto, Canada.

[5] Lapidus, L. and Luus, R.: *Optimal Control of Engineering Processes*, Blaisdell, Waltham, Mass. (1967), pp. 61-64.

[6] Luus, R. and Bojkov, B.: "Global optimization of the bifunctional catalyst problem", *Can. J. Chem. Eng.* **72** (1994), 160-163.

[7] Luus, R., Dittrich, J., and Keil, F.J. : "Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor", *Can. J. Chem. Eng.* **70** (1992), 780-785.

[8] Luus, R., Zhang, X., Hartig, F., and Keil, F.J. : "Use of piecewise linear control continuous optimal control for time-delay systems", *Ind. Eng. Chem. Res.* **34** (1995), 4136-4139.

[9] Mutharasan, R. and Luus, R.: "Analysis of time-delay systems by series approximation", *AIChE J.* **21** (1975), 567-571.

[10] Nagurka, M., Wang, S., and Yen, V.: "Solving linear quadratic optimal control problems by Chebychev-based state parameterization", *Proceedings of the 1991 American Control Conference*, Boston, MA; American Control Council, IEEE Service Center: Piscataway, NJ, 1991; pp. 104-109.

[11] Oh, S.H. and Luus, R.: "Optimal feedback control of time-delay systems", *AIChE J.* **22** (1975), 144-147.

[12] Rosen, O. and Luus, R.: "Evaluation of gradients for piecewise constant optimal control", *Comput. Chem. Eng.* **15** (1991), 273-281.

Chapter 7

Piecewise linear control

7.1 Introduction

Up to now, in using IDP to solve an optimal control problem, the given time interval was divided into P time stages of equal length and at each time-stage the control would be held constant. Such a piecewise constant control policy provides a reasonably good approximation to the continuous control policy, and the approximation can be improved by taking a larger number of time stages, as we saw with the last example in Chapter 6. In many situations the optimal control policy is quite smooth, and therefore we may get very good approximation by using piecewise linear, rather than constant, sections. We furthermore impose the constraint that the linear sections form a continuous curve. Piecewise linear continuous control was first used in IDP for several systems by Luus [4]. It was found that the use of piecewise linear control does indeed give an excellent result with a relatively small number of time stages, and allows the optimal control policy for very high dimensional systems to be determined accurately [3,5].

7.2 Problem formulation

Let us consider the system described by the vector differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{7.1}$$

with $\mathbf{x}(0)$ given, where $\mathbf{x}$ is an $(n \times 1)$ state vector and $\mathbf{u}$ is an $(m \times 1)$ control vector bounded by

$$\alpha_j \leq u_j \leq \beta_j, \qquad j = 1, 2, \cdots, m. \tag{7.2}$$

Associated with the system is a performance index

$$I = \phi[\mathbf{x}(t_f)] \tag{7.3}$$

where ϕ is a positive definite function of $\mathbf{x}$ at the given final time t_f. The optimal control problem is to find the control policy $\mathbf{u}$(t) in the time interval $0 \leq t < t_f$ so that the performance index is minimized.

Let us divide the given time interval $(0, t_f)$ into P subintervals $(0, t_1), (t_1, t_2), \cdots, (t_k, t_{k+1}), \cdots, (t_{P-1}, t_P)$, each of length L, so that

$$L = \frac{t_f}{P}. \tag{7.4}$$

We seek a piecewise linear continuous control policy to minimize the performance index. Therefore, we calculate the control policy in the time interval (t_k, t_{k+1}) by the expression

$$\mathbf{u}(t) = \mathbf{u}(k) + \frac{\mathbf{u}(k+1) - \mathbf{u}(k)}{L}(t - t_k) \tag{7.5}$$

where $\mathbf{u}(k)$ is the value of $\mathbf{u}$ at the time t_k and $\mathbf{u}(k+1)$ is the value of $\mathbf{u}$ at time t_{k+1}. The optimal control problem is now to find $\mathbf{u}(k), k = 0, 1, 2, \cdots, P$, inside the range specified by Eq. (7.2), so that the performance index in Eq. (7.3) is minimized. At the last stage we must determine both $\mathbf{u}(P)$ and $\mathbf{u}(P-1)$. This means that for the last stage we must determine $2m$ values for the m control variables. For all the other stages there are only m values to be determined because of the continuity restriction.

7.3 Algorithm for IDP for piecewise linear control

Here we use IDP with a single grid point, i.e., $N = 1$, in a multi-pass fashion, where the region size is contracted by an amount γ after every iteration. After a specified number of iterations making up a pass, the region size is restored to a value η times the region size at the beginning of the pass to give the region size to be used in the subsequent pass. To illustrate this underlying logic in IDP, an algorithm is given to solve the optimal control problem as outlined in Eq.(7.1)-Eq.(7.4), where it is required to minimize the performance index in Eq. (7.3) with the use of piecewise linear control, as specified in Eq. (7.5) over P stages, each of the same length:

1. Divide the time interval $[0, t_f]$ into P time stages, each of length L.

2. Choose the number of test values for $\mathbf{u}$ denoted by R, an initial control policy and the initial region size $\mathbf{r}_{in}$; also choose the region contraction factor γ used after every iteration and the region restoration factor η used after every pass.

3. Choose the number of iterations to be used in every pass and the number of passes to be used in the run.

4. Set the pass number index $q = 1$ and the iteration number index to $j = 1$.

5. Set the region size vector $\mathbf{r}^j = \eta^q \mathbf{r}_{in}$.

6. By using the best control policy (the initial control policy for the first iteration of the first pass), integrate Eq. (7.1) from $t = 0$ to t_f to generate the $\mathbf{x}$-trajectory and

store the value of $\mathbf{x}$ at the beginning of each time stage, so that $\mathbf{x}(k-1)$ corresponds to the value of $\mathbf{x}$ at the beginning of stage k.

7. Starting at stage P, corresponding to time $t_f - L$, integrate Eq. (7.1) from $t_f - L$ to t_f, using as the initial state the stored value $\mathbf{x}(P-1)$ from step 6, once with each of the R allowable values for the control vector calculated from Eq. (7.5), using

$$\mathbf{u}(P-1) = \mathbf{u}^{*j}(P-1) + \mathbf{D}_1\mathbf{r}^j \tag{7.6}$$

$$\mathbf{u}(P) = \mathbf{u}^{*j}(P) + \mathbf{D}_2\mathbf{r}^j \tag{7.7}$$

where $\mathbf{u}^{*j}(P-1)$ and $\mathbf{u}^{*j}(P)$ are the best values obtained in the previous iteration and $\mathbf{D}_1$ and $\mathbf{D}_2$ are diagonal matrices of different random numbers between -1 and 1. Clip suitably any of the control variables at α_j or β_j to satisfy Eq. (7.2). Out of the R values for the performance index, choose the control values that give the minimum value, and store these values as $\mathbf{u}^*(P-1)$ and $\mathbf{u}^*(P)$. We now have the best control for the last stage.

8. Step back to stage $P-1$, corresponding to time $t_f - 2L$. Choose R values for $\mathbf{u}(P-2)$ as in the previous step, and by taking as the initial state $\mathbf{x}(P-2)$ integrate Eq. (7.1) over one stage length by using Eq. (7.5) for the control. Continue integration over the last time stage by using the stored value of $\mathbf{u}^*(P-1)$ and $\mathbf{u}^*(P)$from step 7. Compare the R values of the performance index and store the best control as $\mathbf{u}^*(P-2)$ that gives the minimum value for the performance index.

9. Continue the procedure until stage 1, corresponding to the initial time $t = 0$ and the given initial state, is reached. As before, integrate Eq. (7.1) and compare the R values of the performance index and store the control $\mathbf{u}(0)$ as $\mathbf{u}^*(0)$ that gives the minimum performance index. Store also the corresponding $\mathbf{x}$-trajectory.

10. Reduce the region size for allowable control

$$\mathbf{r}^{j+1} = \gamma\mathbf{r}^j \tag{7.8}$$

where j is the iteration number index. Use the best control policy from step 9 as the midpoint for the allowable values for the control denoted by the superscript *.

11. Increment the iteration index j by 1 and go to step 6; continue the procedure for the specified number of iterations to finish a pass.

12. Increment the pass index q by 1 and go to step 5; continue the procedure for the specified number of passes, and interpret the results.

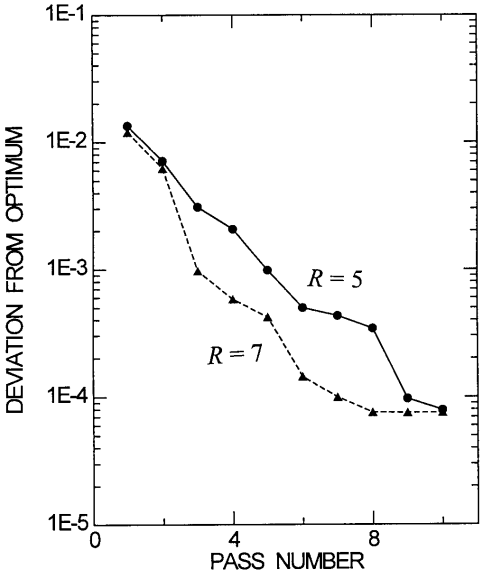

Figure 7.1: Convergence of IDP for nonlinear CSTR showing deviation of the performance index from the optimum $I = 0.133094$

7.4 Numerical examples

To show the advantages of using piecewise linear control, we consider several examples. As a guide to the amount of computational effort required, the computation time on a PentiumII/350 is reported.

7.4.1 Nonlinear CSTR

The nonlinear CSTR discussed in Chapter 1 provides a good test problem for optimization procedures. It has been used by Lapidus and Luus [2], Rao and Luus [9], Luus and Cormack [6], and recently by Luus and Galli [7] to examine the multiplicity of solutions with IDP. The equations describing the chemical reactor are

$$\frac{dx_1}{dt} = -(2+u)(x_1+0.25) + (x_2+0.5)\exp[\frac{25x_1}{x_1+2}] \tag{7.9}$$

$$\frac{dx_2}{dt} = 0.5 - x_2 - (x_2+0.5)\exp[\frac{25x_1}{x_1+2}] \tag{7.10}$$

$$\frac{dx_3}{dt} = x_1^2 + x_2^2 + 0.1u^2 \tag{7.11}$$

where x_1 represents deviation from dimensionless steady-state temperature and x_2 represents deviation from dimensionless steady-state concentration. In the present work, the initial conditions $x_1(0) = 0.09$ and $x_2(0) = 0.09$ are used. The variable

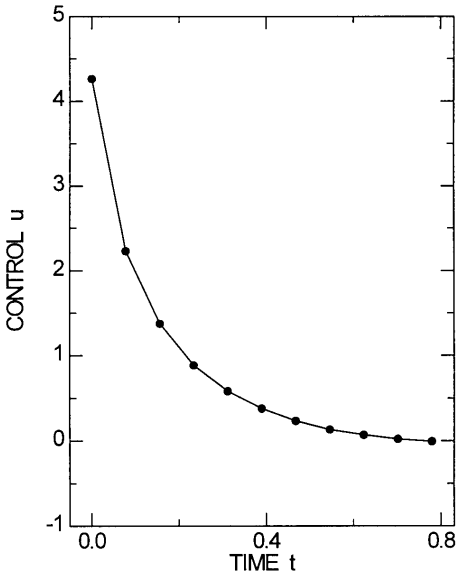

Figure 7.2: Piecewise linear continuous optimal control policy with $P = 10$ for the nonlinear CSTR, giving $I = 0.133169$

x_3, with initial condition $x_3(0) = 0$, is introduced to put the optimal control problem into the standard form, so the performance index to be minimized is

$$I = x_3(t_f) \tag{7.12}$$

where the final time $t_f = 0.78$. The control u is unbounded.

Using control vector iteration procedure based on Pontryagin's maximum principle, Luus and Cormack [6] showed that there exists a local optimum of $I = 0.244425$ and a global optimum of $I = 0.133094$. The above algorithm for piecewise linear continuous control with IDP, using $P = 10$ time stages, initial control policy $u^{(0)} = 2$ and initial region size $r^{(0)} = 2$, with region contraction factor $\gamma = 0.85$ and region restoration factor $\eta = 0.70$, using 10 passes, each pass consisting of 10 iterations, yielded rapid convergence to the global optimum $I = 0.133169$ with the use of $R = 5$ and $R = 7$, as is shown in Figure 7.1. The use of $R = 3$ and $R = 10$ gave convergence to the local optimum. The problem of convergence to the local optimum was explored in detail by Luus and Galli [7]. Here we want to emphasize that taking a large number of random points R does not necessarily guarantee getting the global optimum. The computation time for 10 passes, using the fourth order Runge-Kutta method with integration time-step length of 0.00975, was 1.32 s with $R = 5$ and 1.70 s with $R = 7$ on a PentiumII/350.

The resulting optimal control policy is given in Figure 7.2 and the state trajectories are given in Figure 7.3. It is interesting to note that at the final time $t_f = 0.78$ the state variable x_2 is further from the desired state (origin) than at the initial time.

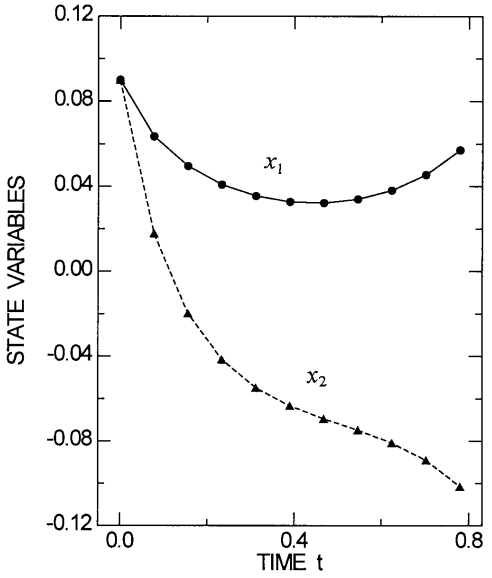

Figure 7.3: Trajectories of the state variables of the nonlinear CSTR

This shows the need for specifying state constraints, or changing the performance index if the desired state is to be reached within some tolerance. Handling of state constraints is considered in Chapter 11.

To get a more refined answer, a larger number of time stages was used. In order to get the global and not the local optimum, the initial control policy and the initial region size were increased to $u^{(0)} = 2.5$ and $r^{(0)} = 3$, and 20 passes, each consisting of 10 iterations, were used. The use of $P = 20$ stages yielded a slight improvement of the performance index to $I = 0.133101$, which is within 0.005% of the optimum. The use of $P = 40$ stages yielded a further improvement to $I = 0.133096$, which is within 0.001% of the optimum obtained by Luus and Cormack [6] with the second variation method. The computation time for 20 passes with $P = 20$ stages was 6.3 s and with $P = 40$ the computation time was 11.3 s.

7.4.2 Nondifferentiable system

We now consider the system used by Thomopoulos and Papadakis [10], who showed difficulties of convergence of several optimization procedures. The system, also considered by Luus [4], is described by

$$\frac{dx_1}{dt} = x_2 \tag{7.13}$$

$$\frac{dx_2}{dt} = -x_1 - x_2 + 100[H(t-0.5) - H(t-0.6)] \tag{7.14}$$

$$\frac{dx_3}{dt} = 5x_1^2 + 2.5x_2^2 + 0.5u^2 \tag{7.15}$$

with the initial state

$$\mathbf{x}(0) = [\ 0 \quad 0 \quad 0\ ]^T. \tag{7.16}$$

In Eq. (7.14) there is a disturbance term in the form of a rectangular pulse of magnitude 100 from $t = 0.5$ until $t = 0.6$. The optimal control problem is to find the control u in the time interval $0 \leq t < 2$ so that the performance index

$$I = x_3(t_f) \tag{7.17}$$

is minimized. Here the final time $t_f = 2$.

By using an exponential smoothing technique Thomopoulos and Papadakis [10] overcame the discontinuity problem and obtained a minimum value $I = 58.538$. With IDP the discontinuity produced by the rectangular pulse causes no difficulties provided that the integration step size is chosen so that the state equations can be properly integrated. By using IDP, Luus [4] reported a minimum of $I = 58.20$ by using 40 stages of piecewise constant control and $I = 58.18$ with the use of piecewise linear control over 40 time stages of equal length. Here we consider this problem with 20 stages of equal length.

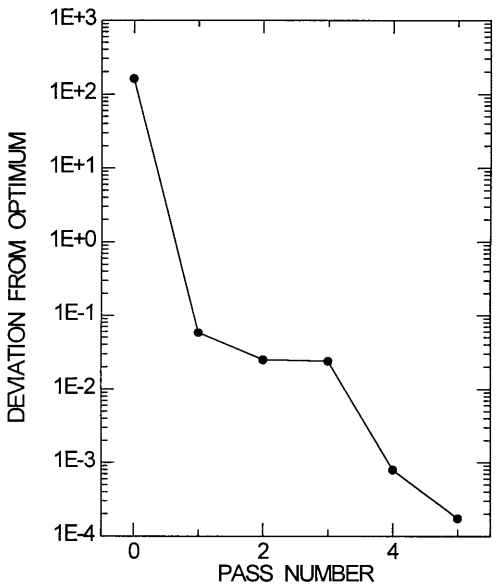

Figure 7.4: Convergence profile for the nondifferentiable system, showing the deviation of the performance index from 58.185 as a function of the pass number with $P = 20$

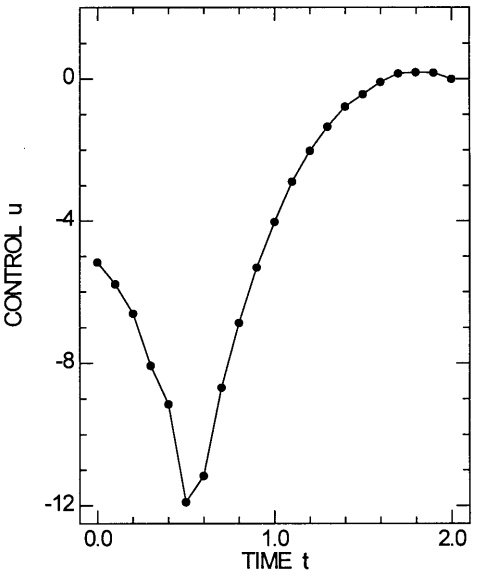

Figure 7.5: Piecewise linear continuous optimal control policy with $P = 20$ for the nondifferentiable system

For IDP we chose $P = 20$, $u^{(0)} = 0$, $r^{(0)} = 10$, $\gamma = 0.95$, $\eta = 0.7$, and 20 iterations per pass. As is shown in Figure 7.4, convergence from the initial value of 220.395 to the minimum value of $I = 58.185$ was obtained very rapidly. For integration of the equations we used the Runge-Kutta method with an integration step-size of 0.025. The computation time for the 5 passes was 0.98 s.

The resulting piecewise linear optimal control policy is given in Figure 7.5 and the state trajectories are given in Figure 7.6. It is observed that the rectangular pulse in the time interval from 0.5 to 0.6 causes a very rapid increase in the state variable x_2 during this time interval.

7.4.3 Linear system with quadratic performance index

In Chapter 6 we considered the mathematical problem presented by Nagurka *et al.* [8] to show the determination of piecewise constant control policy and concluded that the solution of this problem by solving the Riccati equation was much faster than IDP when the number of state variables and control variables was 20. Here we consider this problem again, but use piecewise linear control. Since a fewer number of stages are required, we shall show that the computational disadvantage of IDP decreases as the number of variables n increase. Therefore, let us consider the problem again where the nth-order linear time-invariant system is described by

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{u}(t) \tag{7.18}$$

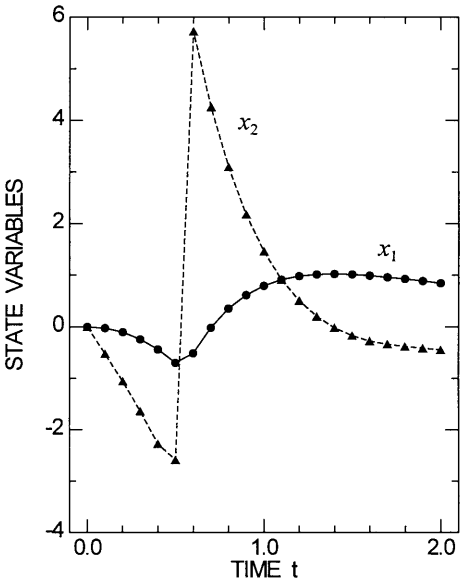

Figure 7.6: State trajectories for the nondifferentiable system

with the initial state

$$\mathbf{x}(0) = [\ 1 \quad 2 \quad \cdots \quad n\ ]^T \tag{7.19}$$

where the coefficient matrix is given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & -2 & 3 & \cdots & (-1)^{n+1}n \end{bmatrix}.$$

The problem is to find the control vector $\mathbf{u}(t)$ in the time interval $0 \leq t < 1$ that minimizes the performance index

$$I = 10\mathbf{x}^T(1)\mathbf{x}(1) + \int_0^1 (\mathbf{x}^T\mathbf{x} + \mathbf{u}^T\mathbf{u})dt. \tag{7.20}$$

Case 1: $n = 20$

Let us examine the convergence of IDP with $n = 20$ and $P = 10$ time stages. By choosing the initial control region size $\mathbf{r}_{in} = \mathbf{40}$, with $\gamma = 0.95$ and $\eta = 0.70$, and using 30 iterations per pass with $R = 7$ allowable test values for control, convergence from the initial value $\mathbf{u}^{(0)} = \mathbf{0}$ to the optimum $I = 6225.46$ was very regular and systematic as is shown in Figure 7.7. The computation time for 30 passes was 49.1 s. As is shown in the same figure, the rate of convergence is improved slightly by increasing

Table 7.1: Effect of the number of time stages P used for piecewise linear control on the optimal performance index

Number of stages P	Performance index I
7	6225.59
8	6225.53
9	6225.49
10	6225.46
11	6225.45
15	6225.42
30	6225.41
40	6225.41

the number of allowable control values to 11. When the region restoration factor η is increased to 0.75, as is shown in Figure 7.8, fewer number of allowable values for control may be used to get adequate convergence in 30 passes.

By solving the Riccati equation with $P = 200$ the minimum value of the performance index $I = 6225.48$ is obtained in 7.7 s on a PentiumII/350 personal computer. Although we can not come even close to this speed with IDP, it is still worthwhile to consider this case, especially to examine the number of time stages needed in the use of piecewise linear continuous control to get the same accuracy as the use of 200 stages for solving the Riccati equation.

As is seen in Table 7.1, the use of 9 stages gives almost the same value $I = 6225.48$ obtained with the Riccati equation with 200 stages. The use of 10 stages for piecewise linear control yields a better result than the optimum obtained by the solution of the Riccati equation. Also it is obvious that very little is gained by taking more than 15 stages.

The optimal control policy for the control variables u_1, u_{10} and u_{20} with $P = 10$ is given in Figure 7.9 and the state trajectories for x_1, x_{10} and x_{20} are given in Figure 7.10.

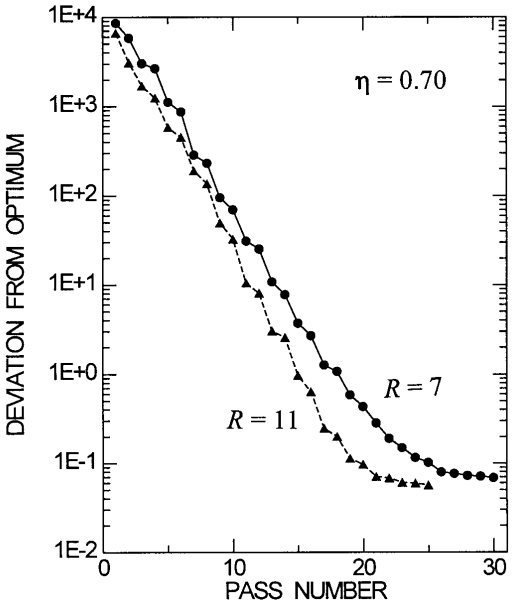

Figure 7.7: Convergence profile for Case 1 showing the deviation of the performance index from $I = 6225.41$ with $\eta = 0.70$

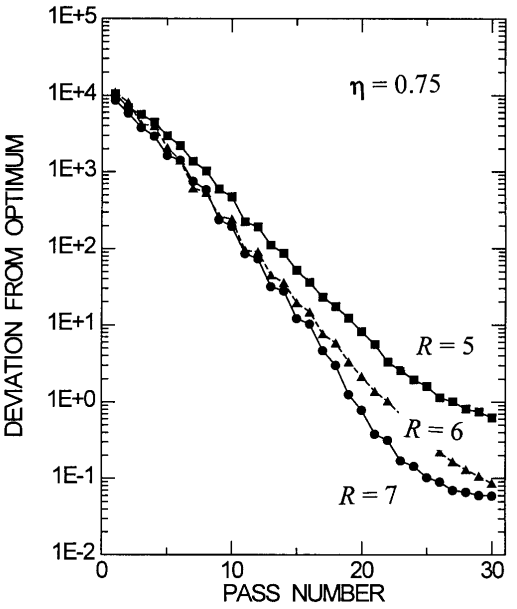

Figure 7.8: Convergence profile for Case 1 showing the deviation of the performance index from $I = 6225.41$ with $\eta = 0.75$

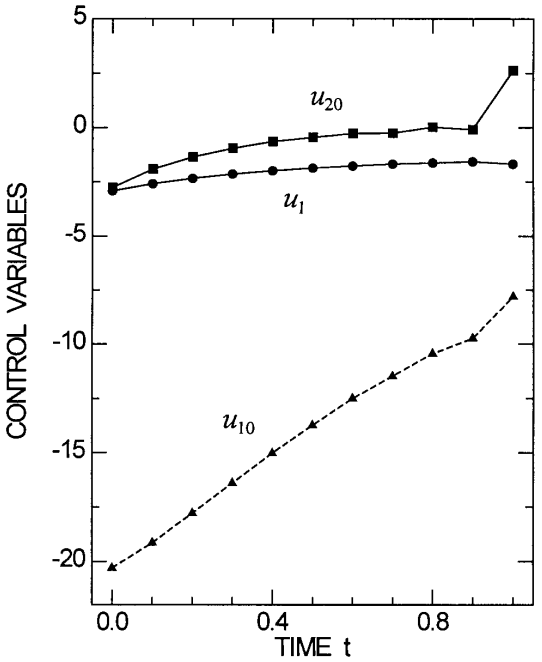

Figure 7.9: Optimal control policies with 10 linear continuous sections for u_1, u_{10} and u_{20} for Case 1 with $n = 20$

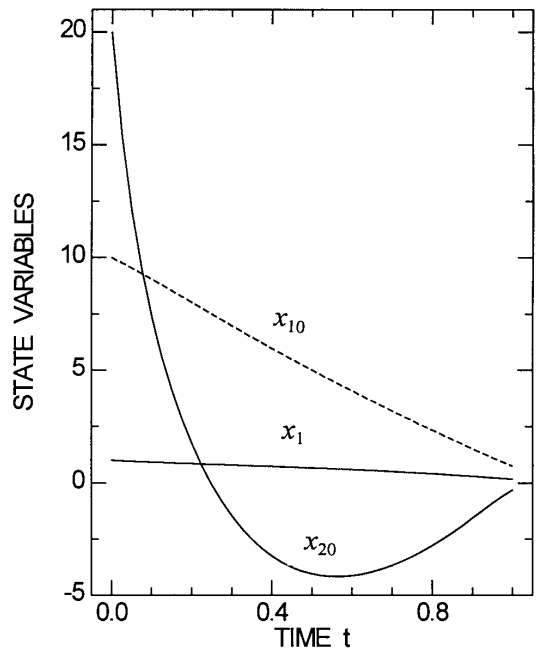

Figure 7.10: State trajectories for x_1, x_{10} and x_{20} for Case 1 with $n = 20$

Case 2: $n = 150$

When the number of state variables and control variables was increased to 150, some changes were made to cope with the larger number of variables. The number of iterations in a pass was increased to 50, and the region restoration factor η was kept closer to 0.90, but the region reduction factor γ was kept at 0.95. The number of time stages was chosen as $P = 11$, to give sufficient accuracy. To compensate for the larger values in the initial condition, the initial region size was chosen to be $\mathbf{r}_{in} = \mathbf{300}$. By taking $R = 25$ allowable values for control, it was found that convergence to $I = 2.81317 \times 10^6$ was regular and systematic as is shown in Figure 7.11. In each case a slightly better performance index was obtained after 70 passes than $I = 2.81318 \times 10^6$ obtained by solving the Riccati equation with $P = 200$. The computation time for 70 passes with IDP was 5776 s which compares favorably with 3515 s required to solve the optimal control problem through the Riccati equation with DVERK [1] subroutine.

The optimal control policy for typical control variables is given in Figure 7.12 and typical state trajectories are given in Figure 7.13.

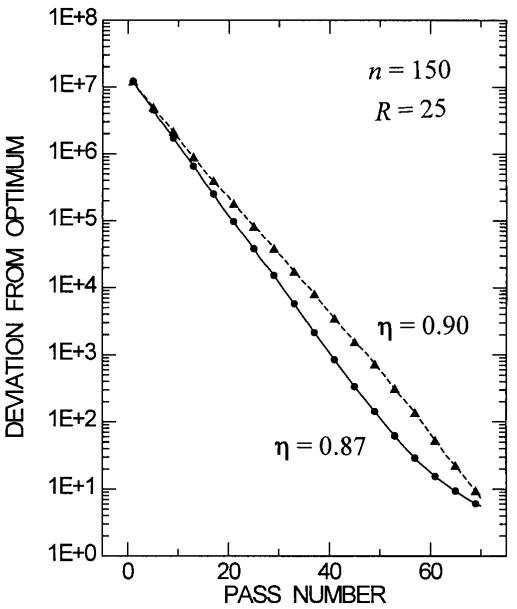

Figure 7.11: Convergence profile for Case 2 showing the deviation of the performance index from $I = 2813167.2$ as a function of the pass number for two values of the region restoration factor η

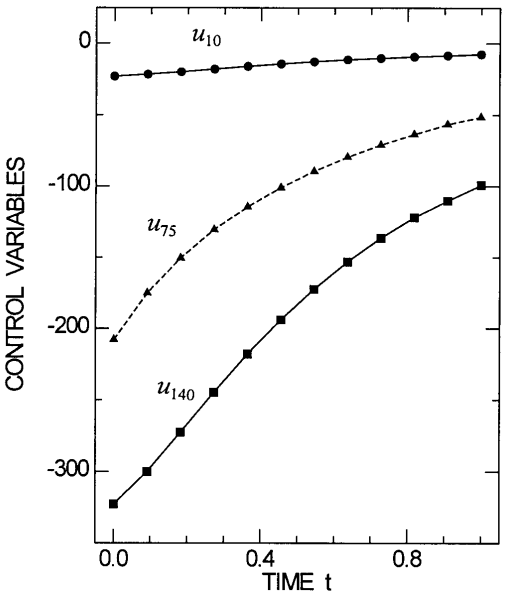

Figure 7.12: Optimal piecewise linear control policy for typical control variables with $P = 11$ stages for Case 2 with $n = 150$

Case 3: $n = 250$

Increasing the number of variables to 250 did not present any difficulties in the use of IDP. By taking $P = 11$ and choosing the initial region $\mathbf{r}_{in} = \mathbf{250}$, with $R = 40$ and 70 iterations per pass and $\gamma = 0.95$, convergence to $I = 1.31030 \times 10^7$ was systematic and regular for $\eta = 0.90$, but the use of $\eta = 0.89$ caused the region size to collapse around the 60th pass, giving $I = 1.31031 \times 10^7$. The convergence profiles are shown in Figure 7.14. One effective means of preventing the region from collapsing is to use initially a small value for η and then switch to a larger one. This scheme was used by Luus [5] who took for this problem $\eta = 0.89$ for the first 40 passes and then switched to $\eta = 0.91$ for the remainder of the passes to obtain convergence to $I = 1.31030 \times 10^7$. Typical optimal control profiles and typical state variables are shown in Figure 7.15 and 7.16. The computation time for 70 passes was 8.4 h on a PentiumII/350 personal computer.

Attempts to solve the Riccati equation for this problem on the PentiumII/350 with $P = 200$ and $P = 100$ caused memory problems, but with $P = 75$ the Riccati equation was solved in 4.1 h, yielding $I = 1.31037 \times 10^7$. Luus [5] solved the Riccati equation with $P = 200$ on a Pentium/120 to give $I = 1.31030 \times 10^7$ without any difficulty, except that the computation time was 55.0 h. No attempt was made to reduce the number of equations to be solved in the Riccati equation from 62500 to 32500 because the Riccati matrix is symmetric. For this case IDP is clearly preferred.

Also, in using IDP, the progress can be followed after every iteration and after every pass, whereas in solving the Riccati equation intermediate results are not available.

Although there is a systematic and regular reduction of the performance index from pass to pass, during each individual pass there are three distinct sections as is shown for pass 20 in Figure 7.17. In the first section, consisting of 13 iterations, the performance index does not change at all because the region over which the allowable values for control are taken is too large. In the next section consisting of 8 iterations the performance index becomes larger, reaching a peak at the 21st iteration. In the last section the performance index decreases in size to a value lower than the one at the start of the pass. The analysis of this convergence pattern allows one to determine whether the region size is adequate and whether a proper number of iterations have been used in a particular pass. This type of convergence profile was observed by Luus [5] for this system with $n = 100$. Further results are given in that paper. The listing of the computer program for this problem is given in Appendix D.

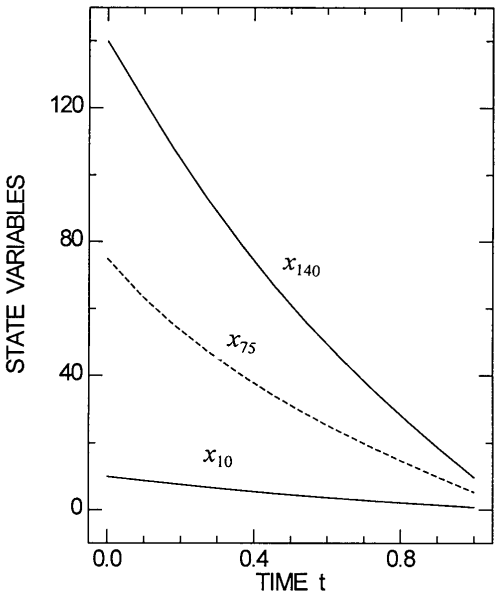

Figure 7.13: Typical state trajectories for Case 2 with $n = 150$

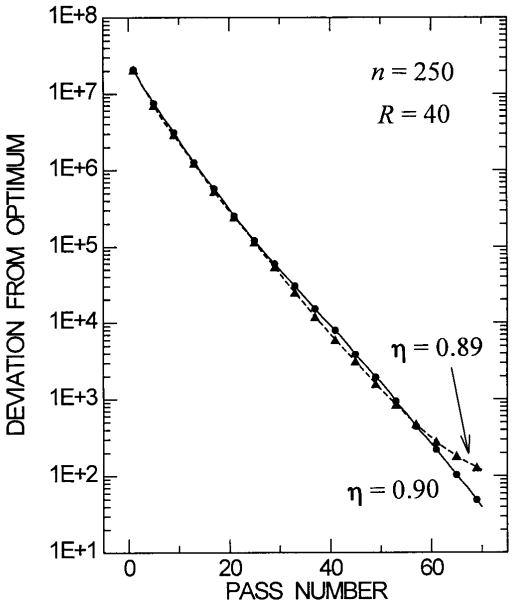

Figure 7.14: Convergence profile for Case 3 with $n = 250$ showing the variation of the deviation of the performance index from $I = 1.31030 \times 10^7$ for two values of the region restoration factor η

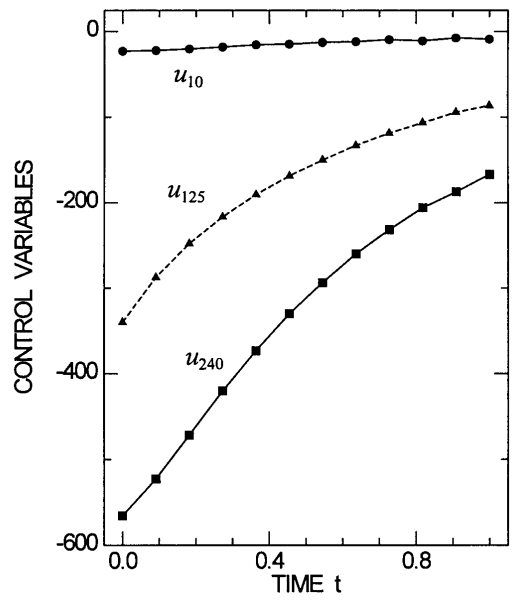

Figure 7.15: Optimal piecewise linear control policy with $P = 11$ stages for typical control variables for Case 3 with $n = 250$

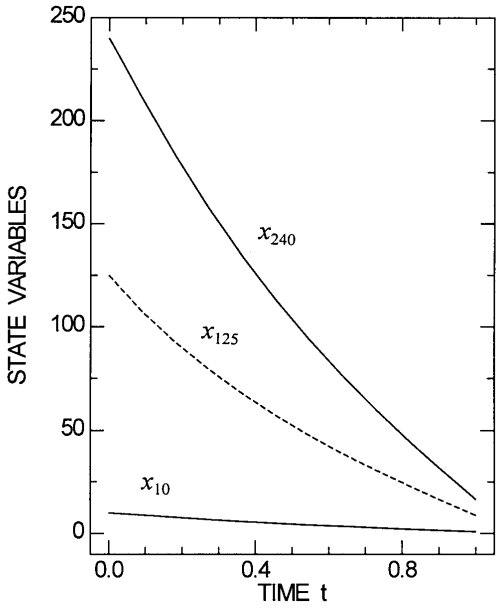

Figure 7.16: Typical state trajectories for Case 3 with $n = 250$

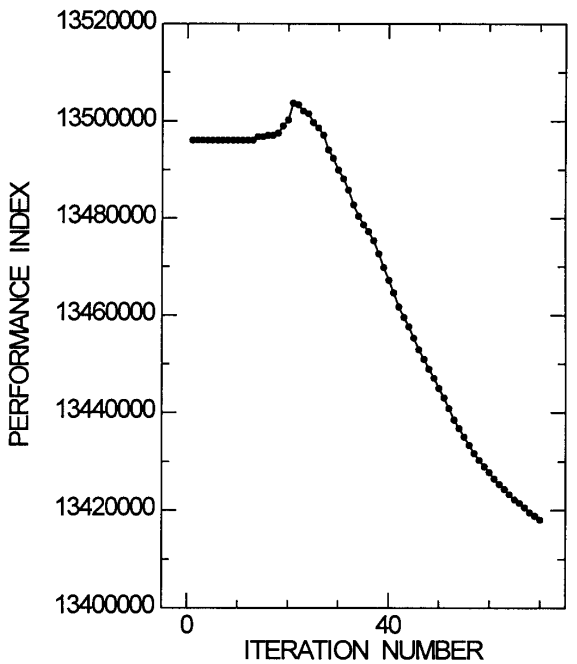

Figure 7.17: Convergence profile of IDP during the 20th pass

7.4.4 Gas absorber with a large number of plates

Let us consider the model of the n-plate gas absorber discussed in Chapter 1, where the model consists of the linear time-invariant system described by

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{7.21}$$

where $\mathbf{A}$ is a tridiagonal coefficient matrix given by

$$\mathbf{A} = \text{tridiag}[\ 0.538998 \quad -1.173113 \quad 0.634115\ ] \tag{7.22}$$

and the $(n \times 2)$ control coefficient matrix $\mathbf{B}$ is given by

$$\mathbf{B}^T = \begin{bmatrix} 0.538998 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0.634115 \end{bmatrix}. \tag{7.23}$$

The initial condition is chosen as

$$x_i(0) = -0.0307 - (\frac{i-1}{n-1})(0.1273 - 0.0307), \quad i = 1, 2, \cdots, n. \tag{7.24}$$

The performance index to be minimized is

$$I = \int_0^{t_f} (\mathbf{x}^T\mathbf{x} + \mathbf{u}^T\mathbf{u})dt, \tag{7.25}$$

where we choose the final time $t_f = 15$ min. We keep the number of plates n as a parameter to investigate the effect of increasing the dimensionality of the problem.

Since there are only two control variables, we expect rapid convergence with a very small number of randomly chosen values for control. For all the runs we choose $P = 20$, $R = 7$, $\mathbf{r}_{in} = 0.05, \mathbf{u}^{(0)} = \mathbf{0}$, $\gamma = 0.85$, $\eta = 0.25$, and allow 2 passes of 20 iterations each, for $n = 10$, 25, 50, and 100. For all the runs convergence was obtained within the two passes. Comparison of the computation times with IDP and the results obtained by solving the Riccati equation with $P = 100$ are shown in Table 7.2.

It is interesting to note that the computation time for IDP increases linearly with n, whereas with the Riccati equation the computation time goes up as n^3. Already with $n = 50$, IDP is 3 times faster than solving the Riccati equation. When $n = 100$, IDP is 25 times faster. Actually with $n = 100$, the solution of the Riccati equation yielded $I = 9.73327$ with $P = 100$. The optimum value of $I = 9.73326$ was obtained with $P = 200$, requiring 500.9 s of computation time.

The optimal control policy for $n = 100$ is given in Figure 7.18 and the trajectories for typical state variables are given in Figure 7.19.

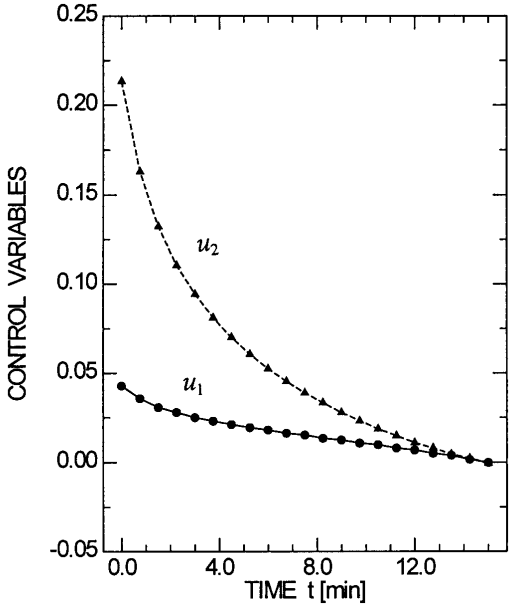

Figure 7.18: Optimal piecewise linear control policy with $P = 20$ stages for the 100-plate gas absorber

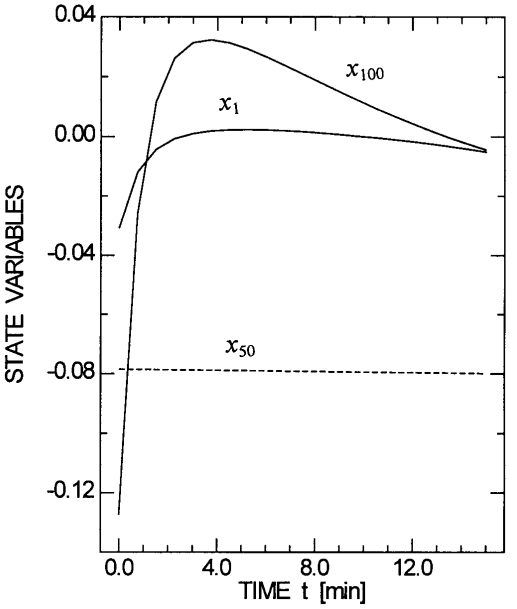

Figure 7.19: Typical state trajectories for the 100-plate gas absorber

Table 7.2: Comparison of IDP to solving the Riccati equation for the gas absorber problem

Number of plates n	Optimal Performance index I^o	CPU time for IDP s	CPU time for Riccati equation s
10	0.37196	2.9	0.3
25	1.86581	5.9	4.1
50	4.47877	10.9	34.1
100	9.73326	20.9	269.1

7.5 References

[1] Hull, T.E., Enright, W.D., and Jackson, K.R.: *User Guide to DVERK– a Subroutine for Solving Nonstiff ODE's*. Report 100 (1976), Department of Computer Science, University of Toronto, Canada.

[2] Lapidus, L. and Luus, R.: *Optimal Control of Engineering Processes*, Blaisdell, Waltham, Mass. (1967).

[3] Luus, R.: "Application of iterative dynamic programming to very high-dimensional systems", *Hung. J. Ind. Chem.* **21** (1993), 243-250.

[4] Luus, R.: "Piecewise linear continuous optimal control by iterative dynamic programming", *Ind. Eng. Chem. Res.* **32** (1993), 859-865.

[5] Luus, R.: "Numerical convergence properties of iterative dynamic programming when applied to high dimensional systems", *Chem. Eng. Res. Des.* **74** (1996), 55-62.

[6] Luus, R. and Cormack, D.E.: "Multiplicity of solutions resulting from the use of variational methods in optimal control problems", *Can. J. Chem. Eng.* **50** (1972), 309-312.

[7] Luus, R. and Galli, M.: "Multiplicity of solutions in using dynamic programming for optimal control", *Hung. J. Ind. Chem.* **19** (1991), 55-62.

[8] Nagurka, M., Wang, S., and Yen, V.: "Solving linear quadratic optimal control problems by Chebychev-based state parameterization", *Proceedings of the 1991 American Control Conference*, Boston, MA; American Control Council, IEEE Service Center: Piscataway, NJ, 1991; pp. 104-109.

[9] Rao, S.N. and Luus, R.: "Evaluation and improvement of control vector iteration procedures for optimal control", *Can. J. Chem. Eng.* **50** (1972), 777-784.

[10] Thomopoulos, S.C.A. and Papadakis, I.N.M.: "A single shot method for optimal step computation in gradient algorithms", *Proceedings of the 1991 American Control Conference*, Boston, MA; American Control Council, IEEE Service Center: Piscataway, NJ, 1991; pp. 2419-2422.

Chapter 8

Time-delay systems

8.1 Introduction

Engineering systems often contain delayed elements such as recycle streams, transportation lags, and time delays associated with the measurement of output variables. These time delays can have significant effect on the control of the system under consideration, so it is important to incorporate them into the mathematical model. This gives rise to differential equations with delayed arguments.

Due to the difficulty of solving the optimal control of nonlinear time-delay systems, Oh and Luus [8] suggested a suboptimal control approach based on the use of Taylor series expansion of the delayed state to retain sufficient dynamic features of the original time-delay system, and then using direct search to obtain the best feedback gain matrix in the sense of minimizing the associated performance index. For small time delays this approach gave very good results.

With IDP there are no conceptual difficulties, since no equations have to be integrated backwards. The first attempt to use IDP with time-delay systems was made by Dadebo and Luus [3], who used piecewise constant control policy, and obtained good results with several problems. In the present chapter, instead of using only piecewise constant control, we wish to use also piecewise linear continuous control, so that better results could be obtained with a fewer number of time stages. Here we will use a single grid point ($N = 1$). If a single grid point is used, then the problem of how to determine the initial profiles, which are required for the delay terms, does not arise. The only difficulty is that of determining sufficiently good starting conditions for highly nonlinear systems to ensure convergence. The recent work by Luus *et al.* [7] shows that a good way of overcoming such difficulty is to use the Taylor series expansion for the delay terms to convert the given system into a nondelay system for which the optimal control policy can be readily established. Then the optimal control policy for the nondelay system provides the starting control policy for the time-delay system. We will illustrate this approach with the last example.

8.2 Problem formulation

Let us consider the system described by the vector differential equation with a constant time delay τ

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}[\mathbf{x}(t), \mathbf{x}(t-\tau), \mathbf{u}(t)] \tag{8.1}$$

with the given initial state profile

$$\mathbf{x}(t) = \mathbf{g}(t), \quad -\tau \leq t < 0 \tag{8.2}$$

and the initial condition

$$\mathbf{x}(0) = \mathbf{k}. \tag{8.3}$$

Here $\mathbf{x}$ is an $(n \times 1)$ state vector, $\mathbf{u}$ is an $(m \times 1)$ control vector, τ is a constant delay factor and $\mathbf{g}$ is a given function of time that gives the initial profile for the state. Let us assume that each element in the control vector is bounded by

$$\alpha_j \leq u_j \leq \beta_j, \quad j = 1, 2, \cdots, m. \tag{8.4}$$

Associated with the system is a performance index to be minimized:

$$I = \Phi(\mathbf{x}(t_f)) \tag{8.5}$$

where Φ is some function of the final state and the final time t_f is specified. The optimal control problem is to find the control $\mathbf{u}(t)$ in the time interval $0 \leq t < t_f$ such that the performance index is minimized.

Let us divide the given time interval into P subintervals $(0, t_1), (t_1, t_2), \cdots, (t_{P-1}, t_P)$, each of length L, so that

$$L = \frac{t_f}{P}. \tag{8.6}$$

Instead of seeking only piecewise constant control policy, we will use IDP to also find a piecewise linear control policy that minimizes the performance index. Conceptually there are no difficulties and the algorithm used earlier can be used. For integration of the differential equations with time delay we use the integration routine given by Hairer *et al.* [4]. To get an appreciation of IDP for the time delay problems, we consider several examples.

8.3 Examples

8.3.1 Example 1

Let us consider the linear time-delay system used for optimal control studies by Chan and Perkins [2], Oh and Luus [8], Dadebo and Luus [3], and Luus [6]. The system is described by the differential equations

$$\frac{dx_1}{dt} = x_2(t) \tag{8.7}$$

Table 8.1: Performance index for piecewise constant control policy using different values of delay factor and number of stages

Time delay	Number of time steps			Control vector
τ	$P = 5$	$P = 10$	$P = 20$	iteration
0.10	2.5764	2.5709	2.5667	2.562
0.25	2.5374	2.5335	2.5283	2.526
0.50	2.6231	2.6189	2.6101	2.609
0.75	2.7918	2.7818	2.7704	2.769
1.00	2.9683	2.9471	2.9343	2.932

$$\frac{dx_2}{dt} = -10x_1(t) - 5x_2(t) - 2x_1(t-\tau) - x_2(t-\tau) + u(t) \tag{8.8}$$

$$\frac{dx_3}{dt} = 0.5(10x_1^2 + x_2^2 + u^2) \tag{8.9}$$

with the initial state profile

$$x_1(t) = x_2(t) = 1.0, \quad -\tau \leq t \leq 0, \quad x_3(0) = 0. \tag{8.10}$$

The performance index to be minimized is

$$I = x_3(t_f) \tag{8.11}$$

where the final time is set at $t_f = 5$.

To solve this problem by IDP, we chose a single grid point $N = 1$, with $R = 3$ randomly chosen values. The initial control policy was chosen to be zero, and the initial region size was chosen to be 0.2. Two passes, each consisting of 10 iterations were used, with the region reduction factor $\gamma = 0.80$ and the region restoration factor $\eta = 0.70$. For integration of the equations, the integration program of Hairer *et al.* [4] was used. Runs were made for different values of the delay factor τ and the number of stages P chosen were 5, 10, and 20 for both piecewise constant and piecewise linear control.

Table 8.1 shows the results obtained with the use of piecewise constant control. It is noted that even with $P = 20$, the results are quite close to the results obtained by control vector iteration procedure based on Pontryagin's maximum principle by Oh and Luus [8]. The use of $P = 20$ already gives very close results to those reported by control vector iteration.

The results for piecewise linear control are given in Table 8.2. It is obvious that the use of $P = 5$ gives an inadequate approximation to the optimal control. However, with only 10 stages excellent results are obtained. It must be realized that the control vector iteration procedure based on Pontryagin's maximum principle was slow to

Table 8.2: Performance index for piecewise linear control policy using different values of delay factor τ and number of stages

Time delay	Number of time steps			Control vector
τ	$P = 5$	$P = 10$	$P = 20$	iteration
0.10	2.5700	2.5653	2.5646	2.562
0.25	2.5332	2.5270	2.5258	2.526
0.50	2.6243	2.6076	2.6064	2.609
0.75	2.7996	2.7668	2.7658	2.769
1.00	2.9570	2.9302	2.9292	2.932

converge, especially for higher values for τ, and therefore the results obtained here for these cases are marginally better than reported by Oh and Luus [8]. Here the convergence was very rapid, as is shown in Figure 8.1 with $\tau = 1$, and $P = 10$, where after 10 iterations with $R = 3$, convergence to 2.9477 was obtained with piecewise constant control and 2.9325 was obtained with piecewise linear control. During the second pass, convergence was obtained to the values reported in the Tables.

The piecewise linear control policy with $P = 10$ with the delay factor $\tau = 1.0$, giving $I = 2.9302$, is given in Figure 8.2. A smoother control policy with $P = 20$, yielding $I = 2.9292$, is given by Luus [6]. The computation time for two passes with $P = 10$ was 0.7 s, and with $P = 20$ was 1.5 s on a PentiumII/350. Therefore, the computational effort here is negligible.

8.3.2 Example 2

Let us consider the linear time-delay system considered by Palanisamy *et al.* [9] for the optimal control of linear time-varying delay systems via single term Walsh series, which was also used by Dadebo and Luus [3] to test IDP. The system is described by the delay differential equations

$$\frac{dx_1(t)}{dt} = tx_1(t) + x_1(t-\tau) + u(t) \tag{8.12}$$

$$\frac{dx_2(t)}{dt} = x_1^2(t) + u^2(t) \tag{8.13}$$

with the initial state profile

$$x_1(t) = 1.0, \quad -1 \le t \le 0, \quad x_2(0) = 0. \tag{8.14}$$

The optimal control problem is to find the control u in the time interval $0 \le t < 2$, so that the following performance index is minimized:

$$I = x_2(t_f), \quad t_f = 2. \tag{8.15}$$

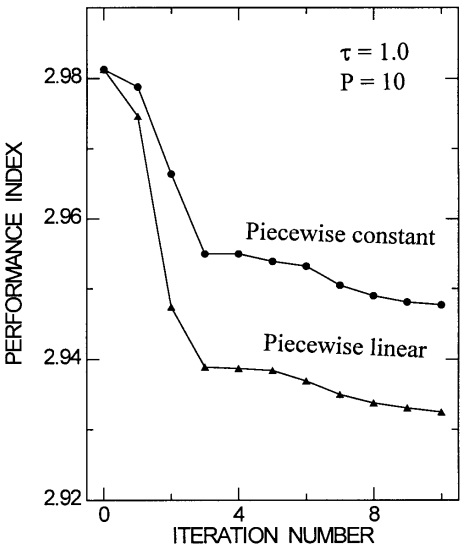

Figure 8.1: Convergence profile for Example 1 with the use of $N = 1$ and $R = 3$

The delay term $\tau = 1$.

To solve this problem by IDP, we chose again a single grid point $N = 1$, with $R = 3$ randomly chosen values, and chose $P = 10$ time stages of equal length. The initial control policy was chosen to be zero, and the initial region size was chosen to be 1.0. To establish piecewise linear control, ten passes, each consisting of 10 iterations, were used, with the region reduction factor $\gamma = 0.80$ and the region restoration factor $\eta = 0.70$. From the initial value of 80.3436 the performance index I was reduced very rapidly to 4.7968 in a computation time of 1.2 s on a PentiumII/350.

The optimal control policy is given in Figure 8.3. It is seen that the control policy is very smooth indeed and there is no need to take a larger number of stages to improve the accuracy.

8.3.3 Example 3 – Nonlinear two-stage CSTR system

The third example consists of a two-stage nonlinear CSTR system in series with a first-order irreversible chemical reaction occurring in each reactor. The system is based on the continuous stirred tank reactor originally modelled by Aris and Amundson [1] and used for Lapidus and Luus [5] for stability studies. The system is described by the differential-difference equations

$$\frac{dx_1}{dt} = 0.5 - x_1(t) - R_1 = f_1(t) \tag{8.16}$$

$$\frac{dx_2(t)}{dt} = -2(x_2(t) + 0.25) - u_1(t)[x_2(t) + 0.25] + R_1 = f_2(t) \tag{8.17}$$

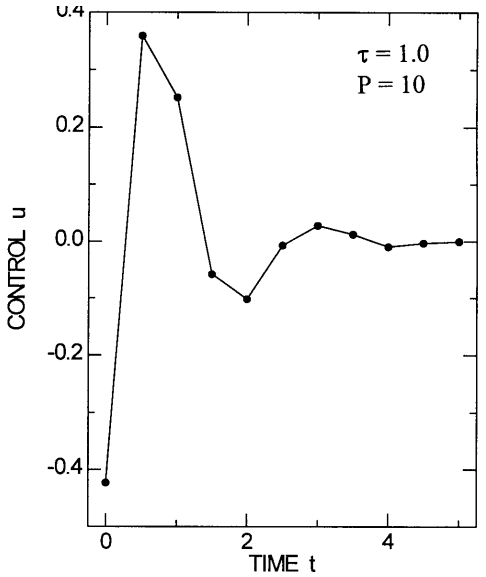

Figure 8.2: Piecewise linear continuous optimal control policy for Example 1 with $\tau = 1$

$$\frac{dx_3(t)}{dt} = x_1(t-\tau) - x_3(t) - R_2 + 0.25 \tag{8.18}$$

$$\frac{dx_4(t)}{dt} = x_2(t-\tau) - 2x_4(t) - u_2(t)[x_4(t) + 0.25] + R_2 - 0.25 \tag{8.19}$$

$$\frac{dx_5}{dt} = x_1^2 + x_2^2 + x_3^2 + x_4^2 + 0.1(u_1^2 + u_2^2). \tag{8.20}$$

We have added the last equation to give us the performance index to be minimized in the standard form, namely,

$$I = x_5(t_f) \tag{8.21}$$

where the dimensionless final time $t_f = 2$. The state variables x_1 and x_3 are normalized concentration variables in tanks 1 and 2, respectively, and x_2 and x_4 are normalized temperature variables in tanks 1 and 2, respectively. The reaction terms in tanks 1 and 2 are given by

$$R_1 = [x_1 + 0.5]\exp(\frac{25x_2}{x_2 + 2}) \tag{8.22}$$

$$R_2 = [x_3 + 0.25]\exp(\frac{25x_4}{x_4 + 2}). \tag{8.23}$$

The normalized control are bounded by

$$-1 \leq u_j \leq 1, \quad j = 1, 2. \tag{8.24}$$

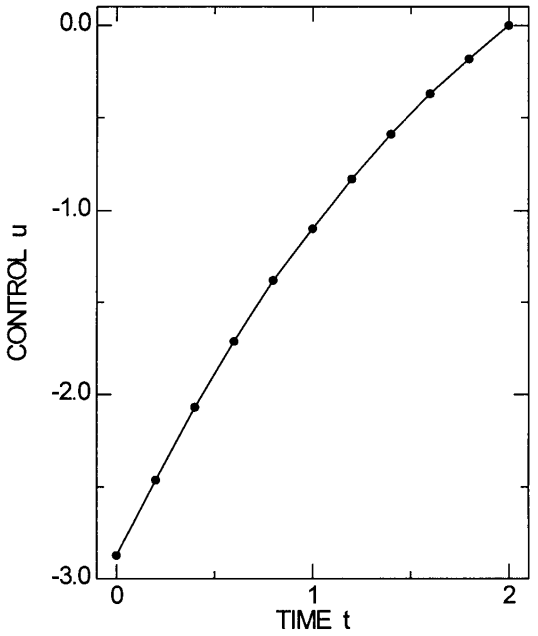

Figure 8.3: Piecewise linear continuous control policy for Example 2

The initial state profiles are given by

$$x_1(t) = 0.15, \quad -\tau \leq t \leq 0 \tag{8.25}$$

$$x_2(t) = -0.03, \quad -\tau \leq t \leq 0 \tag{8.26}$$

$$x_3(0) = 0.10 \tag{8.27}$$

$$x_4(0) = 0. \tag{8.28}$$

The optimal control problem is to find the controls $u_1(t)$ and $u_2(t)$ in the time interval $0 \leq t < 2$ so that the performance index given in Eq. (8.21) is minimized for different values of the delay factor τ. Rather than piecewise constant control as was done by Dadebo and Luus [3], here we seek piecewise linear control. In order to get convergence with $N = 1$, as is required for piecewise linear continuous control, we need a good initial estimate of the control policy. We can obtain that by approximating the delay-time system by a nondelay system where we still retain the dynamic characteristics. We expand the delay terms by Taylor series expansion retaining only the linear terms:

$$x_1(t-\tau) = x_1(t) - \tau \frac{dx_1}{dt} + \cdots \approx x_1(t) - \tau f_1(t) \tag{8.29}$$

and

$$x_2(t-\tau) = x_2(t) - \tau \frac{dx_2}{dt} + \cdots \approx x_2(t) - \tau f_2(t). \tag{8.30}$$

By substituting Eqs. (8.29)-(8.30) into Eqs. (8.18)-(8.19), we get the approximate non-delay system

$$\frac{dx_1}{dt} = f_1 \tag{8.31}$$

$$\frac{dx_2}{dt} = f_2 \tag{8.32}$$

$$\frac{dx_3}{dt} = x_1 - x_3 - \tau f_1 - R_2 + 0.25 \tag{8.33}$$

$$\frac{dx_4}{dt} = x_2 - 2x_4 - u_2[x_4 + 0.25] - \tau f_2 + R_2 - 0.25 \tag{8.34}$$

$$\frac{dx_5}{dt} = x_1^2 + x_2^2 + x_3^2 + x_4^2 + 0.1(u_1^2 + u_2^2). \tag{8.35}$$

We considered this non-delay system with $\tau = 0.1$ in Chapter 6, and found that more than one grid point was required for obtaining the optimal control. With the use of 11 grid points, the optimal 30-stage piecewise constant control policy was obtained without any difficulty. From this 30-stage piecewise constant control policy, we can get an approximation for the 10-stage piecewise linear control policy that may be used as a starting policy for the delay-time system.

By taking the delay term $\tau = 0.1$, with $\gamma = 0.95$, $\eta = 0.85$, and initial region size of 0.05, and allowing 50 passes, each consisting of 50 iterations, we found that the effect of the number of randomly chosen points R had very little effect if R was at least 25. Convergence from the initial value of $I = 0.085504$ to the vicinity of 0.023 was very rapid. The use of $R = 15$, however, gave inadequate convergence. The results are given in Table 8.3, where it is quite clear that at least 25 randomly chosen points are required. In none of these runs was the optimum obtained within 5 figures; so, a second run was necessary. For subsequent runs the initial region was chosen as 0.0001 and 30 passes were performed, where the best control policy was used at the start. After 3 such runs the optimum $I = 0.023166$ for $\tau = 0.1$ was obtained. Now using the optimal control policy for $\tau = 0.1$ as the initial control policy, the optimal control policy for $\tau = 0.2$ was obtained. Such a continuation approach was used to establish the optimal control policies for the higher values of the time delay factor, giving the results in Table 8.4.

As can be seen in Figure 8.4, the control policies do not change too much when the delay factor is increased from 0.1 to 0.8.

Luus *et al.* [7] showed that almost the same values for the performance index can be obtained with the use of 7 rather than 10 stages for piecewise continuous control. For larger values of τ, the values for the performance index are somewhat better than were reported by Oh and Luus [8] using control vector iteration procedure. For example, for $\tau = 0.8$, by IDP we obtained $I = 0.02550$ versus the value $I = 0.02561$ reported by Oh and Luus [8]. To carry out 30 passes, each consisting of 50 iterations, with $R = 25$ took about 4 min of computation time on the PentiumII/350 personal computer.

Table 8.3: Effect of the number of allowable values for control R on convergence with $\tau = 0.1$

Number of random points R	Performance index I
15	0.023241
25	0.023170
50	0.023171
100	0.023171
200	0.023170

Table 8.4: Minimum performance index for piecewise linear control with $P = 10$ for the two-stage CSTR system

Time delay τ	Performance index I
0.1	0.023166
0.2	0.023731
0.3	0.024213
0.4	0.024615
0.5	0.024940
0.6	0.025192
0.7	0.025376
0.8	0.025499

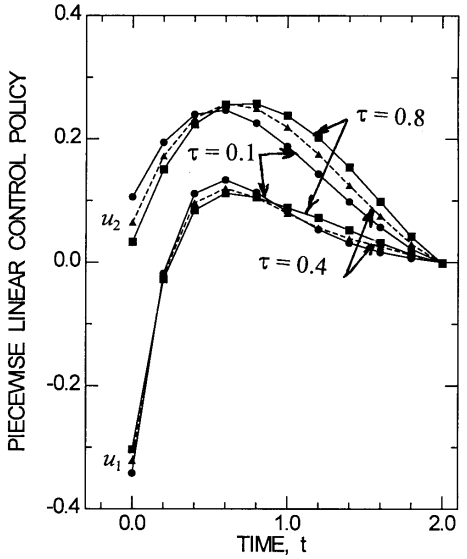

Figure 8.4: Piecewise linear continuous control policy for the two-stage CSTR system, showing the effect of the delay τ

8.4 References

[1] Aris, R. and Amundson, N.R.: "An analysis of chemical reactor stability and control", *Chem. Eng. Sci.* **7** (1958), 121-147.

[2] Chan, H.C. and Perkins, W.R.: "Optimization of time delay systems using parameter imbedding", *Automatica* **9** (1973), 257-261.

[3] Dadebo, S. and Luus, R.: "Optimal control of time-delay systems by dynamic programming", *Optimal Contr. Applic. Methods* **13** (1992), 29-41.

[4] Hairer, E., Norsett, S.P., and Wanner, G.: *Solving ordinary differential equations 1*, Springer Series in Computational Mathematics, Vol. 8; Springer, Berlin (1987).

[5] Lapidus, L. and Luus, R.: *Optimal Control of Engineering Processes*, Blaisdell, Waltham, Mass. (1967).

[6] Luus, R.: "Iterative dynamic programming: From curiosity to a practical optimization procedure", *Control and Intelligent Systems* **26** (1998), 1-8.

[7] Luus, R., Zhang, X., Hartig, F., and Keil, F.J. : "Use of piecewise linear control continuous optimal control for time-delay systems", *Ind. Eng. Chem. Res.* **34** (1995), 4136-4139.

[8] Oh, S.H. and Luus, R.: "Optimal feedback control of time-delay systems", *AIChE J.* **22** (1976), 140-147.

[9] Palanisamy, K.R., Balachandran, K., and Ramasamy, R.S.: "Optimal control of linear time-varying delay systems via single-term Walsh series", *Proc. IEE* **135** pt. D (1988), 332.

Chapter 9

Variable stage lengths

9.1 Introduction

We saw in Chapter 7 that in using piecewise constant control, increasing the number of time stages always increases the accuracy with which the control policy is approximated for a linear system with a quadratic performance index. However, there are nonlinear optimal control problems where an increase in the number of stages does not necessarily improve the performance index. For example, when the optimal control policy involves a sudden change from one extreme value to the other, the time of this switching is very important. In time optimal control problems one anticipates this type of "bang-bang" control. There are other occasions when one would least expect such a change in the optimal control policy, and it may come as a surprise. To get a good approximation to the optimal control policy then is determined not so much by the number of time stages one uses, but how well the switching time can be approximated. Let us use a simple example for illustration.

Let us consider the optimal control problem of Ko and Stevens [6] involving a first-order, reversible, exothermic chemical reaction $A \Longleftrightarrow B$ in a chemical reactor. The material and energy balances of the reactor are given by:

$$\frac{dx_1}{dt} = (1 - x_1)k_1 - x_1 k_2 \tag{9.1}$$

$$\frac{dx_2}{dt} = 300[(1 - x_1)k_1 - x_1 k_2] - u(x_2 - 290) \tag{9.2}$$

with the initial condition

$$\mathbf{x}^T(0) = [\ 0\ \ 410\ ]. \tag{9.3}$$

The state variable x_1 is the concentration of the desired component B, and x_2 denotes the absolute temperature. The control u is a normalized heat transfer term dependent on the coolant flow rate, and is bounded by

$$0 \leq u \leq 0.5. \tag{9.4}$$

The reaction rate constants are dependent on the temperature in the following manner:

$$k_1 = 1.7536 \times 10^5 \exp\left(\frac{-1.1374 \times 10^4}{1.9872 x_2}\right) \tag{9.5}$$

and

$$k_2 = 2.4885 \times 10^{10} \exp\left(\frac{-2.2748 \times 10^4}{1.9872 x_2}\right). \tag{9.6}$$

The problem is to find u in the time interval $0 \leq t < t_f$ such that the performance index

$$I = x_1(t_f), \quad t_f = 5 \tag{9.7}$$

is maximized. This problem corresponds to Case II of Reddy and Husain [15], who examined the difficulties in solving this optimal control problem using methods based on the gradient. Case I has been considered in some detail by Luus [9].

To obtain a piecewise constant optimal control policy by IDP presents no difficulties. However, what causes difficulties is choosing the number of time stages to solve this problem, since the improvement in the performance index is not very regular when the number of time stages is increased, as is shown in Figure 9.1. However, there is a definite pattern, as the three peaks occur at $P = 17$, $P = 26$, and $P = 35$.

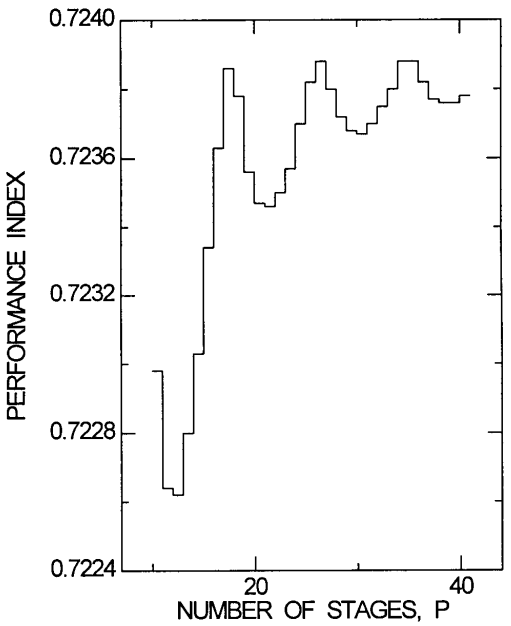

Figure 9.1: Effect of the number of time stages P on the maximum value of the performance index

It is readily observed that the use of 17 time stages gives a better value for the performance index than the use of 40 time stages. To understand the underlying reason, let us examine the optimal control profiles obtained for $P = 16, 17$, and 18. In Figure 9.2 with $P = 16$ it is seen that after two time intervals with $t = 0.625$, we are a bit late for switching to the maximum allowed value. Therefore, there is a small value for control during the second time step.

In Figure 9.3 with $P = 17$, after the second time step with $t = 0.588$, the switching is approximately right and therefore in the third time step we have the control at the maximum allowed value. This enables the performance index $I = 0.72387$ to be obtained.

In Figure 9.4 with $P = 18$, after two time steps with $t = 0.555$ the switching is too early, and the switch is made to a value less than the maximum, and the performance index of $I = 0.72378$ is not quite as good as the value obtained with $P = 17$.

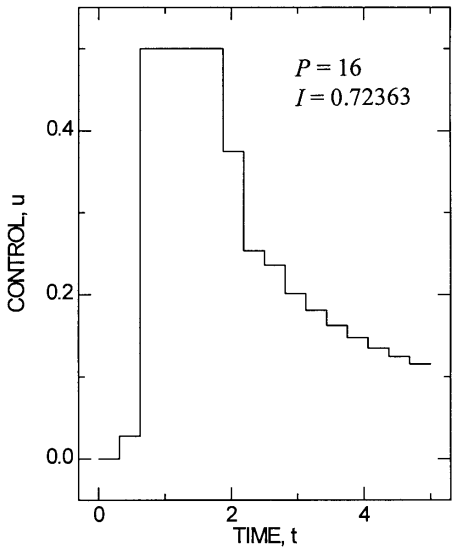

Figure 9.2: Optimal control policy obtained with $P = 16$ time stages

Therefore, we see a need for using flexible stage lengths, rather than stages of constant length. The goal in this chapter is to show how flexible stage lengths can be used in IDP and the improvements that can be obtained. Just to finish off this example, let us look at the type of control policies that are obtained with flexible stage lengths. By using 7 stages of varying length, we obtained the 6-stage optimal control policy in Figure 9.5. Here both stages 2 and 3 gave the same upper bound for the control. The performance index $I = 0.723886$ is marginally better than the value obtained with 17 stages of equal length.

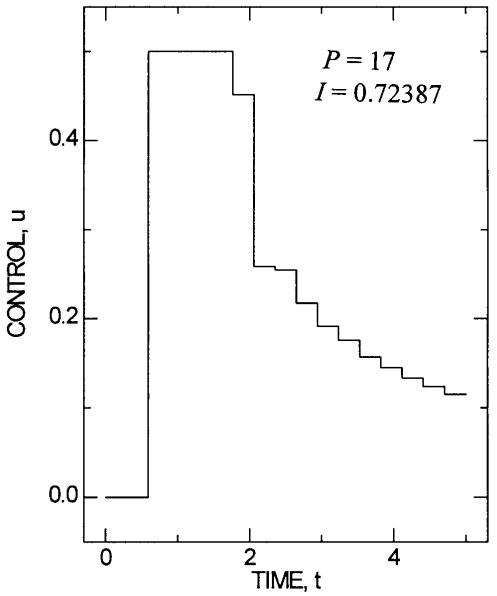

Figure 9.3: Optimal control policy obtained with $P = 17$ time stages

Figures 9.6 and 9.7 show that by increasing the number of stages, the performance index can be improved, but very slightly. The portion of the control policy after the maximum bound is not very important. This very low sensitivity of this part of the control policy makes accurate determination of the optimal control difficult. The most important part of the policy is the switching time, and it is essential to obtain the switching time from 0 to 0.5 at time $t = 0.5796$ as accurately as possible.

We can therefore see benefits in using stages of varying lengths for establishing the optimal control policy. In this chapter we will outline the algorithm to incorporate the optimization of the stage lengths into IDP and also consider the problem where the final time is specified as in the example that we just considered.

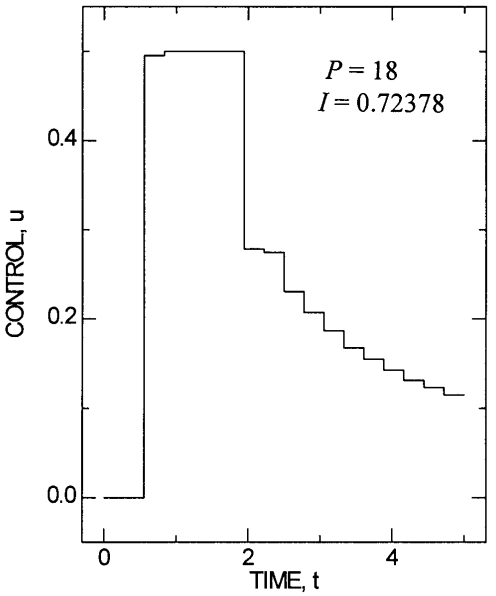

Figure 9.4: Optimal control policy obtained with $P = 18$ time stages

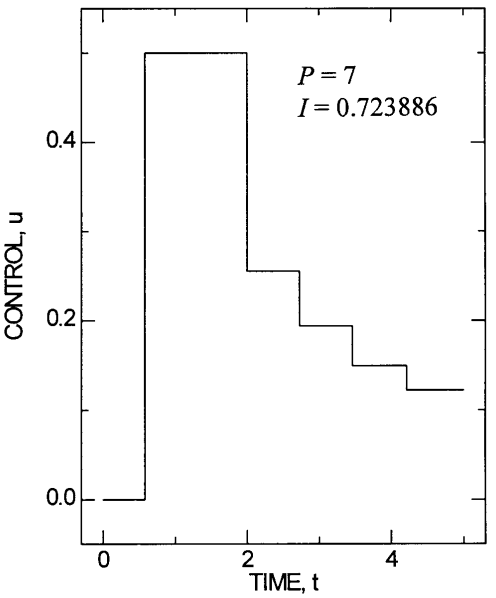

Figure 9.5: Optimal control policy obtained with $P = 7$ time stages of varying length

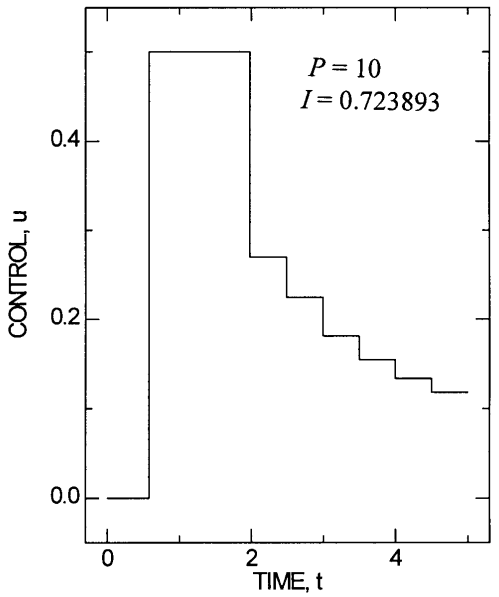

Figure 9.6: Optimal control policy obtained with $P = 10$ time stages of varying length

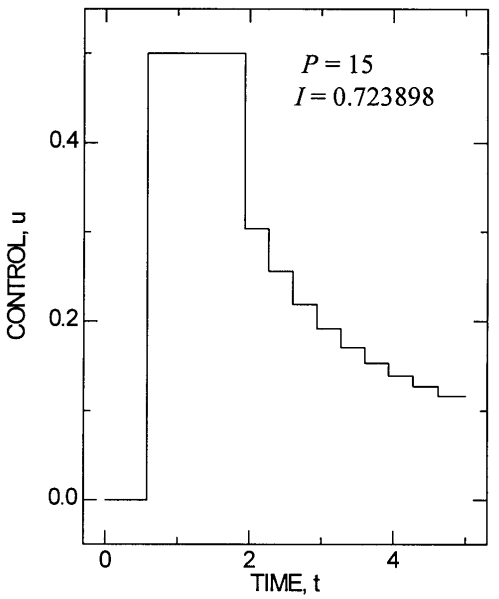

Figure 9.7: Optimal control policy obtained with $P = 15$ time stages of varying length

9.2 Variable stage-lengths when final time is free

The development of the algorithm for IDP to allow stages to be of varying length was done by Bojkov and Luus [2]. We consider the stage lengths to constitute another control variable and the formulation is quite straightforward. Let us consider the optimal control problem where the state equation is

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{9.8}$$

with the initial state $\mathbf{x}(0)$ given. The $(m \times 1)$ control vector is bounded by

$$\alpha_j \leq u_j(t) \leq \beta_j, \quad j = 1, 2, \cdots, m. \tag{9.9}$$

The performance index to be maximized is a scalar function of the state at the final time t_f

$$I = \Phi(\mathbf{x}(t_f)) \tag{9.10}$$

where the final time t_f is not specified.

Let us divide the time interval into P time stages, each of variable length

$$v(k) = t_k - t_{k-1}, \quad k = 1, 2, \cdots, P \tag{9.11}$$

where

$$v(k) \geq 0, \quad k = 1, 2, \cdots, P. \tag{9.12}$$

The control is kept constant in each of these time intervals.

Let us now introduce a normalized time variable τ, by defining $d\tau$ in the time interval $t_{k-1} \leq t < t_k$ through the relationship

$$dt = v(k) P d\tau \tag{9.13}$$

so that

$$t_k - t_{k-1} = v(k) P (\tau_k - \tau_{k-1}). \tag{9.14}$$

Therefore

$$\tau_k - \tau_{k-1} = \frac{1}{P} \tag{9.15}$$

and in the normalized time the stages are of equal length and the final time is $\tau = 1$, so that the algorithm for IDP presented earlier can be used directly. The only change to be made is that the differential equation now becomes

$$\frac{d\mathbf{x}}{d\tau} = v(k) P \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{9.16}$$

in the kth time interval.

9.2.1 IDP algorithm

For this free final-time problem we can consider the stage length as an additional control variable and augment the control vector to an $((m+1)\times 1)$ vector, but for clarity of presentation we keep it separate. It is clear that, as before, for greatest efficiency, IDP should be used in a multi-pass fashion, where after a pass the region is restored to a fraction η of its value at the beginning of the previous pass. The algorithm can be presented in eight steps:

1. Choose the number of time stages P, the number of grid points N, the number of allowable values for control (including stage lengths) R at each grid point, the region contraction factor γ, the region restoration factor η, initial values for the control and the stage lengths, the initial region sizes, the number of iterations to be used in every pass, and the number of passes.

2. By choosing N values for control and the stage length, evenly distributed inside the specified regions, integrate Eq. (9.16) from $\tau = 0$ to $\tau = 1$ to generate N trajectories. The N values for $\mathbf{x}$ at the beginning of each time stage constitute the N grid points at each stage.

3. Starting at stage P, corresponding to the normalized time $\tau = (P-1)/P$, for each grid point generate R sets of values for control and stage length:

$$\mathbf{u}(P) = \mathbf{u}^{*j}(P) + \mathbf{D}\mathbf{r}^{j}(P) \tag{9.17}$$

$$v(P) = v^{*j}(P) + \omega s^{j}(P) \tag{9.18}$$

where $\mathbf{D}$ is an $m \times m$ diagonal matrix with different random numbers between -1 and 1 along the diagonal and ω is another random number between -1 and 1; $\mathbf{u}^{*j}(P)$ is the best value for control, and $v^{*j}(P)$ is the best value for the stage length, both obtained for that particular grid point in the previous iteration. Integrate Eq. (9.16) from $(P-1)/P$ to $\tau = 1$ once with each of the R allowable values for control and stage length to yield $\mathbf{x}(t_f)$ so that the performance index can be evaluated. Compare the R values of the performance index and choose the control and stage length which give the maximum value. The corresponding control and stage length are stored for use in step 4.

4. Step back to stage $P-1$, corresponding to time $\tau = (P-2)/P$. For each grid point generate R allowable sets of control and stage lengths. Integrate Eq. (9.16) from $(P-2)/P$ to $(P-1)/P$ once with each of the R sets. To continue integration, choose the control and the stage length from step 3 that corresponds to the grid point that is closest to the $\mathbf{x}$ at $\tau = (P-1)/P$. Now compare the R values of the performance index and store the control policy and the stage length that yield the maximum value.

5. Step back to stage $P-2$, and continue the procedure in the previous step. Continue until stage 1 corresponding to $\tau = 0$ with the given initial state as the grid point is reached. Make the comparison of the R values of the performance index to give the best control and the stage length for this stage. We now have the best control

policy and the stage length for each stage in the sense of maximizing the performance index from the allowable choices.

6. In preparation for the next iteration, reduce the size of the allowable regions

$$\mathbf{r}^{j+1}(k) = \gamma \mathbf{r}^{j}(k), \quad k = 1, 2, \cdots, P \tag{9.19}$$

$$s^{j+1}(k) = \gamma s^{j}(k), \quad k = 1, 2, \cdots, P \tag{9.20}$$

where γ is the region reduction factor and j is the iteration index. Use the best control policy and the stage lengths from step 5 as the midpoint for the next iteration.

7. Increment the iteration index j by 1 and go to step 2 to generate another set of grid points. Continue for the specified number of iterations.

8. Increment the pass number index by 1, set the iteration index j to 1 and go to step 2. Continue for the specified number of passes, and examine the results.

A reasonable number of iterations per pass is 20 or 30 with the region contraction factor $\gamma = 0.95$, since the region is then well covered. For many problems the optimum is not found within reasonable accuracy in a single pass. Therefore the region should be restored to some reasonable value before the iterations are carried out again. An easy way to restore the region size is to take as its value a fraction η of its value at the beginning of the previous pass. To help in deciding what value of η to use, we can refer to Table 9.1 where it is shown how the initial region is reduced by the choice of η if 20 or 30 passes are made. If 30 passes are used and η is chosen as 0.95, then the initial region size after 30 passes becomes 0.2146 times its value at the beginning of the first pass; if η is chosen as 0.85, then the initial region size after 30 passes becomes 0.0076 times its value at the beginning of the first pass. Therefore, a balance is sought between premature collapse of the region size and the speed of convergence.

Although this method of restoring the region size after every pass is not the best method, since it depends very much on the initial choice of the region size, we will use this method for all the examples in this book. Recently Luus [11] showed that the use of the extent over which a variable changes during a pass as the value to use as the region size for the next pass in steady state optimization yielded significant improvement in the convergence rate, and that approach enabled difficult fed-batch reactor problems to be optimized with great accuracy [13].

9.3 Problems where final time is not specified

The best way to gain insight about the choice of parameters in IDP is to examine some optimal control problems. Here we look at problems where the final time t_f is not specified.

Table 9.1: Effect of the region restoration factor η on the region size after 20 and 30 passes

Region restoration factor	Number of passes	
η	20	30
0.95	0.3584	0.2146
0.94	0.2901	0.1563
0.93	0.2342	0.1134
0.92	0.1887	0.0820
0.91	0.1516	0.0591
0.90	0.1216	0.0424
0.89	0.0972	0.0303
0.88	0.0776	0.0216
0.87	0.0617	0.0153
0.86	0.0490	0.0108
0.85	0.0388	0.0076

9.3.1 Oil shale pyrolysis problem

Difficulties in the determination of the optimal control policy, by methods based on Pontryagin's maximum principle, for oil shale pyrolysis were observed by Wen and Yen [17], and by Luus [7]. This problem was considered by Rosen and Luus [16] with some modifications to SQP, by Luus [9] using IDP, and by Bojkov and Luus [2] using IDP with flexible stage lengths. There are a series of five chemical reactions:

$$\begin{aligned} A_1 &\longrightarrow A_2 \\ A_2 &\longrightarrow A_3 \\ A_1 + A_2 &\longrightarrow A_2 + A_2 \\ A_1 + A_2 &\longrightarrow A_3 + A_2 \\ A_1 + A_2 &\longrightarrow A_4 + A_2 \end{aligned}$$

The system is described by the differential equations

$$\frac{dx_1}{dt} = -k_1 x_1 - (k_3 + k_4 + k_5) x_1 x_2 \tag{9.21}$$

$$\frac{dx_2}{dt} = k_1 x_1 - k_2 x_2 + k_3 x_1 x_2 \tag{9.22}$$

$$\frac{dx_3}{dt} = k_2 x_2 + k_4 x_1 x_2 \tag{9.23}$$

$$\frac{dx_4}{dt} = k_5 x_1 x_2 \tag{9.24}$$

Table 9.2: Kinetic parameters for the pyrolysis problem

Reaction	Kinetic parameters	
i	a_i	b_i
1	8.86	20300/1.9872
2	24.25	37400/1.9872
3	23.67	33800/1.9872
4	18.75	28200/1.9872
5	20.70	31000/1.9872

with the initial condition

$$\mathbf{x}(0) = [\ 1\ \ 0\ \ 0\ \ 0\ \ 0\ ]^T. \tag{9.25}$$

The rate constants are given by

$$k_i = \exp(a_i - \frac{b_i}{T}), \quad i = 1, 2, \cdots, 5, \tag{9.26}$$

where the kinetic parameters a_i and b_i are given in Table 9.2. The optimal control problem is to find both the residence time t_f and the temperature T in the time interval $0 \leq t < t_f$, so that the concentration of pyrolytic bitumen given by x_2 is maximized. The performance index to be maximized is thus

$$I = x_2(t_f). \tag{9.27}$$

The constraints on the temperature are

$$698.15 \leq T \leq 748.15. \tag{9.28}$$

By using control vector iteration based on Pontryagin's maximum principle, Luus [7] solved this optimal control problem by fixing t_f at 17 different values and then interpreting the results to conclude that the maximum value of the performance index of 0.35375 occurs at $t_f = 9.5$ min. Rosen and Luus [16] improved the value to $I = 0.35379$ at time $t_f = 9.123$ min. By using IDP with $P = 80$ time steps of equal length, Luus [9] obtained $I = 0.35382$ at $t_f = 9.3$ min. By using flexible stage lengths in IDP, Bojkov and Luus [2] obtained $I = 0.35381$ with $P = 5$ time stages and $I = 0.35382$ with $P = 10$ flexible stages. Let us examine this latter approach.

Preliminary run

For a preliminary run we chose $P = 10$ stages each of length 1 min initially, $R = 25$ random points, $N = 1$ grid point, $\gamma = 0.95$, 20 iterations per pass, 20 passes with region restoration factor $\eta = 0.85$, initial temperature of 723.15 for each stage, and initial region size for the temperature of 50 and for the stage lengths 0.01.

The run took 42.3 s on a PentiumII/350, and resulted in the temperature profile shown in Figure 9.8. There are 4 distinct stages. Of the original 10 stages, the first four collapsed to the lower limit with a total length of 3.595 min; the next stage of length 1.369 min yielded the upper limit; the sixth stage of length 0.901 min gave a temperature of 699.50; and the last four stages collapsed to a single stage on the lower boundary. The value of the performance index is $I = 0.35377$ and $t_f = 9.368$ min. It is interesting to note that this four-stage control policy gives a marginally better value than the maximum yield of 0.35375 obtained by control vector iteration procedure by Luus [7].

Next run

For the next run, we chose the same parameters for IDP as before. For initial conditions, we divided the third stage into seven stages, each of length 1.369/7 min and kept the other three stages as obtained in the preliminary run. For these seven stages we took an initial temperature of 723.15. For the other three stages we used the values obtained in the preliminary run. After 20 passes we obtained a 9-stage control policy shown in Figure 9.9 with $I = 0.353820$ and $t_f = 9.283$ min. This value of the performance index is marginally better than 0.353794 obtained by Rosen and Luus [16], and just very slightly less than 0.353821 obtained by Bojkov and Luus [2] with the use of $P = 20$ flexible stages. Although Bojkov and Luus used more than 1 grid point, we found here that a single grid point worked very well.

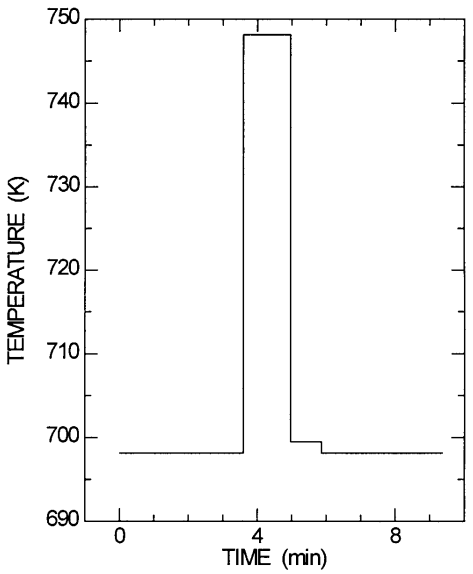

Figure 9.8: Four-stage optimal control policy for the pyrolysis problem resulting from the preliminary run, yielding $I = 0.3537724$

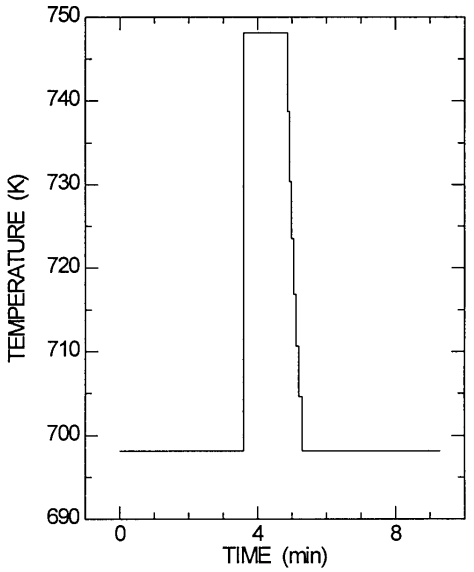

Figure 9.9: 9-stage optimal control policy for the pyrolysis problem, yielding $I = 0.3538200$

9.3.2 Modified Denbigh reaction scheme

Let us now consider the system of reactions used by Denbigh [3] and by Aris [1] with a small modification by adding the reaction where the desired product is converted into waste, as was done by Luus and Dong [12]. The reaction scheme is

$$\begin{aligned} A + B &\longrightarrow X \\ A + B &\longrightarrow P \\ X &\longrightarrow Y \\ X &\longrightarrow Q \\ Y &\longrightarrow Z \end{aligned}$$

where X is an intermediate, Y is the desired product, and P, Q, and Z are waste products. It is desired to maximize the yield of Y by choosing the temperature of the reactor as a function of time. Although there are seven chemical species, we are concerned only with the concentrations of A, X, and Y, and therefore choose as state variables their concentrations expressed as mole fractions, respectively. The equations describing the system are

$$\frac{dx_1}{dt} = -k_1 x_1 - k_2 x_1 \tag{9.29}$$

$$\frac{dx_2}{dt} = k_1 x_1 - (k_3 + k_4) x_2 \tag{9.30}$$

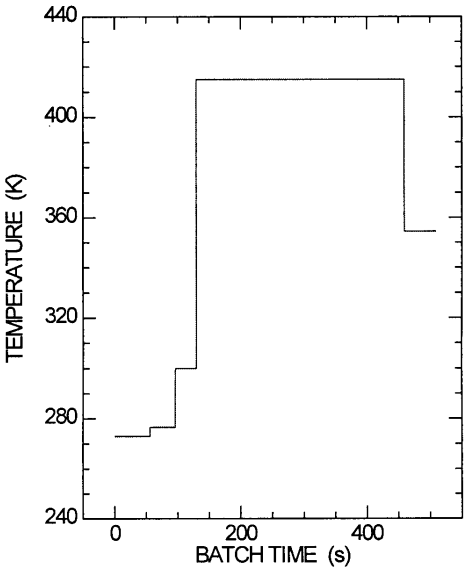

Figure 9.10: 5-stage control policy for the modified Denbigh reaction scheme resulting from the preliminary run giving a bitumen yield of 0.54530

$$\frac{dx_3}{dt} = k_3 x_2 - k_5 x_3 \tag{9.31}$$

with the initial condition

$$\mathbf{x}(0) = [\ 1\ \ 0\ \ 0\ ]^T. \tag{9.32}$$

The rate constants are given by

$$k_i = k_{i0} \exp\left(\frac{-E_i}{RT}\right) \tag{9.33}$$

where the kinetic parameters k_{i0} and E_i/R are given in Table 9.3.

The constraints on the temperature are

$$273 \leq T \leq 415. \tag{9.34}$$

Table 9.3: Kinetic parameters for the modified Denbigh reaction problem

Reaction	Kinetic parameters	
i	k_{i0}	E_i/R
1	10^3	3000
2	10^7	6000
3	10	3000
4	10^{-3}	0
5	1	3100

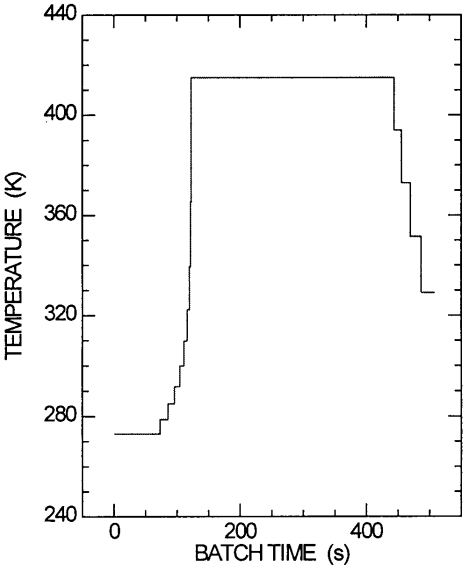

Figure 9.11: 14-stage optimal control policy for the modified Denbigh reaction scheme giving a bitumen yield of 0.54927

Luus and Dong [12] considered the case where the final time t_f was specified as 500 s. Here, we consider the case where t_f is not specified, but we do not want it to be excessively large. Therefore we introduce a penalty for the final time, and choose the performance index in the form

$$I = x_3(t_f) - \theta t_f \tag{9.35}$$

where the choice of the penalty function factor θ influences the final time obtained at the optimum. It will be interesting to see if we can find the value for θ which will give a final time of approximately $t_f = 500$ s.

Preliminary run

We took $P = 10$ time stages each of length 40 s, $N = 1$ grid point, $R = 25$ randomly chosen values for control and stage length, $\gamma = 0.95$, $\eta = 0.85$, penalty function factor $\theta = 10^{-6}$, 20 iterations per pass and allowed 20 passes. The initial temperature was chosen as 344 for each stage, the initial region size for the temperature was 142 and the initial region size for the stage length was 1.0.

Convergence to $x_3(t_f) = 0.54530$ resulted with the 5-stage control policy shown in Figure 9.10. The final time $t_f = 508.35$ s.

Next run

For a more refined control policy, the stages where the control was on the lower and upper bound were left as calculated, but the remaining 3 stages were each split into

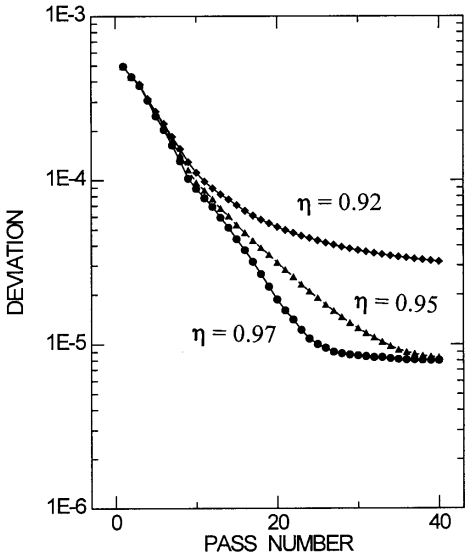

Figure 9.12: Effect of the region restoration factor η on convergence rate, showing the deviations of the yield from the optimum 0.54928 as a function of the pass number

4 stages, resulting in a 14-stage initial control policy. The initial region sizes were reduced by a factor of 10 from the values used in the preliminary run (i.e., 14.2 for the temperature and 0.1 for the stage lengths), the number of passes was doubled to 40, the number of iterations in each pass was increased to 30, and the region restoration factor η was increased to 0.95.

Convergence to $x_3(t_f) = 0.54927$ resulted in a computation time of 3843 s on a PentiumII/350 for 40 passes, giving the 14-stage control policy in Figure 9.11. This value of the yield is very close to 0.54937 obtained by Luus and Dong [12] by using 37 time stages and final time $t_f = 500$ s. By using 50 stages, Bojkov and Luus [2] obtained a yield of 0.54948 with a final time of 508.4 s. The region restoration factor here had to be chosen sufficiently large to avoid premature collapse of the search regions as is shown in Figure 9.12. From this figure it is quite obvious that the initial region sizes have been chosen too small, preventing us from using a smaller value for the restoration factor η. It should also be possible to reduce the number of randomly chosen candidates for control and stage lengths.

To see whether R could be reduced by a better initial choice for region sizes, and also to gain better accuracy by using a larger number of stages, we followed the strategy as suggested by Luus [9]. Let us take a total of $P = 30$ stages. Initially the first stage was taken of length 89 s and stage 27 was taken of length 332 s, with control values of 273 and 415 respectively. The remaining 28 stages were initially chosen of length 0.5 s with control value of 344. The initial region size for each stage length was taken as 5 and the initial region size for control was taken as 25. The

region contraction factor was taken as $\gamma = 0.95$, and the region restoration factor was chosen to be $\eta = 0.85$. A single grid point $N = 1$ was used with 40 passes, each consisting of 30 iterations. To get a final time closer to 500 s, we increased the penalty factor θ to 1.3×10^{-6}.

Convergence to the optimum $x_3(t_f) = 0.54946$ with $t_f = 499.55$ was obtained with both $R = 7$ and $R = 9$, but premature collapse occurred with $R = 5$, giving only $x_3(t_f) = 0.54945$ after 40 passes. The computation time for $R = 7$ was 2802 s on a PentiumII/350. The convergence profile is shown in Figure 9.13. The resulting optimal control policy is given in Figure 9.14, and the state trajectories are shown in Figure 9.15.

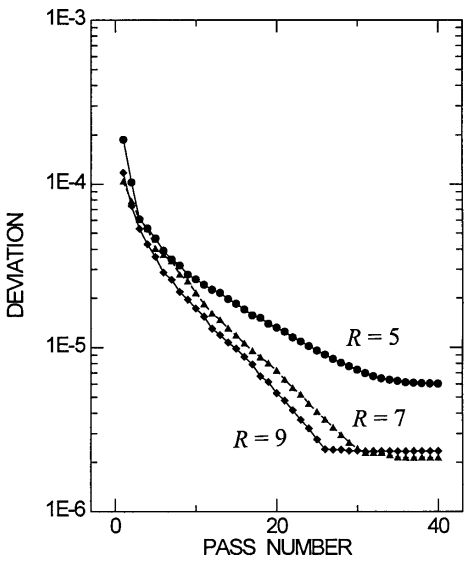

Figure 9.13: Effect of R on the convergence rate for 30-stage control policy, showing deviations from $x_3(t_f) = 0.54946$

By decreasing the penalty function factor θ, the final time can be increased, so that by trial and error the value of θ could be found to yield a final time of 500 s as was specified for this problem by Luus and Dong [12]. To find the appropriate value for θ, we performed a couple of further runs, yielding the results in Table 9.4. A better way of using flexible stage lengths when the final time is specified is given in the next section.

9.4 Systems with specified final time

The use of IDP with variable stage lengths and fixed final time was considered by Luus [10]. The problem is centered around the maximization of a performance index

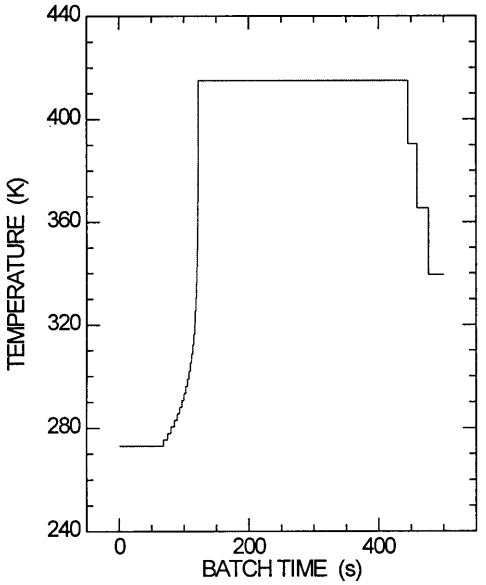

Figure 9.14: Optimal 30-stage control policy yielding $x_3(t_f) = 0.549461$

Table 9.4: Effect of the penalty function factor θ on the resulting final time t_f

Penalty function factor θ	Resulting final time t_f	Bitumen yield $x_3(t_f)$
1.00×10^{-6}	508.42	0.54946
1.28×10^{-6}	499.93	0.54946
1.29×10^{-6}	499.74	0.54946
1.30×10^{-6}	498.78	0.54946

in Eq. (9.10) subject to the constraint

$$v(1) + v(2) + \cdots + v(P) = t_f. \tag{9.36}$$

Let us define the calculated value of the final time as

$$t_{fc} = \sum_{k=1}^{P} v(k). \tag{9.37}$$

Then the restriction of Eq. (9.36) is

$$t_{fc} - t_f = 0. \tag{9.38}$$

Let us now introduce the augmented performance index

$$J = I - \theta(t_{fc} - t_f - s)^2 \tag{9.39}$$

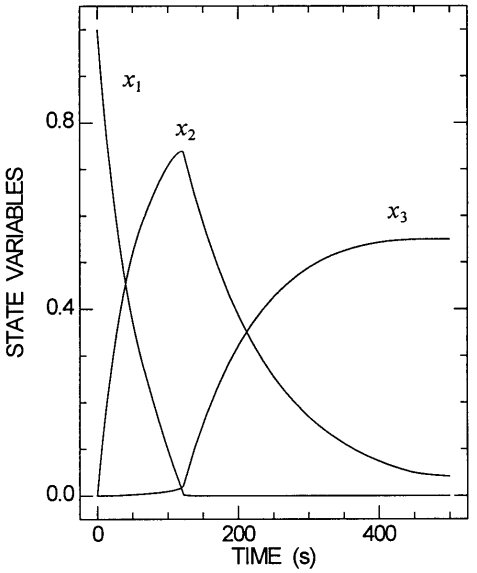

Figure 9.15: State trajectories for the modified Denbigh reaction scheme giving a bitumen yield of 0.54946

where s is a shifting term. The shifting term is updated after every pass by

$$s^{q+1} = s^q - (t_{fc} - t_f) \tag{9.40}$$

where q is the pass number. Unless sensitivity information is available, the initial value for s is taken to be zero. The effect of this shifting term s is to decrease the difference between the calculated and the specified value of the final time, so that θ can be kept reasonably small. Also the use of the shifting parameter enables the optimum to be accurately obtained.

To illustrate this procedure we first consider the modified Denbigh reaction scheme considered in Section 9.3. We will use $P = 30$ stages, where the first stage is initially chosen to be of length 89 s and stage 27 is taken to be of length 332 s, with control values of 273 and 415 respectively. The remaining 28 stages are initially chosen to be of length 0.5 s with control value of 344. We take a single grid point ($N = 1$), $R = 7$ randomly chosen allowable values for control and stage lengths, $\gamma = 0.95$, $\eta = 0.85$, with initial region size for control 25 and for stage lengths 5.0. We allow 40 passes, with 30 iterations in each pass.

When the penalty function factor $\theta \leq 10^{-5}$, then convergence to $I = 0.54946$ was obtained in each run. With $\theta = 2 \times 10^{-5}$, convergence to 0.54945 was obtained, and with $\theta = 5 \times 10^{-5}$, $I = 0.54938$ resulted. When the penalty function factor was less than 10^{-8} the final time constraint was no longer satisfied to 5 figures. With $\theta = 10^{-9}$, for example, the optimum value of the performance index of 0.54946 was obtained, but the final time was 500.02. Thus, there is a wide range for θ where the optimum is

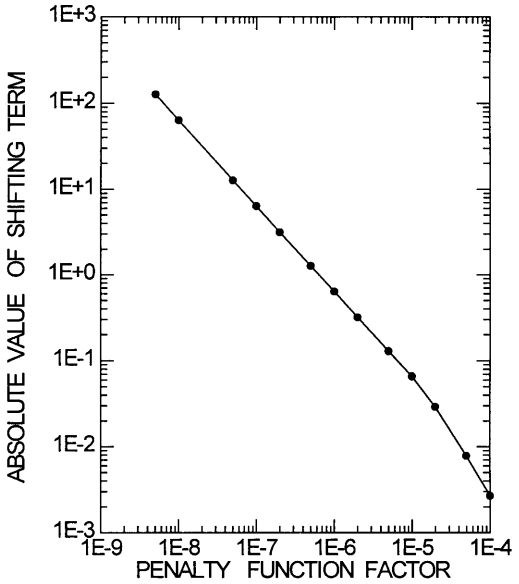

Figure 9.16: Shifting parameter s as a function of the penalty function factor θ, showing a linear relationship on the log-log plot for $\theta \leq 1.0 \times 10^{-5}$

accurately obtained and the final time constraint is satisfied. As is shown in Figure 9.16, there is a relationship between the shifting parameter s and the penalty function factor θ on a log-log plot, showing that θs is a constant. In fact, the product $-2\theta s$ gives the sensitivity of the performance index on the final time. Here s is negative, showing that the performance index is increased if the final time is increased by the relationship

$$\delta I = -2\theta s \delta t_f. \tag{9.41}$$

Here the product $-2\theta s$ is 1.28×10^{-6}, so increasing the final time by 1 s does not change the performance index in the fifth figure. It must be realized that this sensitivity information is local information only, like the sensitivity information provided by Lagrange multipliers or shadow prices in linear programming.

9.4.1 Fed-batch reactor

Let us consider the fed-batch reactor model presented by Park and Ramirez [14], and used for optimal control studies by Luus [8], Yang and Wu [18] and Gupta [4]. The difficulties in establishing the optimal control policy are partly due to the low sensitivity of the performance index on the control policy. The bioreactor is described by five differential equations

$$\frac{dx_1}{dt} = g_1(x_2 - x_1) - \frac{u}{x_5}x_1 \tag{9.42}$$

$$\frac{dx_2}{dt} = g_2 x_3 - \frac{u}{x_5} x_2 \tag{9.43}$$

$$\frac{dx_3}{dt} = g_3 x_3 - \frac{u}{x_5} x_3 \tag{9.44}$$

$$\frac{dx_4}{dt} = -7.3 g_3 x_3 + \frac{u}{x_5}(20 - x_4) \tag{9.45}$$

$$\frac{dx_5}{dt} = u \tag{9.46}$$

where

$$g_3 = \frac{21.87 x_4}{(x_4 + 0.4)(x_4 + 62.5)} \tag{9.47}$$

$$g_2 = \frac{x_4 e^{-5x_4}}{0.1 + x_4} \tag{9.48}$$

$$g_1 = \frac{4.75 g_3}{0.12 + g_3} \tag{9.49}$$

with the initial condition

$$\mathbf{x}(0) = [\ 0\ \ 0\ \ 1\ \ 5\ \ 1\ ]^T. \tag{9.50}$$

Here x_1 denotes the concentration of secreted SUC-s2 in culture and x_2 denotes the concentration of total SUC-s2 in culture, both expressed in arbitrary units; x_3 is the culture cell density (g/L), x_4 is the culture glucose level (g/L), and x_5 is the culture volume (L). The feed rate to the reactor u is bounded by

$$0 \le u \le 2. \tag{9.51}$$

The performance index to be maximized is the amount of secreted SUC2-s2 produced at the final time t_f

$$I = x_1(t_f) x_5(t_f) \tag{9.52}$$

where the final time t_f is specified as 15 h. We also analyze the effect of changing t_f.

For integration of the differential equations with DVERK [5], the local error tolerance of 10^{-6} was used. However, to ensure stability during integration, each time step was broken into a number of subsections approximately one-half of the number of stages P. The first run consisted of taking $P = 15$, and the number of subsections for integration was chosen as 7 to ensure stability. For the first run it was decided to use $R = 5$, $\gamma = 0.95$, $\eta = 0.85$, initial values for control and stage lengths of 1 and 0.5, respectively, 50 passes, each consisting of 40 iterations, penalty function factor $\theta = 100$, and initial value for the shifting parameter $s = 0$. The computer program is listed in Appendix D. In a computation time of 372 s on a PentiumII/350, convergence to $I = 32.6691$ and $s = -0.04878$ was obtained, yielding the control policy shown in Figure 9.17.

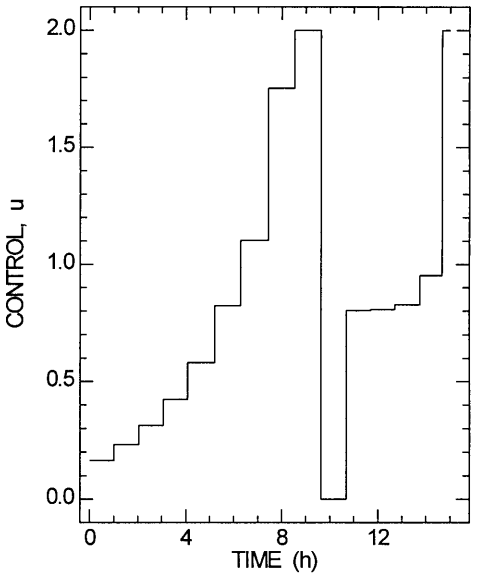

Figure 9.17: Control policy resulting from the first run with $P = 15$, giving $I = 32.6691$

To see if the result could be improved, a second run with $P = 15$ was performed where the number of passes was increased to 100, and the control policy in Figure 9.17 was used as the initial control policy. The region restoration factor was increased to $\eta = 0.88$ and the value of $s = -0.04879$, as obtained from the previous run, was used as the starting value for the shifting parameter. The performance index was improved to $I = 32.6754$ with the resulting optimal control policy in Figure 9.18. The only noticeable change is the more regular sized steps before the first peak. This value of the performance index is slightly better than the value 32.6134 obtained with 31 stages of equal length by Luus and Hennessy [13].

Table 9.5: Comparison of predicted and computed values for the performance index with the use of $P = 30$ and $\theta = 100$

Batch time t_f	Shifting parameter s	Sensitivity factor $-2\theta s$	Predicted I_p	Computed I
15.0	-0.04880	9.760	-	32.693
15.1	-0.05025	10.050	33.459	33.683
15.2	-0.05173	10.346	34.688	34.703
15.3	-0.05323	10.646	35.738	35.753
15.4	-0.05477	10.954	36.818	36.833
15.5	-0.05635	11.270	37.928	37.944

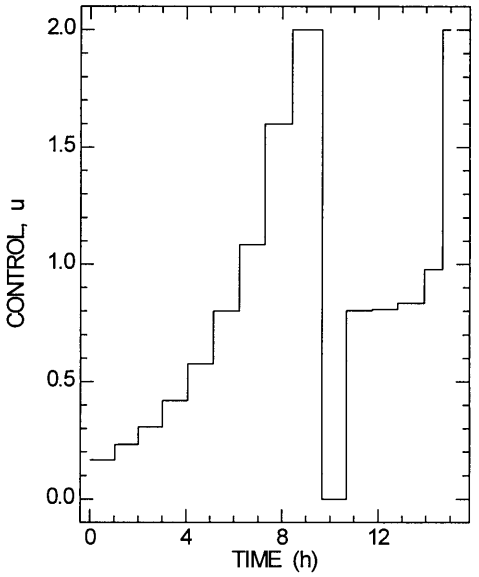

Figure 9.18: Optimal control policy for the fed-batch reactor with $P = 15$, giving $I = 32.6754$

The same approach was used with $P = 20$, where the preliminary run with $R = 7$ and $N = 1$ gave the control policy in Figure 9.19, with $I = 32.6697$. After several runs, with $R = 25$ and $N = 1$, always using the best control policy obtained from the previous run with a slight perturbation as the initial control policy, the performance index was improved to $I = 32.6939$. The resulting optimal control policy is quite smooth, as is shown in Figure 9.20, with the shifting parameter at the optimum of $s = -0.04880$.

Effect of t_f

One advantage of having the shifting parameter s is having the sensitivity information. From the product $-2\theta s$ we can predict the effect of the final time without carrying out the computations. By rewriting Eq. (9.39) in the form

$$J = I + 2\theta s(t_{fc} - t_f) - \theta(t_{fc} - t_f)^2 - \theta s^2 \tag{9.53}$$

we see that $-2\theta s$ provides the Lagrange multiplier for the equality constraint at the optimum.

The information is only local, but it provides a very good estimate. This is shown in Table 9.5, where we have provided results obtained by increasing the final time by increments of 0.1 from 9 to 9.5 h. The predicted values are obtained by adding the sensitivity times 0.1 to the previous value of the performance index obtained by computation. It is interesting to note that s increases in magnitude as the batch time

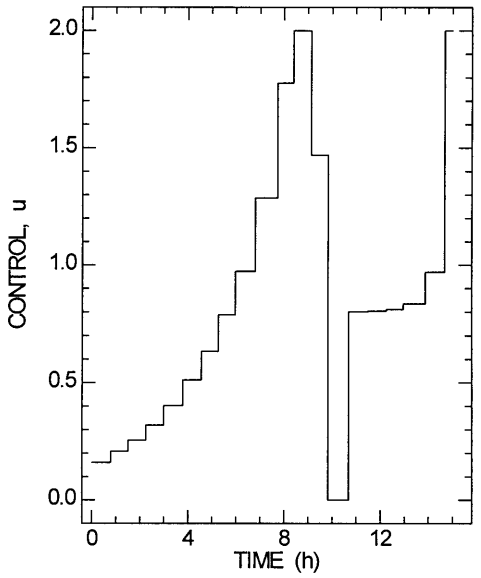

Figure 9.19: Control policy resulting from the first run with $P = 20$, giving $I = 32.6697$

t_f increases, showing that the curve of performance index versus batch time curves upward.

There is no visible difference between the computed values and the estimated values in Figure 9.21. It is also noted that the curve is convex upward, as predicted from the values of s. The optimal control policy for $t_f = 15.5$ h is shown in Figure 9.22. It is noted that the plateau of the first peak is longer than with $t_f = 15.0$ and the zero stage occurs at a slightly later time.

The real difference in the control policy is noted with $t_f = 19$ and $t_f = 20$ h, as shown in Figures 9.23 and 9.24. The very short zero-stage disappears as the batch time is increased beyond $t_f = 20$ h.

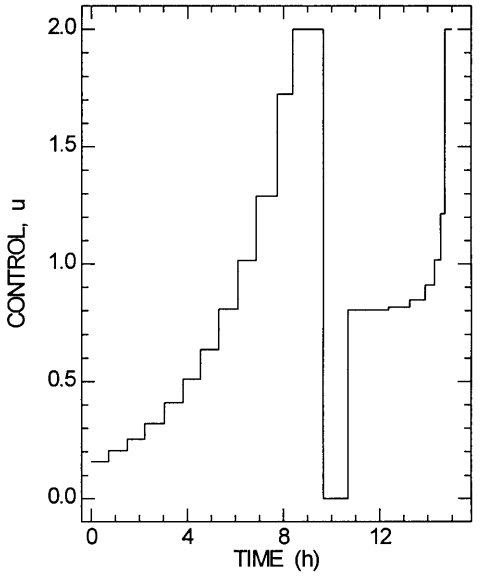

Figure 9.20: Optimal control policy for the fed-batch reactor with $P = 20$, giving $I = 32.6939$

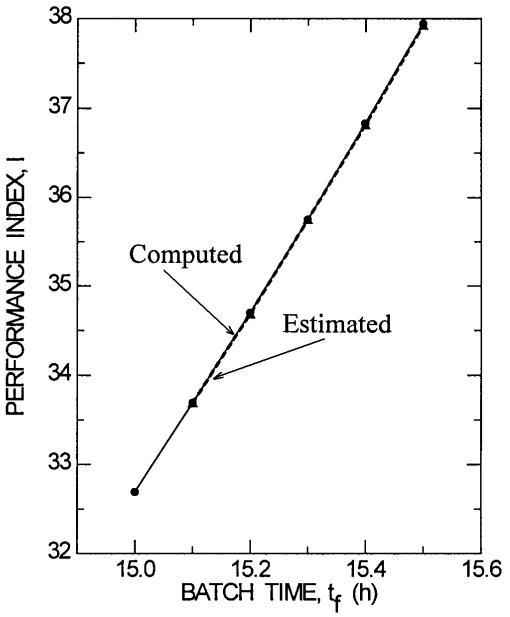

Figure 9.21: Use of the shifting term s to predict the effect of the final time t_f on the performance index

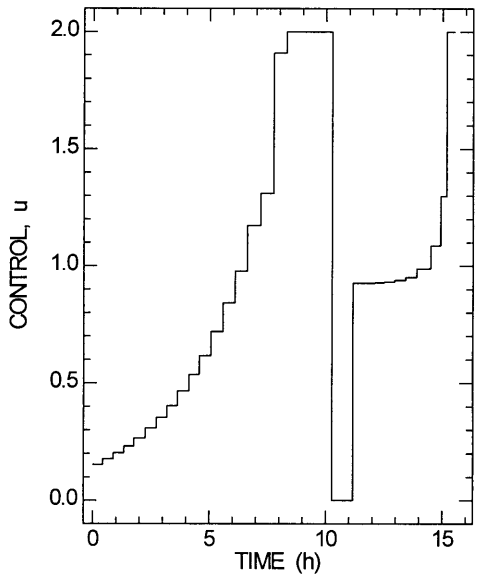

Figure 9.22: Optimal control policy for the fed-batch reactor with $t_f = 15.5$ h, giving $I = 32.9438$

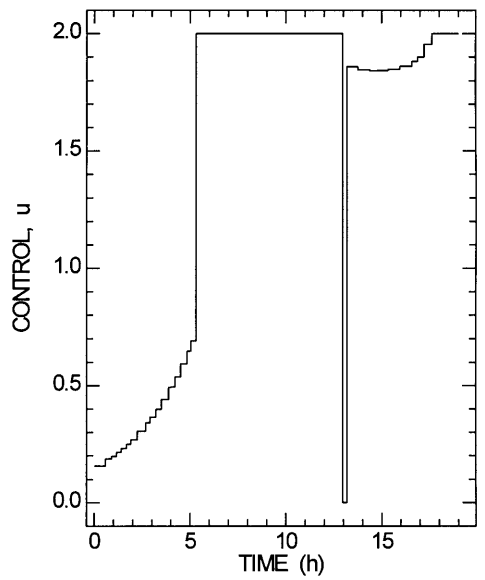

Figure 9.23: Optimal control policy for the fed-batch reactor with $t_f = 19$ h, giving $I = 97.1118$

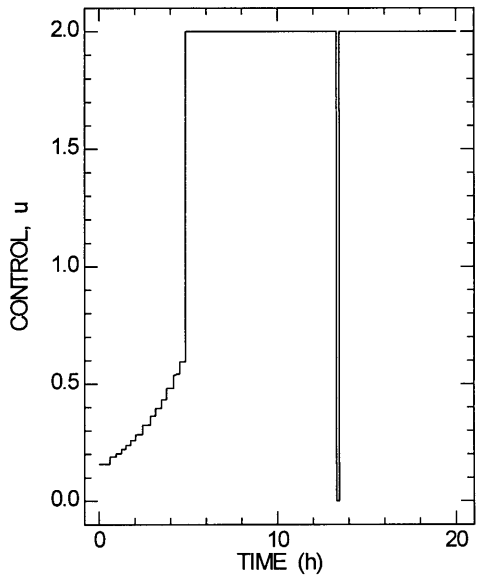

Figure 9.24: Optimal control policy for the fed-batch reactor with $t_f = 20$ h, giving $I = 120.9208$

9.5 References

[1] Aris, R.: "On Denbigh's optimum temperature sequence", *Chem. Eng. Sci.* **12** (1960), 56-64.

[2] Bojkov, B. and Luus, R.: "Optimal control of nonlinear systems with unspecified final times", *Chem. Eng. Sci.* **51** (1996), 905-919.

[3] Denbigh, K.G.: "Optimum temperature sequence in reactors", *Chem. Eng. Sci.* **8** (1958), 125-132.

[4] Gupta, Y.P.: "Semiexhaustive search for solving nonlinear optimal control problems", *Ind. Eng. Chem. Res.* **34** (1995), 3878-3884.

[5] Hull, T.E., Enright, W.D., and Jackson, K.R.: *User Guide to DVERK–a Subroutine for Solving Nonstiff ODE's.* Report 100 (1976), Department of Computer Science, University of Toronto, Canada.

[6] Ko, D.Y.C. and Stevens, W.F.: "Study of singular solutions in dynamic optimization", *AIChE J.* **17** (1971), 160-166.

[7] Luus, R.: "On the optimization of oil shale pyrolysis", *Chem. Eng. Sci.* **33** (1978), 1403-1404.

[8] Luus, R.: "Effect of the choice of final time in optimal control of nonlinear systems", *Can. J. Chem. Eng.* **69** (1991), 144-151.

[9] Luus, R.: "Optimal control of batch reactors by iterative dynamic programming", *J. Process Control* **4** (1994), 217-226.

[10] Luus, R.: "Use of iterative dynamic programming with variable stage lengths and fixed final time", *Hung. J. Ind. Chem.* **24** (1996), 279-284.

[11] Luus, R.: "Determination of the region sizes for LJ optimization procedure", *Hung. J. Ind. Chem.* **26** (1998), 281-286.

[12] Luus, R. and Dong, V.: "Use of iterative dynamic programming in optimization of reaction systems", *Proc. of the 4th IFAC Symposium on Dynamics and Control of Chemical Reactors, Distillation Columns and Batch Processes.* (1995), Helsingor, Denmark, June 7-9, pp. 45-49.

[13] Luus, R. and Hennessy, D.: "Optimization of fed-batch reactors by the Luus-Jaakola optimization procedure", *Ind. Eng. Chem. Res.* **38** (1999), 1948-1955.

[14] Park, S. and Ramirez, W.F.: "Optimal production of secreted protein in fed-batch reactors", *AIChE J.* **34** (1988), 1550-1558.

[15] Reddy, K.V. and Husain, A.: "Computation of optimal control policy with singular subarc", *Can. J. Chem. Eng.* **59** (1981), 557-559.

[16] Rosen, O. and Luus, R.: "Global optimization approach to nonlinear optimal control", *J. Optim. Theory and Applic.* **73** (1992), 547-562.

[17] Wen, C.S. and Yen, T.F.: "Optimization of oil shale pyrolysis", *Chem Eng. Sci.* **32** (1977), 346-349.

[18] Yang, R. L. and Wu, C. P.: "Global optimal control by accelerated simulated annealing", Paper presented at the First Asian Control Conference, Tokyo, Japan, 1994.

Chapter 10

Singular control problems

10.1 Introduction

When the structure of the mathematical model is such that the control vector appears linearly in the state equation, i.e.,

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}) + \mathbf{B}(\mathbf{x})\mathbf{u} \tag{10.1}$$

and if conditions exist such that over some finite time interval(s) the gradient of the Hamiltonian with respect to the control vector

$$\frac{\partial H}{\partial \mathbf{u}} = \mathbf{B}^T(\mathbf{x})\mathbf{z} \tag{10.2}$$

is zero, where $\mathbf{z}$ is the adjoint vector, then we have a *singular* optimal control problem. Singular optimal control problems are very difficult to solve, since the gradient of the Hamiltonian does not provide any information with respect to control when it is zero. In using IDP we are not concerned with such auxiliary variables, as the adjoint vector $\mathbf{z}$ or the Hamiltonian H. Nevertheless, computationally such problems are still difficult since the sensitivity of the performance index with respect to changes in control is very low, and care must be exercised to obtain accurately the optimal control policy. Such low sensitivity of singular control problems was pointed out by Luus [12], and a large number of figures have to be retained in the performance index to get an accurate optimal control policy. The initial intent for the development of IDP was to handle problems of this type [9-11], but now it is apparent that there is a wide variety of problems for which IDP is well suited.

We have already encountered a singular optimal control problem in Chapter 9, dealing with the determination of the optimal feed rate policy for the fed-batch reactor problem of Park and Ramirez [16]. Although care must be taken, we saw that the solution can be obtained quite readily with IDP, using the computer program for variable stage lengths given in Appendix D. Due to the low sensitivity, a large number of passes is usually required to obtain an adequately accurate solution.

In this chapter the goal is to present initially several singular optimal control problems that have puzzled researchers for over two decades, three of which were solved just recently by Luus [13]. The aim is to show that even simple-looking problems can be very challenging, and to provide guidelines of how to approach a problem when the solution does not arise in a single computer run. We conclude the chapter by including a realistic optimal control problem from chemical engineering that has been very difficult to solve by using methods based on Pontryagin's maximum principle, but can be solved quite readily with IDP. To solve singular optimal control problems we use stages of varying length, and a quadratic penalty function with a shifting term for the final time as described in Chapter 9. For integration we use the subroutine DVERK [6], which is listed in Appendix E, with a local error tolerance of 10^{-6}, unless otherwise stated.

10.2 Four simple-looking examples

10.2.1 Example 1

Let us consider a simple example that can be solved analytically simply by using one's intuition. The problem was considered by Jacobson *et al.* [7] and more recently by Chen and Huang [1] who showed that getting the solution numerically is not quite as easy. The equations of the system are

$$\frac{dx_1}{dt} = u \tag{10.3}$$

$$\frac{dx_2}{dt} = 0.5x_1^2 \tag{10.4}$$

with $\mathbf{x}(0) = [\ 1\ \ 0\ ]^T$, and final time $t_f = 2$. The constraints on the control are

$$-1 \leq u \leq 1. \tag{10.5}$$

It is required to find the control policy that minimizes the performance index

$$I = x_2(t_f). \tag{10.6}$$

It is obvious that to minimize the performance index, we should decrease the value of x_1 as fast as possible, which means using the control policy $u = -1$ until x_1 reaches zero, and then use $u = 0$. The value of the performance index then is

$$I = 0.5\int_0^1 (1-t)^2 dt = \frac{1}{6}, \tag{10.7}$$

since the value of the integral from $t = 1$ to $t = 2$ is zero. Although the result is obtained very easily analytically, numerically it has been much more difficult to get this result.

Jacobson *et al.* [7] reported a value $I = 0.1717$, and Chen and Huang [1] managed to improve the result to $I = 0.1683$, which is still quite far from the optimal value of $I = 0.1666667$.

To put the problem into the form that allows us to use IDP with variable stage lengths, we introduce the augmented performance index

$$J = x_2(t_f) + \theta(t_{fc} - t_f - s)^2 \tag{10.8}$$

where t_{fc} is the sum of the variable stage lengths, θ is a positive penalty function factor, and s is a shifting term, chosen initially zero, and is then updated after every pass by the deviation of the calculated final time from the specified final time t_f

$$s^{q+1} = s^q - (t_{fc} - t_f) \tag{10.9}$$

where q is the pass number. The goal then is to minimize the augmented performance index J. However, to keep track with respect to progress made, we report the value of the original performance index I and check whether the final time that is calculated is sufficiently close to the specified final time.

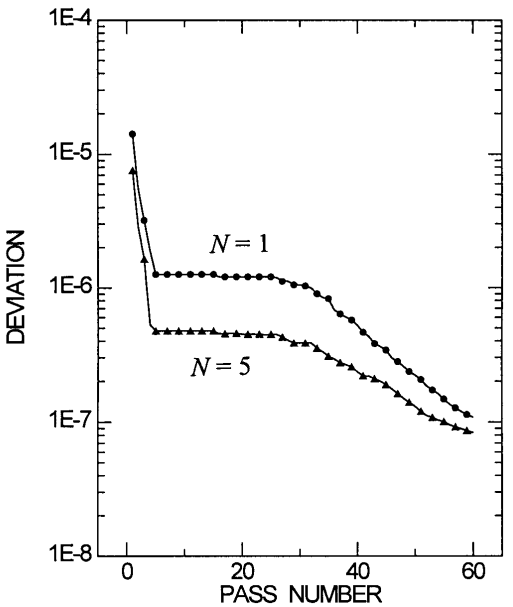

Figure 10.1: Convergence profile for Example 1 with $P = 20$ time stages, showing the deviation from $I = 0.1666666$

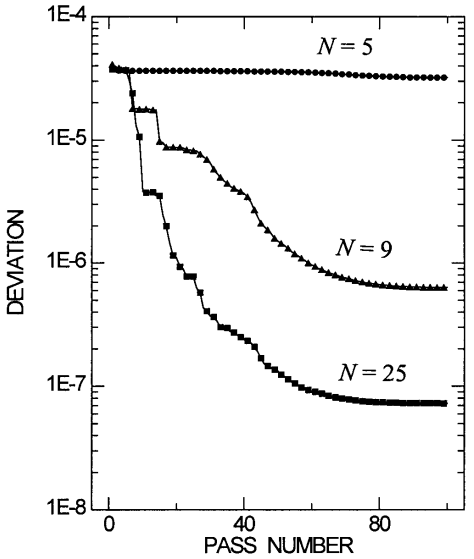

Figure 10.2: Convergence profile for Example 1 with $P = 15$ time stages, showing the deviation from $I = 0.1666666$

To solve this as a preliminary problem by IDP, we chose $P = 20$ stages, initially each of length 0.1, $\theta = 100$, 60 passes, each of 30 iterations with $\gamma = 0.95$ and $\eta = 0.88$. We chose an initial control policy of $u = 0$, initial region size for the control 1.0, and initial region size for the stage lengths 0.2. For the first 5 passes, the stage-lengths were kept constant and after this the stage-lengths were considered as additional control variables, as in Chapter 9. The number of randomly chosen points was chosen to be $R = 25$, and the number of grid points N was allowed to be a parameter, chosen as 1 for the first run and 5 for the second run. As is shown in Figure 10.1, rapid convergence to $I = 0.1666667$ was obtained with both $N = 1$ and $N = 5$ grid points, yielding the control policy obtained analytically, i.e., $u = -1$ for the time interval $0 \leq t < 1$ followed by $u = 0$. The choice of $P = 20$ made the problem very easy, since the switching occurred exactly at the end of a sampling period. Suppose we had chosen $P = 15$. Then the use of flexible stage-lengths should still allow accurate switching from -1 to zero, but the convergence rate is expected to be slower.

A series of runs was performed with $P = 15$ stages with the same parameters for IDP as earlier with $P = 20$. As is shown in Figure 10.2, the choice of $N = 5$ grid points is no longer adequate. After 100 passes, convergence to $I = 0.1666988$ occurs. With $N = 9$ we get convergence to $I = 0.1666672$, and with $N = 25$, convergence to $I = 0.1666667$ results, giving the switch from -1.0 to 0.0 at almost exactly at $t = 1.000$. The optimal control policy obtained with $N = 25$ is shown in Figure 10.3.

Therefore, for this seemingly simple singular control problem we require more than just a few grid points to ensure accurate switching time. If multiple switchings are required, then it is even more important to use time stages of varying lengths.

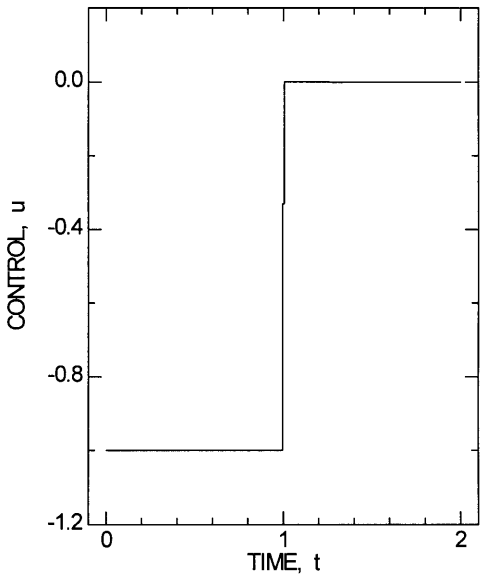

Figure 10.3: Optimal control policy for Example 1 obtained by using $P = 15$ stages of variable length

10.2.2 Example 2

We now consider a slightly more difficult optimal control problem used by Jacobson *et al.* [7], and used for optimal control studies by Flaherty and O'Malley [5], Dadebo and McAuley [2], and Luus [13]. There are only two state variables in the original formulation of the problem, but we have introduced the third variable to give, at the final time, the performance index. The system equations are

$$\frac{dx_1}{dt} = x_2 \tag{10.10}$$

$$\frac{dx_2}{dt} = u \tag{10.11}$$

$$\frac{dx_3}{dt} = x_1^2 \tag{10.12}$$

with $\mathbf{x}(0) = [\ 0\ \ 1\ \ 0\ ]^T$, and final time $t_f = 5$. The constraints on the control are

$$-1 \leq u \leq 1. \tag{10.13}$$

It is required to find the control policy that minimizes the performance index

$$I = x_3(t_f). \tag{10.14}$$

For this problem Jacobson *et al.* [7] reported a minimum performance index $I = 0.2771$. Flaherty and O'Malley [5] reported an improved value of $I = 0.269$, which was

also obtained by Dadebo and McAuley [2]. By using IDP with flexible stage-lengths Luus [13] reported a further improvement to $I = 0.2683941$ with a 4-stage control policy and $I = 0.2683938$ with a 5-stage control policy. Here we wish to provide some computational details to show how IDP can be used to solve this optimal control problem. Since we are using stages of varying lengths, as before, we introduce the augmented performance index

$$J = x_3(t_f) + \theta(t_{fc} - t_f - s)^2 \tag{10.15}$$

where t_{fc} is the sum of the variable stage lengths, θ is a positive penalty function factor, and s is a shifting term, chosen initially zero, and is then updated after every pass by the deviation of the calculated final time from the specified final time t_f

$$s^{q+1} = s^q - (t_{fc} - t_f) \tag{10.16}$$

where q is the pass number. The goal then is to minimize the augmented performance index J. As before, to keep track with respect to progress made in solving the optimal control problem, we report the value of the original performance index I and check whether the final calculated time is sufficiently close to the specified final time $t_f = 5$.

The parameters chosen to solve this problem were essentially the same as in the previous example, except here we chose $P = 10$ stages rather than 20 stages. With $R = 25$, $\gamma = 0.95$, $\eta = 0.88$, 30 iterations per pass, and allowing 100 passes, the convergence rate was very much dependent on the choice of the number of grid points N. Generally lower values of the performance index were obtained with larger N, as is shown in Table 10.1. In each case, after 100 passes the calculated value of the final time was 5.0000.

The convergence profiles for three values of N are shown in Figure 10.4. It is clear that no matter how many grid points are taken, it is not likely that the optimum control policy is obtained in a single run with initial control policy of zero.

Also, it is wise not to take too many grid points, since the computation time increases substantially when N is increased, as is shown in Figure 10.5. The best approach, therefore, is to perform several runs with a relatively small number of grid points, using as the initial control policy the best control policy that can be chosen by using the information gained in the previous run. To illustrate this approach, we examine the control policies in Figures 10.6 and 10.7, and note that the control for the first two stages is on the lower and upper bounds respectively. The control at the third stage is becoming more negative as the performance index is improved very slightly from $I = 0.268423$ obtained with $N = 25$ to $I = 0.268405$ with $N = 101$ grid points.

Table 10.1: Effect of the number of grid points on the value of the performance index after 100 passes

Number of grid points N	Performance index after 100 passes I
1	0.2717966
3	0.2717966
5	0.2708626
7	0.2700064
9	0.2686786
11	0.2685747
13	0.2685627
15	0.2686263
17	0.2685267
19	0.2684801
21	0.2684809
25	0.2684234
37	0.2684354
51	0.2684088
101	0.2684049

Thus a run was performed where the control policy resulting with $N = 25$ after 100 passes was adjusted to make the control at the third stage more negative and a bit shorter, before being used as an initial control policy with $P = 5$ stages. The resulting optimal control policy, consisting of only 4 stages, is shown in Figure 10.8. It gives an improved value of the performance index of $I = 0.2683941$. Now by adding another stage, and rerunning the program, we get the 5-stage optimal control policy shown in Figure 10.9 with $I = 0.2683938$. The addition of the extra stage thus causes an insignificant improvement in the performance index.

This example shows that interpretation of results after a computer run plays an important part in establishing the optimal control policy efficiently. In this particular case, the approach described was used to get the optimal control policy with little computational effort.

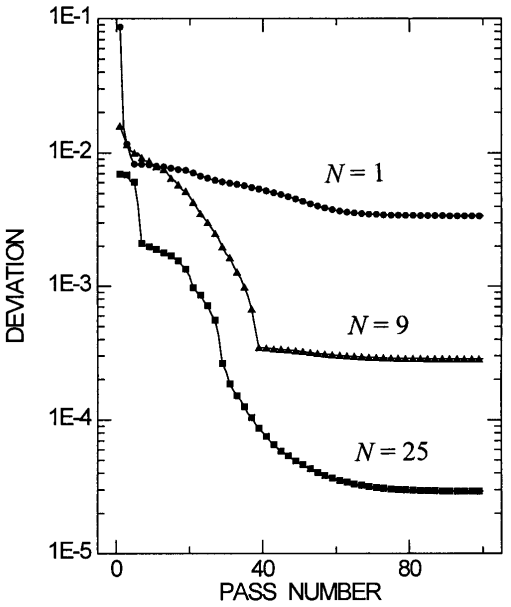

Figure 10.4: Convergence profile for Example 2, showing the deviation from $I = 0.2683941$

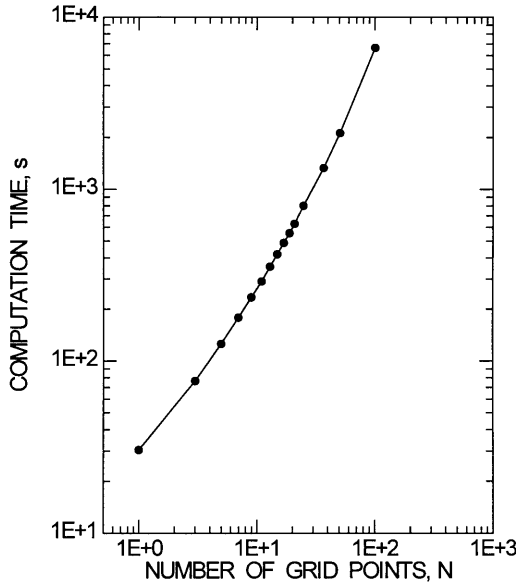

Figure 10.5: Effect of the number of grid points on the computation time on a Pentium3/500 for Example 2 with $R = 25$ and 100 passes, each consisting of 30 iterations

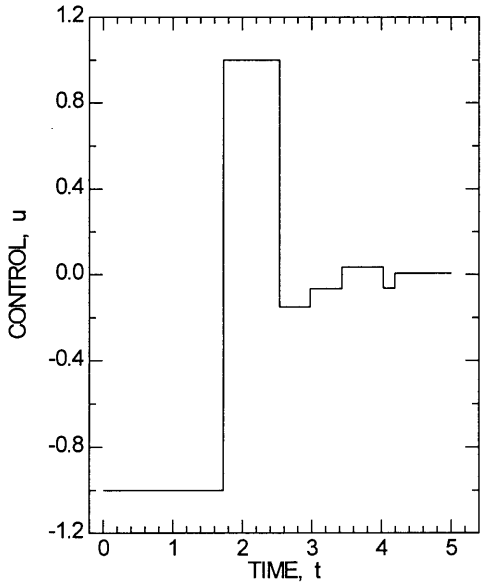

Figure 10.6: Control policy for Example 2, resulting from the first run with the use of $N = 25$ grid points, giving $I = 0.268423$

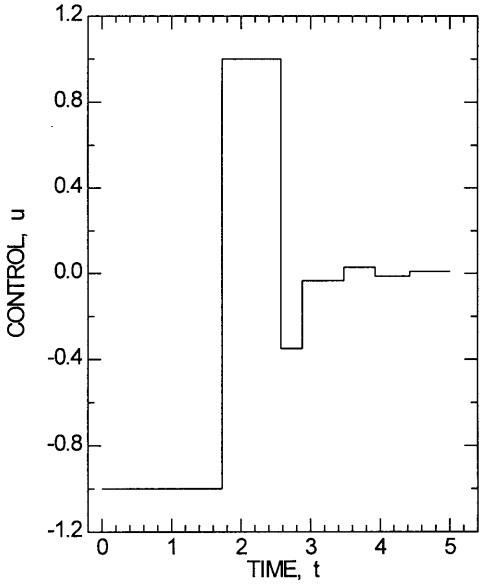

Figure 10.7: Control policy for Example 2, resulting from the use of $N = 101$ grid points, giving $I = 0.268405$

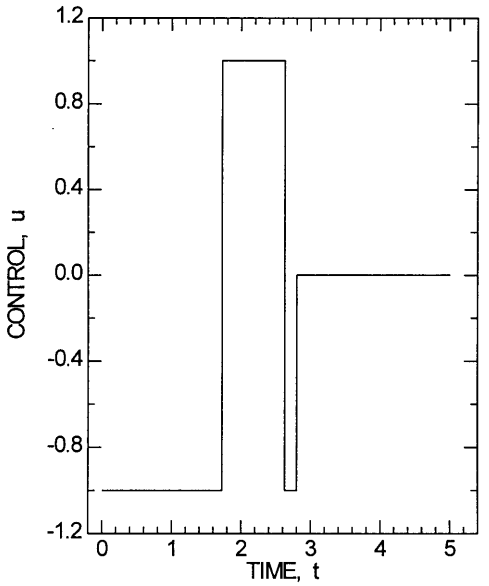

Figure 10.8: Optimal 4-stage control policy for Example 2, giving $I = 0.2683941$

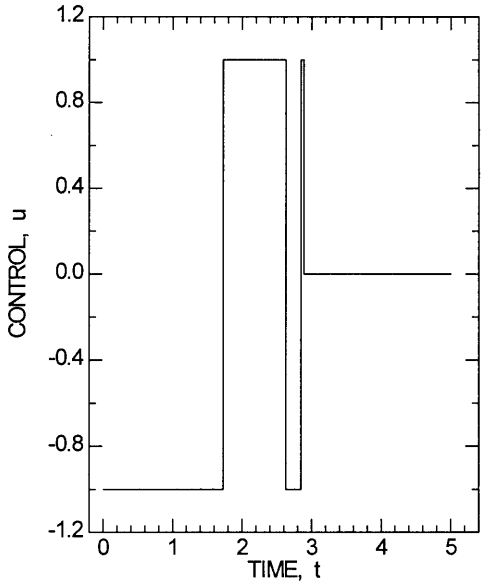

Figure 10.9: Optimal 5-stage control policy for Example 2, giving $I = 0.2683938$

10.2.3 Example 3

In this example the quadratic term x_2^2 is added to the integrand of the performance index in Example 2, so the system of equations is

$$\frac{dx_1}{dt} = x_2 \tag{10.17}$$

$$\frac{dx_2}{dt} = u \tag{10.18}$$

$$\frac{dx_3}{dt} = x_1^2 + x_2^2 \tag{10.19}$$

with $\mathbf{x}(0) = [\ 0\ \ 1\ \ 0\ ]^T$, and final time $t_f = 5$. The constraints on the control are

$$-1 \leq u \leq 1. \tag{10.20}$$

It is required to find the control policy that minimizes the performance index

$$I = x_3(t_f). \tag{10.21}$$

For this problem Jacobson *et al.* [7] reported a minimum performance index $I = 0.8285$, which was improved by Dadebo and McAuley [2] to $I = 0.754016$ by the use of 80 time stages of equal length, and further improved by Luus [13] to $I = 0.7539848$ with the use of a 10-stage control policy, where the stages were of varying lengths. Since we are using stages of varying lengths, as before, we introduce the augmented performance index

$$J = x_3(t_f) + \theta(t_{fc} - t_f - s)^2 \tag{10.22}$$

where t_{fc} is the sum of the variable stage lengths, θ is a positive penalty function factor and s is a shifting term. As before, the shifting term is chosen initially zero, and is then updated after every pass by the deviation of the calculated final time from the specified final time t_f

$$s^{q+1} = s^q - (t_{fc} - t_f) \tag{10.23}$$

where q is the pass number.

The additional term x_2^2 in Eq. (10.19) makes the problem easier to solve than Example 2. With the use of a single grid point $N = 1$, convergence was rapid and systematic, leading to the control policy in Figure 10.10, yielding $I = 0.7539845$. The first 6 stages collapsed into a single stage at the beginning, giving in essence a 15-stage optimal control policy. By using now a larger number of stages, and rerunning the problem, the performance index can be improved marginally, as was done by Luus [13] to get a 19-stage optimal control policy with $I = 0.7539839$ with the use of 19 stages. The improvement, however, is very slight, and quite negligible.

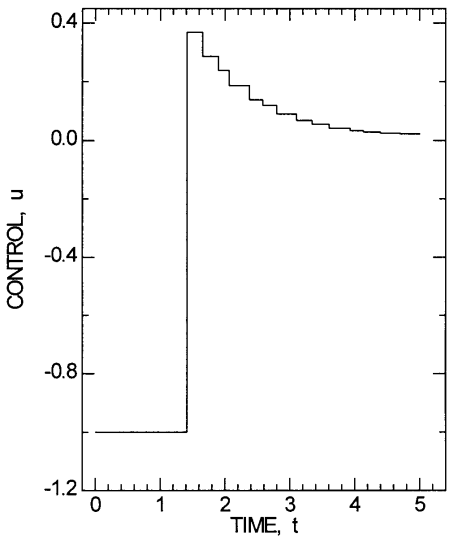

Figure 10.10: Optimal control policy consisting of 15 stages for Example 3, giving $I = 0.7539845$

10.2.4 Example 4

A third-order system considered by Flaherty and O'Malley [5] consists of the following equations

$$\frac{dx_1}{dt} = x_2 \tag{10.24}$$

$$\frac{dx_2}{dt} = x_3 \tag{10.25}$$

$$\frac{dx_3}{dt} = u \tag{10.26}$$

$$\frac{dx_4}{dt} = x_1^2 \tag{10.27}$$

with $\mathbf{x}(0) = [\ 1\ \ 0\ \ 0\ \ 0\ ]^T$, and final time $t_f = 5$. The constraints on the control are

$$-1 \leq u \leq 1. \tag{10.28}$$

It is required to find the control policy that minimizes the performance index

$$I = x_4(t_f). \tag{10.29}$$

Flaherty and O'Malley [5] stated that the optimal control policy consists of an infinite number of switches, yielding a value of $I = 1.2521$, and provided an approximate technique that gave a finite number of switches and $I = 1.2665$. By using IDP with variable stage lengths, Luus [13] obtained an improved value $I = 1.25211$ with only 4

switches, giving a 5-stage control policy. To use stages of flexible length, we consider the augmented performance index

$$J = x_4(t_f) + \theta(t_{fc} - t_f - s)^2 \tag{10.30}$$

where t_{fc} is the sum of the variable stage lengths, θ is a positive penalty function factor and s is a shifting term. The shifting term is chosen initially zero, and is then updated after every pass by the deviation of the calculated final time from the specified final time t_f

$$s^{q+1} = s^q - (t_{fc} - t_f) \tag{10.31}$$

where q is the pass number.

To solve this problem with IDP, we used the same parameters as in the other examples, except we took a tighter tolerance for integration with the DVERK subroutine using a local error tolerance of 10^{-8}, rather than 10^{-6}, and we used the penalty function factor θ as a parameter.

As can be seen in Figure 10.11, the effect of the penalty function factor is quite pronounced. Here it is very important to allow the length of the time stages to vary, and so the use of a very small value $\theta = 0.001$ gives the optimal control policy given in Figure 10.12. The low sensitivity can be appreciated by comparing this result to the previously obtained control policy by Luus [13], which is only 0.00004% higher than the present result, yet the last two stages of the control policy are quite different. The problem of low sensitivity is considered in greater detail in Chapter 14.

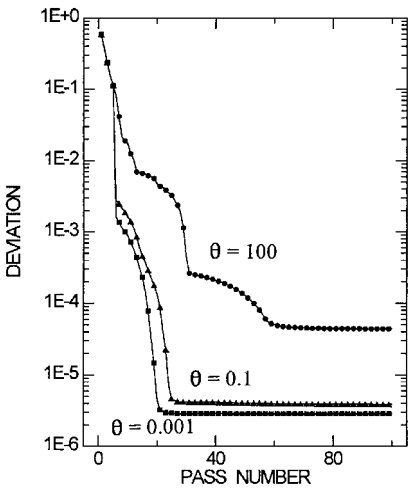

Figure 10.11: Effect of the penalty function factor θ on convergence for Example 4, showing the deviation from $I = 1.25211$ with the use of $R = 25$, $\gamma = 0.95$, $\eta = 0.88$, $N = 1$ and 30 iterations in each pass

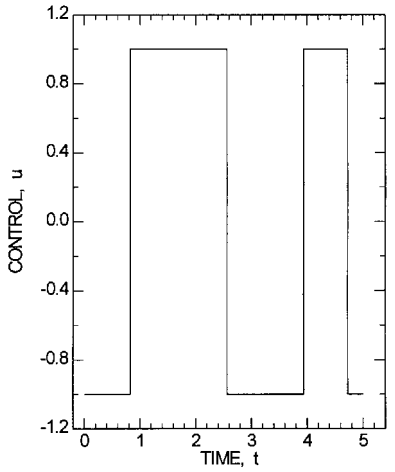

Figure 10.12: Optimal control policy for Example 4, giving $I = 1.2521128$

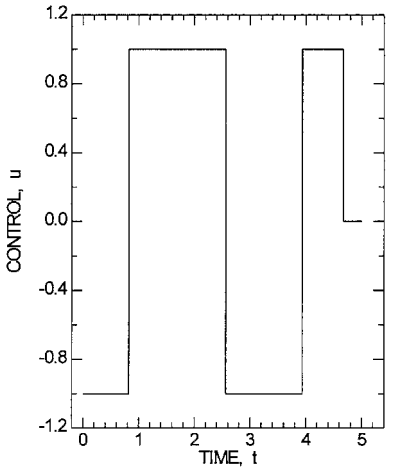

Figure 10.13: Control policy for Example 4, giving $I = 1.2521134$ obtained by Luus [13]

10.3 Yeo's singular control problem

One of the earliest examples tried with iterative dynamic programming ([9], [10]) is the singular control problem used by Yeo [18] to illustrate the use of quasilinearization to solve nonlinear singular control problems. The system is described by

$$\frac{dx_1}{dt} = x_2 \tag{10.32}$$

$$\frac{dx_2}{dt} = -x_3 u + 16x_5 - 8 \tag{10.33}$$

$$\frac{dx_3}{dt} = u \tag{10.34}$$

$$\frac{dx_4}{dt} = x_1^2 + x_2^2 + 0.0005(x_2 + 16x_5 - 8 - 0.1x_3 u^2)^2 \tag{10.35}$$

$$\frac{dx_5}{dt} = 1 \tag{10.36}$$

with $\mathbf{x}(0) = [\ 0 \quad -1 \quad -\sqrt{5} \quad 0 \quad 0\ ]^T$, and final time $t_f = 1$. The constraints on the control are

$$-4 \le u \le 10. \tag{10.37}$$

It is required to find the control policy that minimizes the performance index

$$I = x_4(t_f). \tag{10.38}$$

The optimal value of the performance index depends very much on the number of time stages, if the stages are of equal length, as was shown by Luus [9], and is given in Table 10.2. Although there is an improvement in the performance index as the number of time stages is increased, this improvement is not uniform because the optimal control policy involves two switches and the time of the switches is important. We now consider this optimal control problem by using stages of variable lengths, as we did for the other examples, and consider the augmented performance index

$$J = x_4(t_f) + \theta(t_{fc} - t_f - s)^2 \tag{10.39}$$

where t_{fc} is the sum of the variable stage lengths, the penalty function factor θ is chosen to be 1, and s is a shifting parameter, chosen initially zero, and is then updated after every pass by the deviation of the calculated final time from the specified final time t_f

$$s^{q+1} = s^q - (t_{fc} - t_f) \tag{10.40}$$

where q is the pass number.

To minimize the augmented performance index we choose $P = 25$ stages, each of length 0.04 initially, $R = 25$, $\gamma = 0.95$, $\eta = 0.88$, 30 iterations per pass, and allow

Table 10.2: Effect of the number of time stages P of equal length on the minimum value of the performance index

Number of time stages P	Optimal performance index I
10	0.12011
11	0.11957
12	0.11952
13	0.11975
14	0.11973
15	0.11954
16	0.11951
17	0.11961
18	0.11960
19	0.11948
20	0.11944
21	0.11947
22	0.11944
23	0.11937
24	0.11934
25	0.11937
26	0.11936
27	0.11934
28	0.11933
29	0.11936
30	0.11934
31	0.11935
56	0.11927

100 passes. The initial value for control was chosen as 3.0 for each time stage. The initial region size for the control was 7.0 and for stage lengths, 0.01. For the first 5 passes the stage lengths were kept at their original lengths.

After 100 passes requiring 855 s of computation time on a PentiumII/350, a performance index $I = 0.11928$ was obtained with $t_f = 1.000000$, and shifting parameter $s = 0.02233$. The optimal control policy obtained is shown in Figure 10.14. There are only 13 stages in the optimal control policy. The stages which were not on the boundary were split and another run was made, giving the resulting control policy in Figure 10.15. The refined control policy gave a slight improvement in the performance index from $I = 0.11928$ to $I = 0.11926$. This is a very slight improvement over $I = 0.11927$ obtained by Luus [9] with the use of 56 stages of equal length.

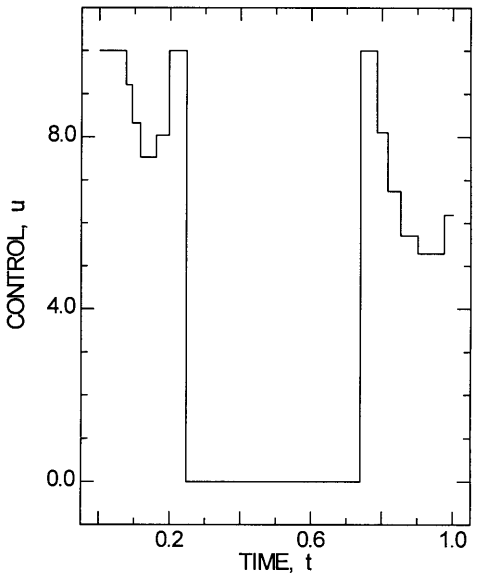

Figure 10.14: Optimal control policy, consisting of 13 stages, for Yeo's problem, giving $I = 0.11928$, obtained by using initially 25 stages

10.4 Nonlinear two-stage CSTR problem

Let us now consider the two-stage CSTR system used for optimal control studies by Edgar and Lapidus [4], Luus [8], Nishida *et al.* [15], Rosen and Luus [17], Dadebo and McAuley [3], and by Luus *et al.* [14]. The system is described by

$$\frac{dx_1}{dt} = -3x_1 + g_1(\mathbf{x}) \tag{10.41}$$

$$\frac{dx_2}{dt} = -11.1558x_2 + g_1(\mathbf{x}) - 8.1558(x_2 + 0.1592)u_1 \tag{10.42}$$

$$\frac{dx_3}{dt} = 1.5(0.5x_1 - x_3) + g_2(\mathbf{x}) \tag{10.43}$$

$$\frac{dx_4}{dt} = 0.75x_2 - 4.9385x_4 + g_2(\mathbf{x}) - 3.4385(x_4 + 0.122)u_2 \tag{10.44}$$

where the reaction term in the first tank is given by

$$\begin{aligned} g_1(\mathbf{x}) &= 1.5 \times 10^7(0.5251 - x_1)\exp(\frac{-10}{x_2 + 0.6932}) \\ &\quad -1.5 \times 10^{10}(0.4748 + x_1)\exp(\frac{-15}{x_2 + 0.6932}) - 1.4280 \end{aligned} \tag{10.45}$$

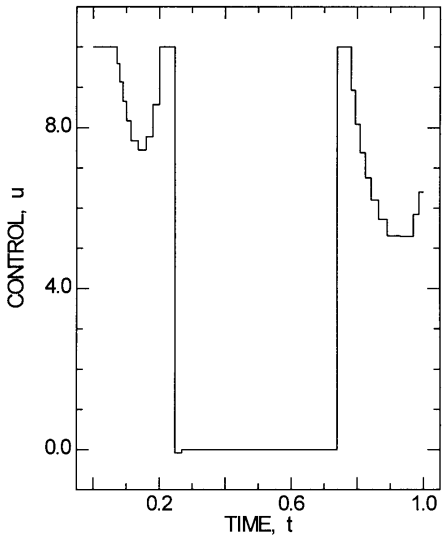

Figure 10.15: Optimal control policy for Yeo's problem, giving $I = 0.11926$, obtained by using initially 24 stages of variable lengths

and the reaction term in the second tank is

$$\begin{aligned} g_2(\mathbf{x}) &= 1.5 \times 10^7 (0.4236 - x_2) \exp\left(\frac{-10}{x_4 + 0.6560}\right) \\ &\quad -1.5 \times 10^{10} (0.5764 + x_3) \exp\left(\frac{-15}{x_4 + 0.6560}\right) - 0.5086. \end{aligned} \tag{10.46}$$

The initial condition is given by

$$\mathbf{x}(0) = [\ 0.1962 \quad -0.0372 \quad 0.0946 \quad 0\ ]^T \tag{10.47}$$

and the constraints on the control are

$$-1 \leq u_j \leq 1, \quad j = 1, 2. \tag{10.48}$$

The state variables x_1 and x_3 are normalized concentration variables in tanks 1 and 2 respectively, and x_2 and x_4 are normalized temperature variables in tanks 1 and 2 respectively. It is required to find the control policy that minimizes the performance index

$$I = x_1^2(t_f) + x_2^2(t_f) + x_3^2(t_f) + x_4^2(t_f) = \mathbf{x}^T(t_f)\mathbf{x}(t_f) \tag{10.49}$$

where the dimensionless final time t_f is not specified, but we would like it to be reasonably small.

Therefore, to penalize the final time, we introduce the augmented performance index where we append the final time multiplied by the penalty function factor θ to the original performance index

$$J = I + \theta t_f \tag{10.50}$$

Table 10.3: Effect of the penalty function factor θ on the value of the performance index and the final time

Penalty function factor θ	Performance index $I = \mathbf{x}^T(t_f)\mathbf{x}(t_f)$	Final time t_f
1×10^{-7}	1.8470×10^{-13}	0.32498
1×10^{-6}	1.6699×10^{-11}	0.32479
3×10^{-6}	1.4728×10^{-10}	0.32444
1×10^{-5}	1.6267×10^{-9}	0.32361
1.2×10^{-5}	2.3350×10^{-9}	0.32353
1.5×10^{-5}	3.6344×10^{-9}	0.32343
3×10^{-5}	1.4735×10^{-8}	0.32316
1×10^{-4}	1.4325×10^{-7}	0.32078

and use IDP to minimize this augmented performance for several values of θ.

With the use of IDP there were no difficulties in using the flexible stage lengths, except to maintain stability, the local error tolerance in DVERK [6] had to be taken very small, namely 10^{-10}. As is seen in Table 10.3, as expected, when θ is increased, then the final time at the optimum is decreased. There is a trade-off between the value of the performance index I and the final time t_f.

For the case with $\theta = 1.2 \times 10^{-5}$, the control policy is given in Figure 10.16 and the resulting state trajectories are given in Figure 10.17. The optimal control policy is very similar to that obtained by Luus [8], and by Luus *et al.* [14]. It can be readily seen that all state variables have reached the origin within adequate tolerance, in line with the 4 decimal point accuracy to which the initial conditions are given.

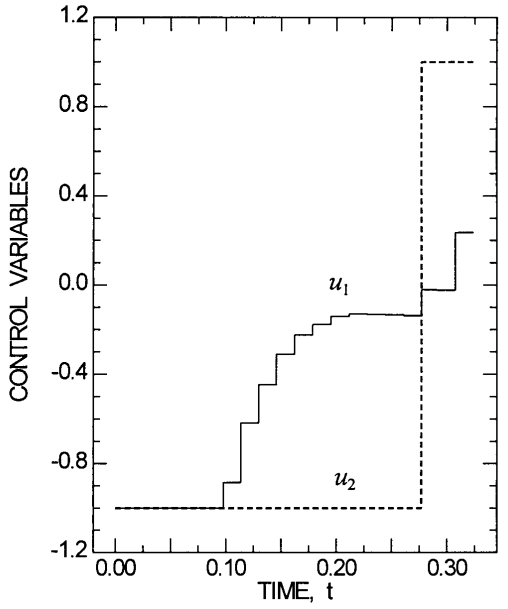

Figure 10.16: Optimal control policy for the two-stage CSTR problem giving $I = 2.335 \times 10^{-9}$ with $t_f = 0.32353$

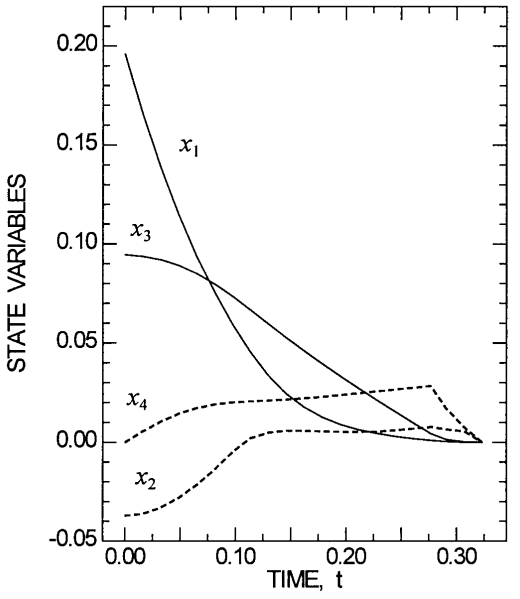

Figure 10.17: Trajectories of the state variables resulting from the use of the optimal control policy shown in Figure 10.16

10.5 References

[1] Chen, Y. and Huang, J.: "A new computational approach to solving a class of optimal control problems", *Int. J. Control* **58** (1993), 1361-1383.

[2] Dadebo, S.A. and McAuley, K.B.: "On the computation of optimal singular controls", *Proc. 1995 IEEE Conf. on Control Applications* (1995), Albany, NY, Sept. 28-29, 150-155.

[3] Dadebo, S.A. and McAuley, K.B.: "Dynamic optimization of constrained chemical engineering problems using dynamic programming", *Comput. Chem. Eng.* **19**(1995), 513-525.

[4] Edgar, T.F. and Lapidus, L.: "The computation of optimal singular bang-bang control II. Nonlinear systems", *AIChE J.* **18** (1972), 780-785.

[5] Flaherty, J.E. and O'Malley, R.E.: "On the computation of singular controls", *IEEE Trans on Auto. Control* **22** (1977), 640-648.

[6] Hull, T.E., Enright, W.D., and Jackson, K.R.: *User Guide to DVERK–a Subroutine for Solving Nonstiff ODE's.* Report 100 (1976), Department of Computer Science, University of Toronto, Canada.

[7] Jacobson, D.H., Gershwin, S., and Lele, M.M.: "Computation of optimal singular controls", *IEEE Trans. on Auto. Control* **15** (1970), 67-73.

[8] Luus, R.: "Optimal control by direct search on feedback gain matrix", *Chem. Eng. Sci.* **29** (1974), 1013-1017.

[9] Luus, R.: "Optimal control by dynamic programming using accessible grid points and region reduction", *Hung. J. Ind. Chem.* **17** (1989), 523-543.

[10] Luus, R.: "Optimal control by dynamic programming using systematic reduction in grid size", *Int. J. Control* **19** (1990), 995-1013.

[11] Luus, R.: "On the application of iterative dynamic programming to singular optimal control problems", *IEEE Trans. Auto. Contr.* **37** (1992), 1802-1806.

[12] Luus, R.:"Sensitivity of control policy on yield of a fed-batch reactor", *Proc. of IASTED International Conference on Modelling and Simulation* (1995), Pittsburgh, PA, April 27-29, pp. 224-226.

[13] Luus, R.:"Use of iterative dynamic programming for optimal singular control problems", *Proc. of IASTED International Conference CONTROL '97.* (1997), Cancun, Mexico, May 28-31, pp. 286-289.

[14] Luus, R., Mekarapiruk, W., and Storey, C.:"Evaluation of penalty functions for optimal control", *Proc. of International Conference on Optimization Techniques and Applications (ICOTA '98)* (1998), Perth, Western Australia, July 1-3, 1998, Curtin Printing Services, pp. 724-731.

[15] Nishida, N., Liu, Y.A., Lapidus, L., and Hiratsuka, S.: "An effective computational algorithm for suboptimal singular and/or bang-bang control", *AIChE J.* **22** (1976), 505-523.

[16] Park, S. and Ramirez, W.F.: "Optimal production of secreted protein in fed-batch reactors", *AIChE J.* **34** (1988), 1550-1558.

[17] Rosen, O. and Luus, R.: "Evaluation of gradients for piecewise constant optimal control", *Comput. Chem. Eng.* **15** (1991), 273-281.

[18] Yeo, B.P.: "A modified quasilinearization algorithm for the computation of optimal singular control", *Int. J. Control* **32** (1980), 723-730.

Chapter 11

State constraints

11.1 Introduction

In optimal control problems constraints are frequently imposed on state variables. For example, if one of the state variables is the reactor temperature, then it should be quite obvious that there should be an upper limit on the temperature. If a state variable is the volume of the reaction mixture, this volume should not surpass the physical volume of the reactor. Such constraints take a form of inequalities that should be incorporated into the model of the system.

Another occasion when we encounter state constraints is in transferring a system from one state of operation to another. For example, in starting up a reactor, the system starting conditions are usually not the same as the desired conditions of the system operation. In these situations the constraints on the system state variables are in the form of equality constraints at the final time t_f, where we want some of the state variables to have the specified desired values at time t_f. The methods of dealing with these two types of state constraints are somewhat different; so, we consider these separately.

11.2 Final state constraints

11.2.1 Problem formulation

Let us consider the system described by the differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{11.1}$$

where the initial state $\mathbf{x}(0)$ is given. The state vector is $\mathbf{x}$ is an $(n \times 1)$ vector and $\mathbf{u}$ is an $(m \times 1)$ control vector bounded by

$$\alpha_j \leq u_j \leq \beta_j, \quad j = 1, 2, \cdots, m. \tag{11.2}$$

At the final time t_f, there are k state variables that must be at the desired values, i.e.,

$$h_i = x_i(t_f) - x_i^d = 0, \quad i = 1, 2, \cdots, k \leq n. \tag{11.3}$$

The optimal control problem is to find the control policy $\mathbf{u}(t)$ in the time interval $0 \leq t < t_f$ that minimizes the performance index

$$I = \Phi(\mathbf{x}(t_f)) \tag{11.4}$$

where the final time t_f is specified. This problem differs from what we have considered earlier by having the k equality constraints that must be satisfied at the final time. Numerically the equality constraints are expressed explicitly as

$$|x_i(t_f) - x_i^d| \leq \epsilon_i, \qquad i = 1, 2, \cdots, k, \tag{11.5}$$

where ϵ_i is the tolerance for the i^{th} constraint and is chosen sufficiently small. We would also like to know how the choice of the tolerance ϵ_i affects the value of the performance index I at the optimum.

11.2.2 Quadratic penalty function with shifting terms

Let us construct an augmented performance index to take into account the k equality constraints by adding to the performance index a quadratic penalty function containing shifting terms

$$J = I + \theta \sum_{i=1}^{k} (h_i - s_i)^2 \tag{11.6}$$

where θ is a positive penalty function factor and the shifting terms s_i are chosen so that computationally Eq. (11.3) becomes satisfied and the performance index in Eq. (11.4) becomes minimized as the iterative procedure progresses. The use of such shifting terms in the quadratic penalty function was first explored by Luus [13] for steady state optimization. It is noted that

$$(h_i - s_i)^2 = h_i^2 - 2s_i h_i + s_i^2 \tag{11.7}$$

so that near the optimum when Eq. (11.3) is almost satisfied, the term h_i^2 is negligibly small with respect to the other two terms and the minimization of the augmented performance index in Eq. (11.6) becomes equivalent to minimizing

$$J = I - 2\theta \sum_{i=1}^{k} s_i h_i, \tag{11.8}$$

since the term involving the square of the shifting term is simply a constant. Thus, the product $-2\theta s_i$ gives the Lagrange multiplier or sensitivity of the performance index to the change in the i^{th} equality constraint in Eq. (11.3). This aspect of the

shifting term was pointed out by Luus [13], in using penalty functions in dealing with difficult equality constraints in steady state optimization, and has been outlined in Chapter 2. The use of shifting terms was illustrated with IDP by Luus [14], and by Luus and Storey [19], and was outlined here in Chapter 10. When prior information on the sensitivity is not available, we choose initially the shifting term s_i equal to zero, and then, after every pass of IDP, update it according to

$$s_i^{q+1} = s_i^q - h_i \tag{11.9}$$

where s_i^q is the value of s_i at pass number q. We may choose to update the shifting terms after every iteration instead of after every pass if a very small number of passes is used. To show the computational aspects of the use of the quadratic penalty function with shifting terms, we consider several examples.

Example 1

Let us consider a very simple example with final state constraint that can be readily solved by Pontryagin's maximum principle. This problem was considered by Goh and Teo [4] and optimized by IDP by Luus [10] and by Dadebo and McAuley [3]. The system is described by

$$\frac{dx_1}{dt} = u \tag{11.10}$$

$$\frac{dx_2}{dt} = x_1^2 + u^2 \tag{11.11}$$

with the initial state $\mathbf{x}(0) = [\ 1\ \ 0]^T$, and final time $t_f = 1$. There are no constraints on the scalar control variable u, but there is the equality constraint to be satisfied at the final time

$$h_1 = x_1(t_f) - 1 = 0. \tag{11.12}$$

It is required to find the unconstrained control policy that minimizes the performance index

$$I = x_2(t_f). \tag{11.13}$$

The augmented performance index to be minimized is therefore chosen as

$$J = I + \theta(h_1 - s_1)^2. \tag{11.14}$$

To determine how closely the equality constraint is satisfied, we examine the absolute value of the deviation from the desired state at the final time $|x_1(t_f) - 1|$.

To solve this problem with IDP, we chose $P = 10$ stages, $N = 1$ grid point, $R = 5$ randomly chosen points, 30 passes with region restoration factor $\eta = 0.70$, and 5 iterations with region reduction factor $\gamma = 0.70$ in each pass. The initial value for control was zero, the initial region size was 1.0, and the initial value for the shifting

Table 11.1: Effect of the penalty function factor θ on convergence for Example 1

Penalty factor θ	Deviation from desired state $\lvert x_1(1) - 1\rvert$	Performance index $I = x_2(1)$	Shifting term s_1	Sensitivity factor $-2\theta s_1$
0.7	5.210×10^{-7}	0.924235	0.660319	−0.92445
0.8	2.585×10^{-7}	0.924235	0.577780	−0.92445
1.0	2.485×10^{-8}	0.924235	0.462224	−0.92445
3.0	5.404×10^{-8}	0.924235	0.154075	−0.92445
10	1.226×10^{-7}	0.924235	0.046223	−0.92446
30	3.077×10^{-8}	0.924235	0.015407	−0.92442
50	1.454×10^{-6}	0.924239	0.009243	−0.92430
80	1.317×10^{-6}	0.925144	0.005847	−0.93552
100	2.257×10^{-5}	0.925661	0.004719	−0.94380

term s_1 was zero. The goal is to choose a piecewise linear control policy to minimize the augmented performance index J.

With both $\theta = 1$ and $\theta = 50$, rapid convergence to the minimum value of $I = 0.92424$ resulted with the absolute difference from the desired state of less than 10^{-6} in 30 passes, as is shown in Figure 11.1. The computation time for 30 passes was 0.8 s on a Pentium3/500 digital computer. From the further results as given in Table 11.1, it is seen that the product $-2\theta s_1$ is constant and equal to $-I$ in the range $0.7 \leq \theta \leq 30$. In this range of θ, the minimum value of the performance index is $I = 0.924235$. For the penalty function factor greater than 30, the deviation from the desired state is larger and also the performance index is larger than the optimum value.

The control policy and the state trajectory are shown in Figure 11.2.

Relationship of the sensitivity factor to the adjoint variable

Let us now solve this optimal control problem analytically by using Pontryagin's maximum principle using the approach outlined by Lapidus and Luus [8]. The Hamiltonian for this system is

$$H = z_1 u + x_1^2 + u^2 \tag{11.15}$$

where the adjoint variable z_1 is defined by

$$\frac{dz_1}{dt} = -\frac{\partial H}{\partial x_1} = -2x_1. \tag{11.16}$$

At the minimum value of the performance index I, the Hamiltonian is at its minimum value obtained through the stationary condition

$$\frac{\partial H}{\partial u} = z_1 + 2u = 0, \tag{11.17}$$

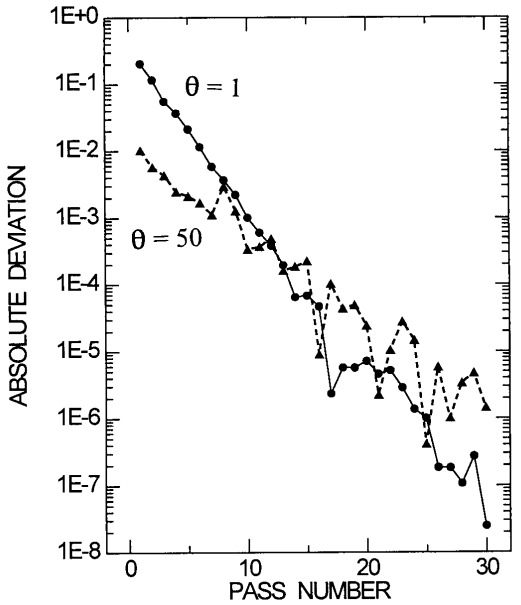

Figure 11.1: Effect of the penalty function factor θ on convergence for Example 1

so that the optimal control is given by

$$u = -0.5z_1. \tag{11.18}$$

By substituting Eq. (11.18) into Eq. (11.10) and differentiating with respect to t, and using Eq. (11.16), we get the second order differential equation

$$\frac{d^2x_1}{dt^2} - x_1 = 0. \tag{11.19}$$

Eq. (11.19) with the given boundary conditions ($x_1(0) = x_1(1) = 1$) is easily solved to give

$$x_1 = \frac{e^t + e^{(1-t)}}{1+e}. \tag{11.20}$$

Differentiating Eq. (11.20) with respect to t gives the optimal control from Eq. (11.10), namely,

$$u = \frac{e^t - e^{(1-t)}}{1+e}. \tag{11.21}$$

The minimum performance index is obtained then by integrating

$$I = \int_0^1 (x_1^2 + u^2)dt \tag{11.22}$$

giving

$$I = \frac{2(e-1)}{e+1} = 0.9242344. \tag{11.23}$$

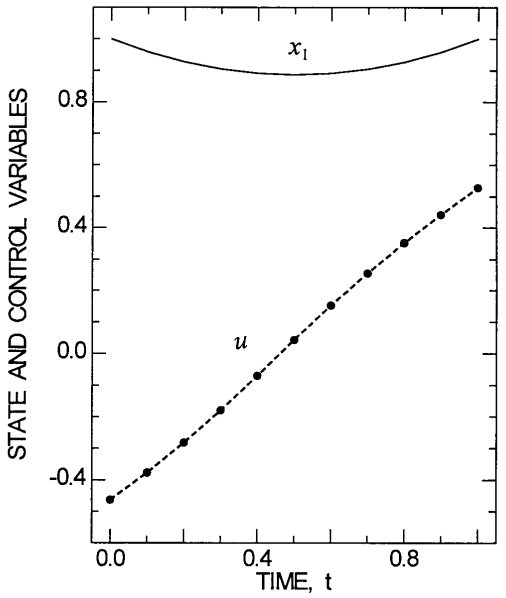

Figure 11.2: Optimal control policy with 10 piecewise linear sections and the corresponding state trajectory for Example 1, giving $I = 0.924235$

It is noted that the performance index $I = 0.924235$ in Table 11.1 is very close to this analytical result.

This example was also used by Luus [10] with $P = 20$ piecewise constant control policy and a quadratic penalty function without the shifting term and by Dadebo and McAuley [3] with the use of $P = 40$. Although the value $I = 0.92431$ obtained by the latter was better than 0.92518 reported by Goh and Teo [4], it comes short of $I = 0.9242345$ obtained here with $P = 10$ piecewise linear control sections. The smoothness of the control policy as seen in Figure 11.2 contributes to the excellent agreement with the analytical solution.

From Eq. (11.18) we see that

$$z_1 = \frac{2(e^{(1-t)} - e^t)}{1+e}, \tag{11.24}$$

so that at the final time $t = 1$,

$$z_1 = \frac{2(1-e)}{1+e} \tag{11.25}$$

which is negative of the performance index I. Therefore, as found computationally and reported in the fifth column of Table 11.1, the adjoint variable at the final time, in terms of the shifting term s_1, is equal to $-2\theta s_1$.

Example 2

Let us consider the first example used by Chen and Mills [1], which was also used by Luus and Storey [19]. The dynamics of the system are described by two state variables, but we introduce the third variable to put the performance index into the standard form for use of IDP:

$$\frac{dx_1}{dt} = x_2 + u_1 \tag{11.26}$$

$$\frac{dx_2}{dt} = -x_1 + u_2 \tag{11.27}$$

$$\frac{dx_3}{dt} = u_1^2 + u_2^2 \tag{11.28}$$

with $\mathbf{x}(0) = [\ -1\ \ 1\ \ 0\ ]^T$, and final time $t_f = \pi/2$. There are no constraints on the control variables, but there are two equality constraints to be satisfied at the final time

$$h_1 = x_1(t_f) - 1 \tag{11.29}$$

and

$$h_2 = x_2(t_f) + 1. \tag{11.30}$$

It is required to find the unconstrained control policy that minimizes the performance index

$$I = x_3(t_f). \tag{11.31}$$

The augmented performance index to be minimized is therefore chosen as

$$J = I + \theta[(h_1 - s_1)^2 + (h_2 - s_2)^2]. \tag{11.32}$$

To determine how closely the equality constraints are satisfied, we consider the sum of squares of deviations

$$Q = h_1^2 + h_2^2. \tag{11.33}$$

For optimization with IDP, we chose $P = 15$ stages, each of equal length and sought to find piecewise linear control policy for each of the control variables. We chose the initial control policy $\mathbf{u} = \mathbf{0}$, the initial region size $\mathbf{r}_{in} = \mathbf{2}$, the reduction factor $\gamma = 0.85$ and the region restoration factor $\eta = 0.90$, $R = 5$, and allowed 100 passes, each consisting of 30 iterations. We used a range of values for the penalty function factor θ. The computation time for 100 passes was 41 s on a PentiumII/350.

As is shown in Table 11.2, there is a wide range $1 \leq \theta \leq 500$ for which convergence to $I = 2.5464796$ occurs, the shifting parameter s_1 is very small, and the factor $-2\theta s_2$ is constant at 2.54648 which is equal to the optimum value of the performance index. As in Example 1, this is not a numerical coincidence, but can be proven, since this example can also be solved analytically, as we see shortly, to give the optimum value for the performance index $I = 8/\pi = 2.546479$. Therefore, the value obtained with 15 stages of piecewise linear control is excellent. The optimal control policy is given in Figure 11.3 and the state trajectories are given in Figure 11.4.

Table 11.2: Effect of the penalty function factor θ on convergence for Example 2

Penalty function factor θ	Sum of squares Q	Performance index I	Shifting terms s_1	s_2
0.1	7.85×10^{-13}	2.5464773	0.00015	-12.73239
1.0	1.99×10^{-15}	2.5464796	0.00000	-1.27324
10	3.01×10^{-15}	2.5464796	0.00000	-0.12732
100	2.32×10^{-16}	2.5464796	0.00000	-0.01273
500	3.49×10^{-16}	2.5464796	0.00000	-0.00255
1000	1.09×10^{-12}	2.5467489	0.00000	-0.00127

Relationship of the sensitivity factors to the adjoint variables at t_f

Let us now solve this optimal control problem analytically by using Pontryagin's maximum principle. The Hamiltonian for this system is

$$H = z_1(x_2 + u_1) + z_2(-x_1 + u_2) + u_1^2 + u_2^2 \tag{11.34}$$

where the adjoint variables z_1 and z_2 are defined by

$$\frac{dz_1}{dt} = -\frac{\partial H}{\partial x_1} = z_2 \tag{11.35}$$

and

$$\frac{dz_2}{dt} = -\frac{\partial H}{\partial x_2} = -z_1. \tag{11.36}$$

At the minimum value of the performance index I, the Hamiltonian is at its minimum value obtained through the stationary condition

$$\frac{\partial H}{\partial u_1} = z_1 + 2u_1 = 0 \tag{11.37}$$

and

$$\frac{\partial H}{\partial u_2} = z_2 + 2u_2 = 0, \tag{11.38}$$

yielding the optimal control policy

$$u_1 = -0.5z_1 \tag{11.39}$$

and

$$u_2 = -0.5z_2. \tag{11.40}$$

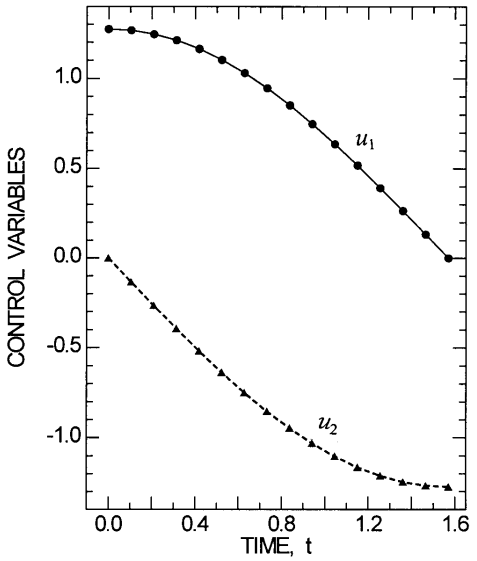

Figure 11.3: Optimal control policy with 15 piecewise linear sections for Example 2, giving $I = 2.5464796$

Therefore, the state equations become

$$\frac{dx_1}{dt} = x_2 - 0.5z_1 \tag{11.41}$$

and

$$\frac{dx_2}{dt} = -x_1 - 0.5z_2. \tag{11.42}$$

In order to remove the adjoint variables from these equations, we differentiate with respect to time t to yield the two equations:

$$\frac{d^2x_1}{dt^2} = x_1 + 2\frac{dx_2}{dt} \tag{11.43}$$

and

$$\frac{d^2x_2}{dt^2} = x_2 - 2\frac{dx_1}{dt}. \tag{11.44}$$

To uncouple these equations, we differentiate Eq. (11.43) twice with respect to t to yield

$$\frac{d^4x_1}{dt^4} = -2\frac{d^2x_1}{dt^2} - x_1 \tag{11.45}$$

which is readily solved to give

$$x_1 = (A_1 + A_2t)\sin t + (B_1 + B_2t)\cos t \tag{11.46}$$

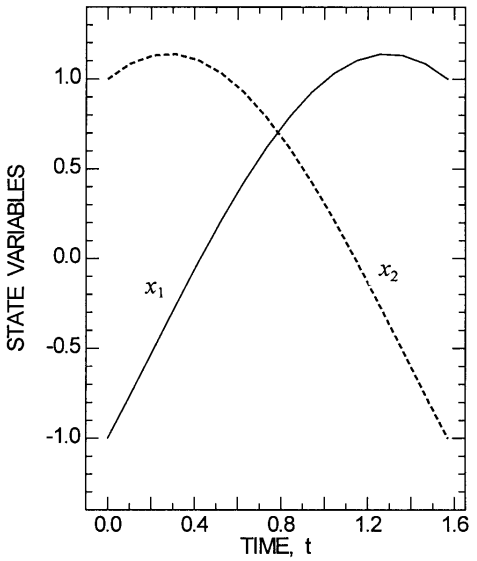

Figure 11.4: State trajectories for Example 2 resulting from the control policy in Figure 11.3

where A_1, A_2, B_1, and B_2 are constants to be determined from the boundary conditions, i.e., the given initial state and the desired state.

It immediately follows that, in terms of the same integration constants, we can write

$$x_2 = (A_1 + A_2 t)\cos t - (B_1 + B_2 t)\sin t. \tag{11.47}$$

So the four constants are readily determined: $A_1 = 1$, $A_2 = 0$, $B_1 = -1$, and $B_2 = 4/\pi$.

Thus the optimal trajectory is given by

$$x_1 = \sin t - (1 - \frac{4t}{\pi})\cos t \tag{11.48}$$

$$x_2 = \cos t - (1 - \frac{4t}{\pi})\sin t. \tag{11.49}$$

The optimal control policy is therefore

$$u_1 = \frac{4\cos t}{\pi} \tag{11.50}$$

$$u_2 = -\frac{4\sin t}{\pi}. \tag{11.51}$$

Thus the minimum value of the performance index is

$$I = \int_0^{\pi/2} (u_1^2 + u_2^2)dt = \frac{8}{\pi} = 2.5464791. \tag{11.52}$$

The adjoint variables at the optimum conditions are

$$z_1 = -\frac{8\cos t}{\pi} \tag{11.53}$$

$$z_2 = \frac{8\sin t}{\pi}, \tag{11.54}$$

so that at the final time $t_f = \pi/2$, $z_1 = 0$, and $z_2 = 8/\pi = I$. Numerically we were able to get these results from the shifting terms.

Example 3: Fed-batch reactor

Let us consider the batch reactor of Park and Ramirez [23] that we considered in Chapter 9. When the batch time is allowed to be increased, then the yield, as measured by the performance index, increases quite rapidly. Also what increases is the volume at the final time, as is shown in Figure 11.5. Suppose we have an upper constraint on the volume of 14.35 L, as postulated by Luus and Rosen [18]. This volume constraint is reached with a batch time of approximately $t_f = 15.3$ h giving a yield of 35.8. Let us suppose that we are able to allow a batch time of 20 h. It is then clear that the performance index can be improved substantially by taking the volume constraint directly into consideration and solving the state constrained optimal control problem. In the presence of the volume constraint and the final time t_f of 20 h, we therefore have the following optimal control problem:

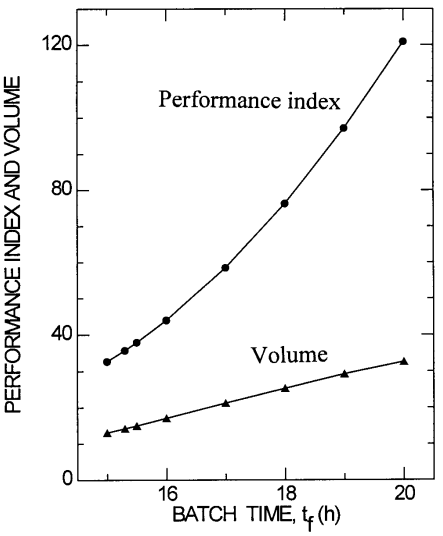

Figure 11.5: Increase of the performance index and volume as the batch time is increased

$$\frac{dx_1}{dt} = g_1(x_2 - x_1) - \frac{u}{x_5}x_1 \tag{11.55}$$

$$\frac{dx_2}{dt} = g_2 x_3 - \frac{u}{x_5}x_2 \tag{11.56}$$

$$\frac{dx_3}{dt} = g_3 x_3 - \frac{u}{x_5}x_3 \tag{11.57}$$

$$\frac{dx_4}{dt} = -7.3 g_3 x_3 + \frac{u}{x_5}(20 - x_4) \tag{11.58}$$

$$\frac{dx_5}{dt} = u \tag{11.59}$$

where

$$g_3 = \frac{21.87 x_4}{(x_4 + 0.4)(x_4 + 62.5)} \tag{11.60}$$

$$g_2 = \frac{x_4 e^{-5x_4}}{0.1 + x_4} \tag{11.61}$$

$$g_1 = \frac{4.75 g_3}{0.12 + g_3} \tag{11.62}$$

with the initial condition

$$\mathbf{x}(0) = [\ 0\ \ 0\ \ 1\ \ 5\ \ 1\ ]^T. \tag{11.63}$$

The feed rate to the reactor u is bounded by

$$0 \le u \le 2. \tag{11.64}$$

The performance index to be maximized is the amount of secreted SUC2-s2 produced at the final time t_f

$$I = x_1(t_f) x_5(t_f) \tag{11.65}$$

where the final time t_f is specified as 20 h. In addition, we also have the volume constraint

$$x_5(t_f) - 14.35 = 0, \tag{11.66}$$

where we have used the equality, rather than inequality, since it is clear from Figure 11.5 that the maximum yield occurs at the maximum volume.

In using IDP with flexible stage lengths, we then have two equality constraints to consider. This presents no problem, since we simply deal with two shifting terms in the augmented performance index to be maximized

$$J = I - \theta[(t_{fc} - t_f - s_1)^2 + (x_5(t_{fc}) - 14.35 - s_2)^2] \tag{11.67}$$

where t_{fc} is the calculated final time obtained by adding up the time stages, and update them after every pass as follows:

$$s_1^{q+1} = s_1^q - (t_{fc} - t_f) \tag{11.68}$$

Table 11.3: Effect of the penalty function factor θ on convergence for Example 3

Penalty function factor θ	Performance index I	Calculated final time t_{fc}	Calculated final volume $x_5(t_{fc})$	Shifting terms in J s_1	s_2
0.4	78.611	20.0000	14.3500	-9.7545	-4.7113
0.5	78.662	20.0000	14.3500	-7.7980	-3.7742
1	78.656	20.0000	14.3500	-3.8996	-1.8867
3	78.647	20.0000	14.3500	-1.3001	-0.6288
10	78.652	20.0000	14.3500	-0.3900	-0.1886
30	78.662	20.0000	14.3500	-0.1300	-0.0630
100	78.647	20.0000	14.3500	-0.0390	-0.0189

$$s_2^{q+1} = s^q - (x_5(t_{fc}) - 14.35). \tag{11.69}$$

We have used the same penalty function factor θ for both of the equality constraints to keep everything as simple as possible.

We chose $P = 30$ stages, each taken initially of length 2/3 h, $N = 1$ grid point, $R = 5$, $\gamma = 0.95$, $\eta = 0.85$, and allowed 50 passes, each consisting of 30 iterations. As is seen in Table 11.3, the penalty function factor θ can be chosen between 0.5 and 30 to get reliable convergence after 50 passes from the initial control policy of $u = 0$ with initial region size $r = 2$, and initial region size for the stage lengths of 0.1. By using as initial condition the control policy and shifting terms as obtained in the first run with $\theta = 0.5$, a run was performed to refine the control policy, giving the optimal control policy in Figure 11.6 with $I = 78.6625$. There is only a slight improvement over $I = 78.60$, obtained by Luus and Rosen [18] with the use of 20 stages of equal length.

Example 4

Let us now take an optimal control problem where four final states are specified. We consider the third example used by Chen and Mills [1] and by Luus and Storey [19]. The system is described by

$$\frac{dx_1}{dt} = x_2 + \cos 2t + tu_1 + \exp(t^2)u_2 + t^2u_3 \tag{11.70}$$

$$\frac{dx_2}{dt} = -x_1 + \sin 5t + t^2u_2 \tag{11.71}$$

$$\frac{dx_3}{dt} = x_3 + e^t + 2tu_1 + tu_3 \tag{11.72}$$

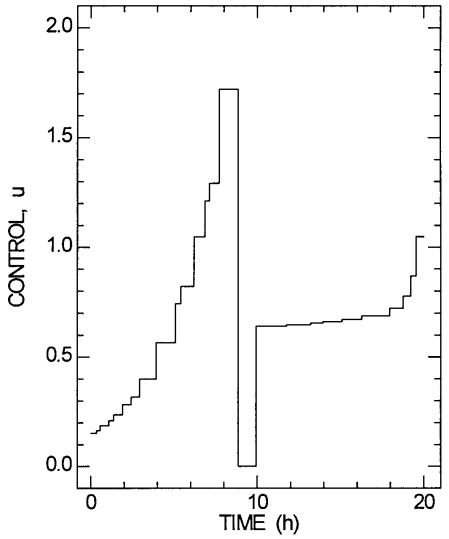

Figure 11.6: Optimal control policy for the fed-batch reactor with $t_f = 20$ when the volume is constrained by $V \leq 14.35L$, giving $I = 78.6625$

$$\frac{dx_4}{dt} = 2x_4 + \sin 10t + t^2 u_1 + \cos 3t u_2 + 4t u_3 \tag{11.73}$$

$$\begin{aligned} \frac{dx_5}{dt} &= [t^4 x_1 + \sin t x_2 + (4t_2 - 1)x_3 + 3e^{-t} x_4 - \cos 4t]^2 \\ &\quad + (3+t)u_1^2 + u_1 u_2 + 2u_1 u_3 + (4+t^2)u_2^2 + (3+t^3)u_3^2 \end{aligned} \tag{11.74}$$

with the initial state

$$\mathbf{x}(0) = [\ 1 \quad -1 \quad 0 \quad -2 \quad 0\ ]^T. \tag{11.75}$$

The desired state at the final time $t_f = 2$ is specified as

$$\mathbf{x}^d = [\ -1 \quad 0.5 \quad -2 \quad 1 \quad free\ ]^T. \tag{11.76}$$

The fifth variable has been added to put the problem into the standard form; so, the performance index to be minimized is

$$I = x_5(t_f). \tag{11.77}$$

We introduce the augmented performance index with a quadratic penalty function containing four shifting terms

$$\begin{aligned} J &= I + \theta[(x_1(t_f) + 1 - s_1)^2 + (x_2(t_f) - 0.5 - s_2)^2 \\ &\quad + (x_3(t_f) + 2 - s_3)^2 + (x_4(t_f) - 1 - s_4)^2]. \end{aligned} \tag{11.78}$$

Table 11.4: Effect of the penalty function factor θ on convergence for Example 4

Penalty function factor θ	Sum of squares Q	Performance index I	Shifting terms in J s_1	s_2	s_3	s_4
0.2	2.132×10^{-7}	118.62	-32.2118	30.9303	-50.7931	0.1217
0.3	3.560×10^{-10}	118.63	-21.4789	20.6224	-33.8669	0.0810
0.5	1.565×10^{-11}	118.63	-12.8874	12.3735	-20.3202	0.0486
1.0	4.338×10^{-11}	118.63	-6.4437	6.1867	-10.1601	0.0243
5.0	1.187×10^{-9}	118.63	-1.2887	1.2371	-2.0320	0.0049
10.0	7.264×10^{-10}	118.70	-0.6446	0.6222	-1.0182	0.0029

Again, for simplicity, we use the same penalty function factor θ for all the equality constraints.

The optimal control problem is then to find the control policy $\mathbf{u}$ in the time interval $0 \leq t < 2$ that minimizes the augmented performance index J. To solve this optimal control problem, Luus and Storey [19] used initially 50 stages of constant length and piecewise constant control, and then showed that the use of 40 stages of piecewise linear continuous control yielded a lower value of the performance index, namely $I = 114.81$, as opposed to $I = 121.20$. Here we use $P = 30$ stages of piecewise linear control with $N = 1$ and provide detailed computational results.

To minimize the augmented performance index we chose $\theta = 1$, initial control policy $\mathbf{u}^{(0)} = \mathbf{0}$, 30 iterations per pass with reduction factor $\gamma = 0.90$, 200 passes with the region restoration factor $\eta = 0.95$. The shifting terms were initially put to zero and updated after every pass according to Eq. (11.9). To measure the rate of convergence we examine the sum of squares of deviations from the desired states

$$Q = (x_1(t_f) + 1)^2 + (x_2(t_f) - 0.5)^2 + (x_3(t_f) + 2)^2 + (x_4(t_f) - 1)^2. \tag{11.79}$$

As is seen in Figure 11.7, it makes very little difference whether $R = 3$ or $R = 21$ randomly chosen values for control are chosen. With $R = 3$ we obtained $Q = 3.73 \times 10^{-11}$, whereas with $R = 21$, we obtained $Q = 9.29 \times 10^{-12}$ after 200 passes, both giving the minimum value of $I = 118.63$.

The effect of the penalty function factor θ is shown in Table 11.4 that was obtained by using $R = 5$ with the parameters used earlier. In the range $0.3 \leq \theta \leq 5$, we get reliable values for the minimum value of the performance index and have the final state constraints satisfied with $Q < 1.2 \times 10^{-9}$.

The optimal control policy is given in Figure 11.8. The control profiles u_1 and u_3 are smooth for the entire time interval. It is interesting to note, however, the highly oscillatory nature of the control variable u_2 after $t = 1.5$. The same type of behavior was observed by Luus and Storey [19] with the use of 40 stages.

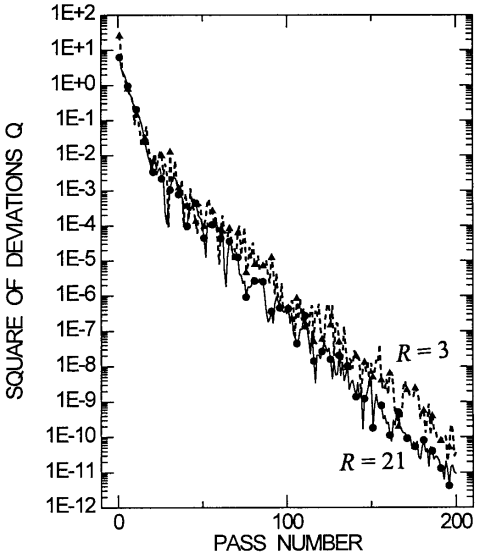

Figure 11.7: Sum of squares of deviations from the desired states as a function of the number of passes, using piecewise linear control with $P = 30$ time stages of equal length

The corresponding state trajectories are shown in Figure 11.9 where a very rapid change in the state variable x_1 is noted near the final time.

Example 5: Two-stage CSTR system

Let us now consider the two-stage CSTR system we used in Section 10.4, but changing it very slightly with respect to the performance index. Instead of choosing the performance index as the sum of squares of the state variables at the final time, we introduce another variable x_5 to reflect the sum of squares of deviations from the desired state and the control effort, and impose the final state constraints at the final time $t_f = 0.325$. The system is described by

$$\frac{dx_1}{dt} = -3x_1 + g_1(\mathbf{x}) \tag{11.80}$$

$$\frac{dx_2}{dt} = -11.1558x_2 + g_1(\mathbf{x}) - 8.1558(x_2 + 0.1592)u_1 \tag{11.81}$$

$$\frac{dx_3}{dt} = 1.5(0.5x_1 - x_3) + g_2(\mathbf{x}) \tag{11.82}$$

$$\frac{dx_4}{dt} = 0.75x_2 - 4.9385x_4 + g_2(\mathbf{x}) - 3.4385(x_4 + 0.122)u_2 \tag{11.83}$$

$$\frac{dx_5}{dt} = x_1^2 + x_2^2 + x_3^2 + x_4^2 + 0.1(u_1^2 + u_2^2) \tag{11.84}$$

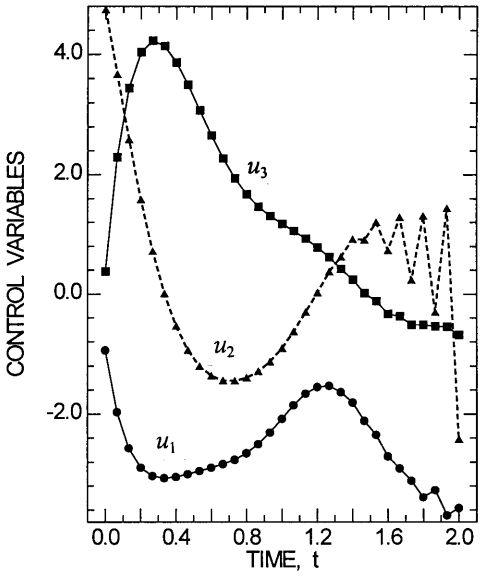

Figure 11.8: Optimal control policy using $P = 30$ piecewise linear sections for Example 4, yielding $I = 118.63$ and $Q = 9.3 \times 10^{-12}$

where the reaction term in the first tank is given by

$$\begin{aligned} g_1(\mathbf{x}) &= 1.5 \times 10^7 (0.5251 - x_1) \exp\left(\frac{-10}{x_2 + 0.6932}\right) \\ &\quad -1.5 \times 10^{10} (0.4748 + x_1) \exp\left(\frac{-15}{x_2 + 0.6932}\right) - 1.4280 \end{aligned} \tag{11.85}$$

and the reaction term in the second tank is

$$\begin{aligned} g_2(\mathbf{x}) &= 1.5 \times 10^7 (0.4236 - x_2) \exp\left(\frac{-10}{x_4 + 0.6560}\right) \\ &\quad -1.5 \times 10^{10} (0.5764 + x_3) \exp\left(\frac{-15}{x_4 + 0.6560}\right) - 0.5086. \end{aligned} \tag{11.86}$$

The initial condition is given by

$$\mathbf{x}(0) = [\ 0.1962 \quad -0.0372 \quad 0.0946 \quad 0\ ]^T \tag{11.87}$$

and the constraints on the control are

$$-1 \leq u_j \leq 1, \quad j = 1, 2. \tag{11.88}$$

The state variables x_1 and x_3 are normalized concentration variables in tanks 1 and 2 respectively, and x_2 and x_4 are normalized temperature variables in tanks 1 and 2

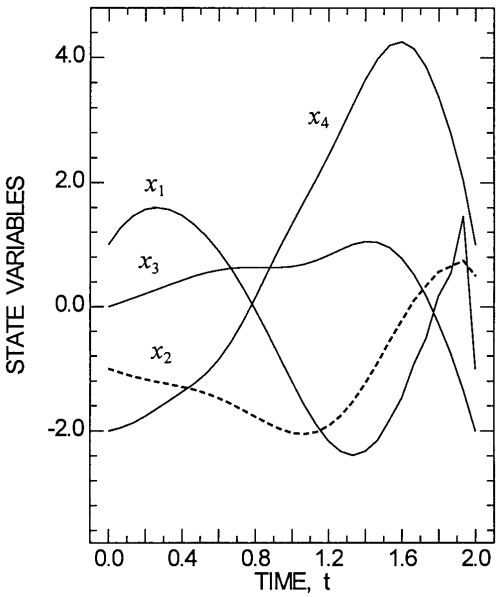

Figure 11.9: State trajectories for Example 4 resulting from the use of the optimal control policy shown in Figure 11.8

respectively. It is required to find the control policy that minimizes the performance index

$$I = x_5(t_f) \tag{11.89}$$

where the dimensionless final time $t_f = 0.325$. We impose the final state constraint

$$\mathbf{x}(t_f) = \mathbf{0}. \tag{11.90}$$

Therefore, we introduce the augmented performance index

$$J = I + \theta \sum_{i=1}^{4} (x_i(t_f) - s_i)^2 \tag{11.91}$$

and use IDP to minimize this augmented performance for several values of the penalty function factor θ.

We used the same parameters as were used by Luus *et al.* [17] for this problem. We chose $P = 20$ stages of equal length, $N = 1$ grid point, 30 iterations per pass with region contraction factor $\gamma = 0.70$, and allowed 250 passes with region restoration factor $\eta = 0.95$. The initial region sizes were chosen as $r_1^{(0)} = r_2^{(0)} = 1$, and the initial control was chosen zero. To measure the closeness of meeting the final state constraints, we used the norm

$$\|\mathbf{x}(t_f)\| = (\sum_{i=1}^{4} x_i^2(t_f))^{\frac{1}{2}}. \tag{11.92}$$

Table 11.5: Effect of the penalty function factor θ on convergence for Example 5

Penalty function factor θ	Final state norm $\|\mathbf{x}(t_f)\|$	Performance index I	Two shifting terms in J	
			s_1	s_3
20	3.072×10^{-4}	0.044334	1.242×10^{-1}	-1.105×10^{-1}
50	1.224×10^{-4}	0.045608	6.652×10^{-2}	-5.401×10^{-2}
2×10^2	3.028×10^{-5}	0.046478	2.348×10^{-2}	-1.767×10^{-2}
5×10^2	9.568×10^{-6}	0.046737	1.102×10^{-2}	-8.082×10^{-3}
1×10^3	1.299×10^{-6}	0.046853	5.760×10^{-3}	-4.181×10^{-3}
1.5×10^3	1.158×10^{-7}	0.046870	3.861×10^{-3}	-2.799×10^{-3}
2×10^3	8.110×10^{-9}	0.046872	2.898×10^{-3}	-2.101×10^{-3}
5×10^3	2.386×10^{-10}	0.046872	1.159×10^{-3}	-8.403×10^{-4}
1×10^4	3.643×10^{-10}	0.046872	5.797×10^{-4}	-4.201×10^{-4}
1.5×10^4	6.515×10^{-7}	0.046863	4.409×10^{-4}	-3.198×10^{-4}
2×10^4	1.963×10^{-6}	0.046853	2.818×10^{-4}	-2.052×10^{-4}
5×10^4	4.505×10^{-7}	0.046967	8.027×10^{-5}	-5.902×10^{-5}
1×10^5	2.228×10^{-7}	0.047169	3.481×10^{-5}	-2.559×10^{-5}
5×10^5	8.078×10^{-8}	0.047651	5.890×10^{-6}	-4.076×10^{-6}
1×10^6	2.345×10^{-8}	0.047830	1.205×10^{-6}	-9.998×10^{-7}
5×10^6	3.244×10^{-9}	0.048259	-1.969×10^{-8}	-8.700×10^{-8}

Instead of Runge-Kutta method we used DVERK [6] with local error tolerance of 10^{-6}. Figures 11.10 and 11.11 show that $R = 7$ does not lead to convergence and that $R = 10$ is a good number of randomly chosen points to use.

In Table 11.5 we see that there is a range for the penalty function factor $2000 \leq \theta \leq 10000$ for which convergence to the minimum value of the performance index $I = 0.046872$ occurs so that all the final states are within 10^{-8} of the origin. With $\theta = 5000$ the shifting terms are $s_1 = 1.159 \times 10^{-3}$, $s_2 = -2.440 \times 10^{-5}$, $s_3 = -8.403 \times 10^{-4}$, and $s_4 = -3.450 \times 10^{-4}$. The shifting terms s_1 and s_3 are given in Table 11.5, to show that in the acceptable range for θ, the product $-2\theta s_i$ is constant, but outside the range where the optimum is not obtained, this is not necessarily true. When θ is too small, the final state constraint is not adequately satisfied, and if θ is too large then the performance index is greater than the minimum value. This is shown graphically in Figure 11.12.

These five examples have shown the viability of a quadratic penalty function with shifting terms to provide the optimal control policy, and also how the shifting terms provide sensitivity information with respect to the final state constraints. We now turn to another type of penalty function that has been found useful in IDP to deal with final state equality constraints.

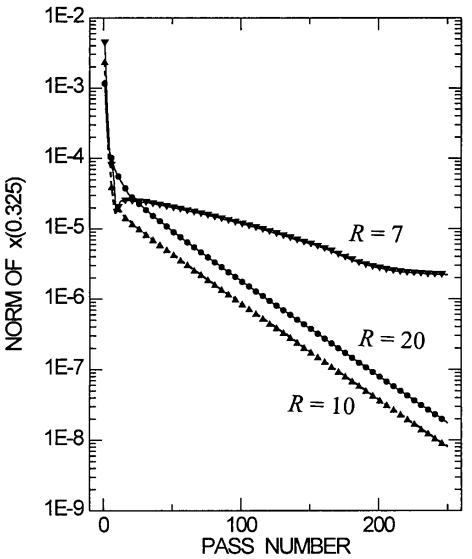

Figure 11.10: Convergence profile for Example 5, showing $\|\mathbf{x}(t_f)\|$ as a function of pass number with penalty function factor $\theta = 2000$

11.2.3 Absolute value penalty function

Although for steady state optimization difficult equality constraints are handled very well by the use of quadratic penalty function with shifting terms [13], Luus and Wyrwicz [20] showed that absolute penalty functions can also be used effectively to get accurate optimum results for high-dimensional systems in the presence of difficult equality constraints.

Dadebo and McAuley [3] reported successful results in using in IDP an absolute value penalty function. Therefore, in this section,we consider the augmented performance index in the form

$$J = I + \sum_{i=1}^{k} \theta_i |x_i(t_f) - x_i^d|. \tag{11.93}$$

There are k positive weighting factors θ_i, which must be provided by the user, or must be determined during the course of the calculations. One effective method of getting good values for these weighting factors during the course of the calculations is to use the suggestion of Mekarapiruk and Luus [22] where the weighting factors are adjusted after every pass of IDP, being increased in magnitude if the tolerance is not met, and decreased in value if the final state constraint is satisfied within the given tolerance. Thus, after starting with some reasonable values for each θ_i, these penalty function factors are changed after every pass by

$$\theta_i^{q+1} = (1+\delta)\theta_i^q \quad \text{if} \quad |x_i(t_f) - x_i^d| > \epsilon_i \tag{11.94}$$

$$\theta_i^{q+1} = (1-\delta)\theta_i^q \quad \text{if} \quad |x_i(t_f) - x_i^d| \leq \epsilon_i \tag{11.95}$$

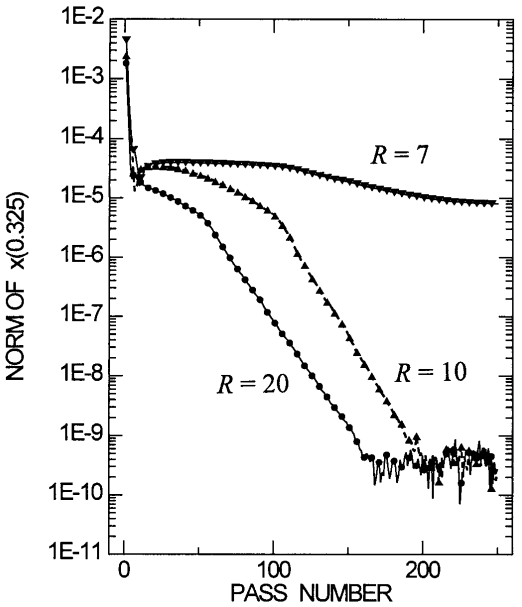

Figure 11.11: Convergence profile for Example 5, showing $\|\mathbf{x}(t_f)\|$ as a function of pass number with penalty function factor $\theta = 5000$

where δ is a small positive factor such as 0.05, q is the pass number, and ϵ_i is the tolerance for the constraint i. To reduce the size of the fluctuations in the control policy, δ is decreased by a small amount such as 1% after every pass. We illustrate this approach with an example where there are six final state constraints.

Example 6: Crane problem

Let us consider the problem of transferring a container from a ship to a cargo truck by a crane so that the container does not sway too much and it comes to rest on the truck in the specified time. The hoist and a trolley motor provide two control variables to achieve the goal. This problem was first presented by Sakawa and Shindo [25], and then considered for optimal control studies by Goh and Teo [4], Luus [10], Teo *et al.* [26], Dadebo and McAuley [3], and by Mekarapiruk and Luus [22]. There are six state variables; so, with the addition of the performance index integrand, we have seven differential equations:

$$\frac{dx_1}{dt} = 9x_4 \tag{11.96}$$

$$\frac{dx_2}{dt} = 9x_5 \tag{11.97}$$

$$\frac{dx_3}{dt} = 9x_6 \tag{11.98}$$

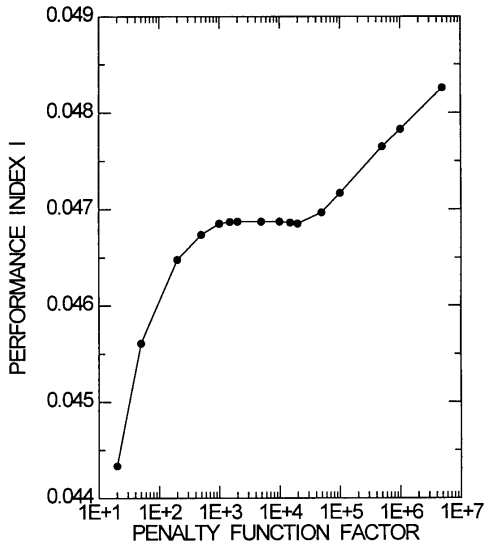

Figure 11.12: Effect of the penalty function factor on the converged value of the performance index I for Example 5

$$\frac{dx_4}{dt} = 9(u_1 + 17.2656x_3) \tag{11.99}$$

$$\frac{dx_5}{dt} = 9u_2 \tag{11.100}$$

$$\frac{dx_6}{dt} = -\frac{9(u_1 + 27.0756x_3 + 2x_5x_6)}{x_2} \tag{11.101}$$

$$\frac{dx_7}{dt} = 4.5(x_3^2 + x_6^2) \tag{11.102}$$

where x_1 is the horizontal distance, and x_2 is the vertical distance the container moves, x_3 is the swing angle of the container, x_4 is the horizontal velocity, x_5 is the vertical velocity and x_6 is the angular velocity of the container. The variable x_7 gives the behaviour of the container during transfer, and is introduced to provide the performance index to be minimized.

The initial state of the container is

$$\mathbf{x}(0) = [\ 0 \quad 22 \quad 0 \quad 0 \quad -1 \quad 0 \quad 0\ ]^T \tag{11.103}$$

and the desired state at the dimensionless time $t_f = 1$ is

$$\mathbf{x}^d = [\ 10 \quad 14 \quad 0 \quad 2.5 \quad 0 \quad 0 \quad \text{free}\ ]^{\mathrm{T}}. \tag{11.104}$$

The two control variables consisting of the torque of the hoist and trolley motors are bounded by

$$-2.834 \leq u_1 \leq 2.834 \tag{11.105}$$

and

$$-0.809 \leq u_2 \leq 0.713. \tag{11.106}$$

In addition, we have the state constraints on x_4 and x_5:

$$-2.5 \leq x_4 \leq 2.5 \tag{11.107}$$

$$-1 \leq x_5 \leq 1. \tag{11.108}$$

The optimal control problem is to find u_1 and u_2 in the time interval $0 \leq t < 1$, so that the performance index

$$I = x_7(t_f) \tag{11.109}$$

is minimized, when $t_f = 1$.

Here we choose the augmented performance index to be minimized, as suggested by Luus[10],

$$J = I + \sum_{i=1}^{6} \theta_i |x_i(t_f) - x_i^d| + \rho \sum_{j=1}^{P} (p_{4j} + p_{5j}) \tag{11.110}$$

where

$$p_{4j} = \begin{cases} -2.5 - x_4(t_j) & \text{if } x_4(t_j) < -2.5 \\ 0 & \text{if } -2.5 \leq x_4(t_j) \leq 2.5 \\ x_4(t_j) - 2.5 & \text{if } x_4(t_j) > 2.5 \end{cases} \tag{11.111}$$

and

$$p_{5j} = \begin{cases} -1 - x_5(t_j) & \text{if } x_5(t_j) < -1 \\ 0 & \text{if } -1 \leq x_5(t_j) \leq 1 \\ x_5(t_j) - 1 & \text{if } x_5(t_j) > 1. \end{cases} \tag{11.112}$$

In the augmented performance index we set $\rho = 1$, which is sufficiently large to ensure that the state inequality constraints are satisfied, and concentrate on the determination of the six penalty function factors $\theta_1, \theta_2, \cdots, \theta_6$.

For the starting point for the first run we chose $\theta_i = 10^{-2}, \epsilon_i = 10^{-3}, i = 1, 2, \cdots, 6$, $\delta = 0.05$ and chose $P = 10$ time stages, each of length 0.1. For integration we chose the Runge-Kutta method with an integration step size of 0.01. We took initial values for the control variables $u_1^{(0)} = 0.5$ and $u_2^{(0)} = 0$ and initial region sizes for the control variables as $r_1^{(0)} = 1.3$ and $r_2^{(0)} = 0.3$. For IDP we chose $N = 3$ grid points, $R = 25$, $\gamma = 0.70$, $\eta = 0.95$, and allowed 150 passes, each consisting of 30 iterations. After every pass δ was decreased by 1%, and the penalty function factors were updated. The computation time for 150 passes was 617 s on a PentiumII/350. The convergence profile as measured by the sum of absolute deviations of the final states from their desired values is shown in Figure 11.13. The value of the performance index is $I = 0.005370$. The final state is

$$\begin{aligned} \mathbf{x}(t_f) \;=\; [\; & 10.001479 \quad 14.000000 \quad -0.000931 \quad 2.500000 \\ & 0.000000 \quad -0.001116 \quad 0.005370 \;]^T \end{aligned} \tag{11.113}$$

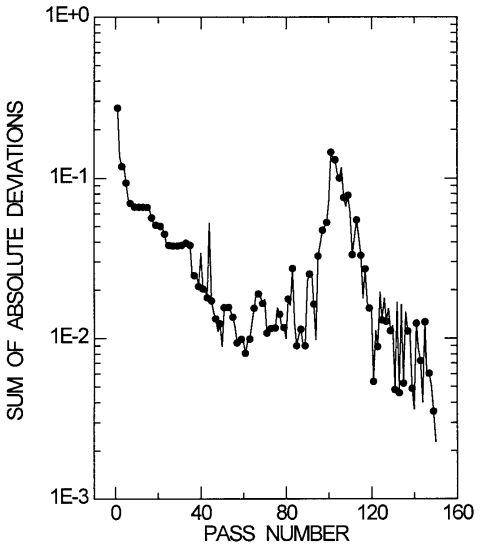

Figure 11.13: Convergence profile for the first run for Example 6, showing the sum of the absolute deviations of the final states from the desired values

giving a sum of absolute deviation from the desired values of 2.272×10^{-3}. The resulting weighting factors are

$$\begin{aligned} \theta \;\; = \;\; [\; & 5.860 \times 10^{-4} \quad 1.940 \times 10^{-4} \quad 6.321 \times 10^{-2} \quad 6.685 \times 10^{-3} \\ & 4.823 \times 10^{-4} \quad 7.431 \times 10^{-2} \;]^T . \end{aligned} \tag{11.114}$$

For the second run we reduced the size of the initial region sizes for the control to $r_1^{(0)} = r_2^{(0)} = 0.1$, and reduced ϵ_i to 10^{-6}. The control policy obtained in the first run was used as the initial control policy and the initial values for the weighting factors were the ones obtained in the first run. The parameters for IDP were kept the same. As expected, a lower value of the sum of deviations was obtained, namely, 1.4×10^{-5} with $I = 0.005527$.

For the final run we chose as initial region sizes $r_1^{(0)} = r_2^{(0)} = 0.001$, which yielded a sum of deviations of less than 10^{-6} in 38 passes, as is shown in Figure 11.14. The resulting control policy is given in Figure 11.15 and the state trajectories are given in Figure 11.16. The final state is

$$\begin{aligned} \mathbf{x}(t_f) \;\; = \;\; [\; & 10.000000 \quad 14.000000 \quad -0.000001 \quad 2.500000 \\ & 0.000000 \quad 0.000000 \quad 0.005526 \;]^T \end{aligned} \tag{11.115}$$

giving a sum of absolute deviations from the desired values of 2.272×10^{-3}. The resulting weighting factors are

$$\begin{aligned} \theta \;\; = \;\; [\; & 7.012 \times 10^{-4} \quad 5.030 \times 10^{-5} \quad 5.521 \times 10^{-2} \quad 6.662 \times 10^{-3} \\ & 3.278 \times 10^{-4} \quad 7.255 \times 10^{-2} \;]^T . \end{aligned} \tag{11.116}$$

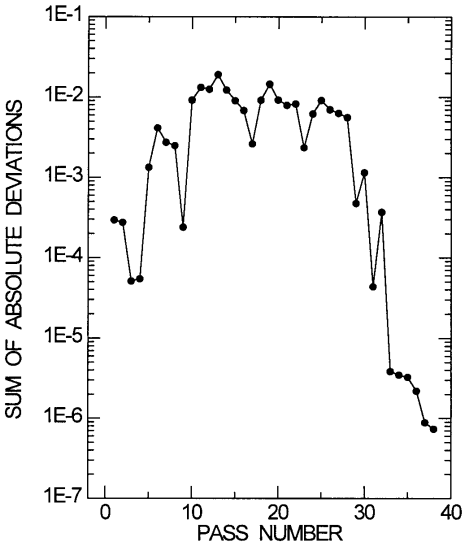

Figure 11.14: Convergence to sum of deviations to 7.37×10^{-7} giving $I = 0.005526$ for Example 6

These values of the weighting factors differ somewhat from those obtained by Mekarapiruk and Luus [22], showing that the weighting factors are not unique, and are not related directly to the sensitivities, as the shifting terms are in the quadratic penalty function.

As is shown by Mekarapiruk and Luus [22], the performance index can be reduced slightly to $I = 0.005264$ by increasing the number of time stages to 20.

11.2.4 Remarks on the choice of penalty functions

Luus *et al.* [17] found no clear computational advantage of the absolute penalty function over the quadratic penalty function with shifting terms in solving the two-stage CSTR problem considered earlier as Example 5. Therefore, more research in this area is needed before choosing one over the other. The use of the quadratic penalty function with shifting terms has the advantage of yielding sensitivity factors that are related to the adjoint variables when Pontryagin's maximum principle is used. This information is useful and it also provides an indication whether convergence to the optimum has been achieved. It is clear that there must be an upper limit to the number of final state equality constraints that can be considered conveniently in this fashion with the use of penalty functions. Fortunately many real problems involve only a small number of final state constraints; so, the methods used here provide reliable means for dealing with such constraints.

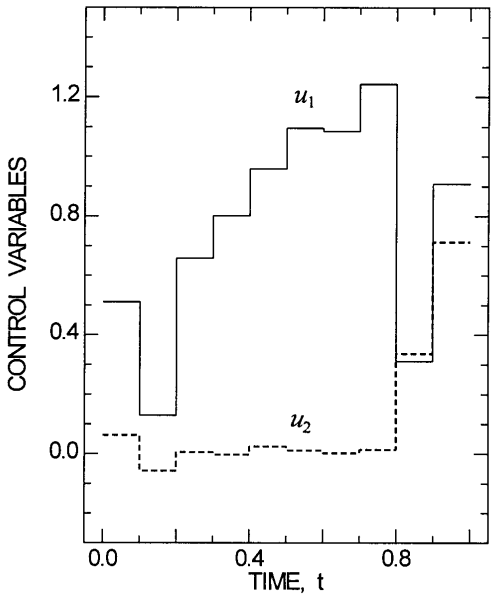

Figure 11.15: Optimal control policy with $P = 10$ stages for Example 6

11.3 State inequality constraints

11.3.1 Problem formulation

Let us consider the system described by the differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{11.117}$$

where the initial state $\mathbf{x}(0)$ is given. The state vector is $\mathbf{x}$ is an $(n \times 1)$ vector and $\mathbf{u}$ is an $(m \times 1)$ control vector bounded by

$$\alpha_j \leq u_j \leq \beta_j, \quad j = 1, 2, \cdots, m. \tag{11.118}$$

Inside the given time interval there are k state constraints of the form

$$\psi_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \cdots, k. \tag{11.119}$$

The optimal control problem is to find the control policy $\mathbf{u}(t)$ in the time interval $0 \leq t < t_f$ that minimizes the performance index

$$I = \Phi(\mathbf{x}(t_f)) \tag{11.120}$$

where the final time t_f is specified.

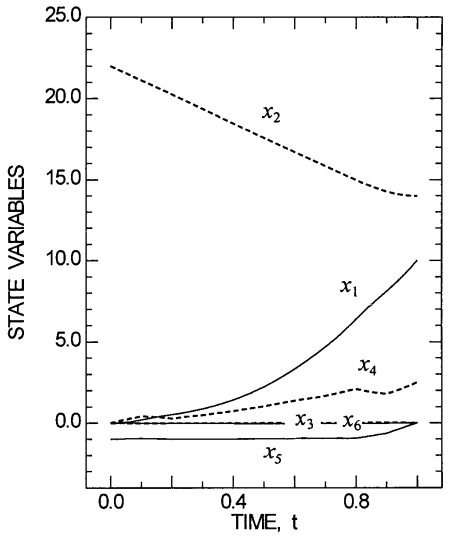

Figure 11.16: Optimal state trajectories for Example 6 resulting from the use of the control policy in Figure 11.15

11.3.2 State constraint variables

To deal with inequality constraints on the state, we follow the approach of Luus [10], which was also used in Example 6 in the previous section, with a small variation. Instead of difference equations, we use differential equations as suggested by Mekarapiruk and Luus [21] to construct the penalty functions. We therefore introduce k *state constraint variables* through the differential equations

$$\frac{dx_{n+i}}{dt} = \begin{cases} 0 & \text{if } \psi_i(\mathbf{x}) \leq 0 \\ \psi_i(\mathbf{x}) & \text{if } \psi_i(\mathbf{x}) > 0 \end{cases} \tag{11.121}$$

for $i = 1, 2, \cdots, k$, with the initial condition

$$x_{n+i}(0) = 0, \qquad i = 1, 2, \cdots, k. \tag{11.122}$$

At the final time t_f, $x_{n+i}(t_f)$ therefore gives the total violation of the i^{th} inequality constraint integrated over time. The advantage of taking such value is that sometimes a violation may occur for a short time inside a time stage, and such a violation may go by unnoticed if the difference equation approach is used where the violations are checked only at the ends of the time stages. We choose the augmented performance index to be minimized as

$$J = I + \sum_{i=1}^{k} \theta_i x_{n+i}(t_f) \tag{11.123}$$

where $\theta_i > 0$ are penalty function factors for the inequality state constraints.

Let us consider some illustrative examples.

Example 7: Chemical reactor control

Let us consider the optimal control of a chemical reactor involving a first-order chemical reaction as studied by Ko and Stevens [7], Reddy and Husain [24], Luus [10,12], and by Mekarapiruk and Luus [21]. The system is described by two differential equations

$$\frac{dx_1}{dt} = (1 - x_1)k_1 - x_1 k_2 \tag{11.124}$$

$$\frac{dx_2}{dt} = 300[(1 - x_1)k_1 - x_1 k_2] - u(x_2 - 290) \tag{11.125}$$

with the initial condition

$$\mathbf{x}^T(0) = [\ 0\ \ 380\ ]. \tag{11.126}$$

The state variable x_1 is the concentration of the desired component, and x_2 denotes the absolute temperature. The control u is a normalized coolant flow rate bounded by

$$0 \leq u \leq 0.5. \tag{11.127}$$

The reaction rate constants are dependent on the temperature x_2 in the following manner:

$$k_1 = 1.7536 \times 10^5 \exp(\frac{-1.1374 \times 10^4}{1.9872 x_2}) \tag{11.128}$$

and

$$k_2 = 2.4885 \times 10^{10} \exp(\frac{-2.2748 \times 10^4}{1.9872 x_2}). \tag{11.129}$$

In addition, there is an upper constraint on the temperature

$$x_2(t) \leq 460. \tag{11.130}$$

The problem is to find u in the time interval $0 \leq t < t_f$ such that the performance index

$$I = x_1(t_f), \quad t_f = 5\ \text{min} \tag{11.131}$$

is maximized.

In order to deal with the inequality state constraint in Eq. (11.130), we introduce the state constraint variable x_3 by

$$\frac{dx_3}{dt} = \begin{cases} 0 & \text{if } x_2 - 460 \leq 0 \\ x_2 - 460 & \text{if } x_2 - 460 > 0 \end{cases} \tag{11.132}$$

with the initial condition $x_3(0) = 0$. Therefore, whenever there is a state constraint violation, x_3 is increased in value. If at t_f, $x_3(t_f)$ is zero, then no violations of the constraint equation have occurred.

Since the performance index I is to be maximized, we choose as the augmented performance index to be minimized as

$$J = -I + \theta x_3(t_f) \tag{11.133}$$

Table 11.6: Performance index obtained after 10 passes as a function of the number of grid points N for Example 7

Number of grid points	Performance index I	
N	$\theta = 1$	$\theta = 0.1$
1	0.39357	0.39431
3	0.67715	0.67715
5	0.67713	0.67713
7	0.63905	0.64110
9	0.67722	0.67722
11	0.67722	0.67722
13	0.67722	0.67722
15	0.67723	0.67723
17	0.67723	0.67723
19	0.67723	0.67723
21	0.67723	0.67723

where θ is a positive penalty function factor.

We chose $\theta = 1$, $P = 48$ stages, $\gamma = 0.9$, $\eta = 0.70$, $R = 10$, initial control policy of $u^{(0)} = 0.25$, initial region size of 0.5, and carried out 10 passes, each consisting of 30 iterations for different number of grid points N. Repetition of the runs with $\theta = 0.1$ gave essentially the same type of results. The penalty function avoided any constraint violation after the 10 passes. The results in Table 11.6 show that the maximum value of $I = 0.67723$ was quite difficult to obtain. More than 13 grid points were required to get the optimum value of the yield to 5 figures. The optimal control policy is given in Figure 11.17 and the temperature profile is given in Figure 11.18. It is observed that the upper constraint on the temperature is not violated.

Example 8: Fed-batch fermentor

The fed-batch fermentor involving biosynthesis of penicillin as studied by Lim *et al.* [9] and revised by Cuthrell and Biegler [2] is described by the four differential equations

$$\frac{dx_1}{dt} = g_1 x_1 - \frac{x_1}{500 x_4} u \tag{11.134}$$

$$\frac{dx_2}{dt} = g_2 x_1 - 0.01 x_2 - \frac{x_2}{500 x_4} u \tag{11.135}$$

$$\frac{dx_3}{dt} = -\frac{g_1 x_1}{0.47} - \frac{g_2 x_1}{1.2} - \frac{0.029 x_3 x_1}{0.0001 + x_3} + \frac{500 - x_3}{500 x_4} u \tag{11.136}$$

$$\frac{dx_4}{dt} = \frac{u}{500} \tag{11.137}$$

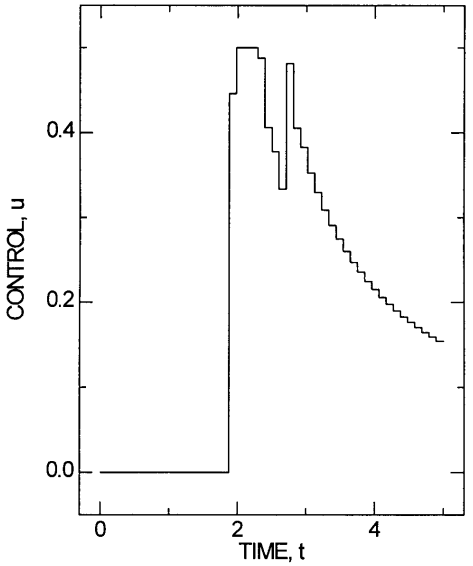

Figure 11.17: Optimal control policy with $P = 48$ for Example 7

with

$$g_1 = \frac{0.11x_3}{0.006x_1 + x_3} \tag{11.138}$$

$$g_2 = \frac{0.0055x_3}{0.0001 + x_3(1 + 10x_3)}. \tag{11.139}$$

The initial state is

$$\mathbf{x}(0) = [1.5 \quad 0 \quad 0 \quad 7 \quad]^T. \tag{11.140}$$

The constraints on the feed rate are

$$0 \leq u \leq 50, \tag{11.141}$$

and on the state variables are

$$0 \leq x_1 \leq 40 \tag{11.142}$$

$$0 \leq x_3 \leq 25 \tag{11.143}$$

$$x_4(t_f) - 10 = 0 \tag{11.144}$$

where t_f is the batch time. The performance index to be maximized is the total amount of penicillin produced at time t_f, given by

$$I = x_2(t_f)x_4(t_f). \tag{11.145}$$

The optimal control problem is to find the batch time t_f and the control policy $u(t)$ in the time interval $0 \leq t < t_f$ so that the performance index given in Eq. (11.145) is

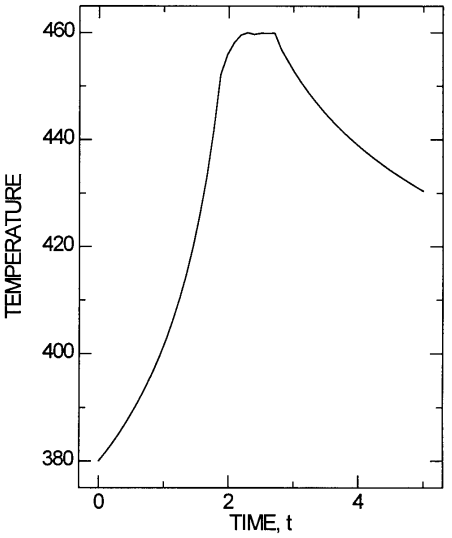

Figure 11.18: Optimal temperature profile for Example 7 resulting from the use of the control policy in Figure 11.17

maximized and all the constraints are satisfied. Cuthrell and Biegler [2] obtained a maximum performance index of 87.69 at a batch time of $t_f = 132.14$ h. By breaking the time interval into 20 time stages of equal length and using iterative dynamic programming Luus [11] was able to improve the yield to 87.948 at $t_f = 132.00$ h. The same control policy was obtained by Gupta [5]. Further refinements in handling inequality constraints led to the value 87.959 by Mekarapiruk and Luus [21]. Recently, Luus and Hennessy [16] showed that the previous investigators had missed the global optimum, which for $t_f = 132.00$ h and 20 stages of equal length is $I = 87.9964$. To enable direct comparisons to be made to these papers, we consider the problem where the batch time is specified as $t_f = 132$ h.

As was done by Mekarapiruk and Luus [21], and by Luus and Hennessy [16] to handle the state inequality constraints, we introduce the state constraint variables x_5 and x_6 through the differential equations

$$\frac{dx_5}{dt} = \begin{cases} x_1 - 40 & \text{if } x_1 > 40 \\ 0 & \text{if } 0 \le x_1 \le 40 \\ 1000 & \text{if } x_1 < 0 \end{cases} \tag{11.146}$$

$$\frac{dx_6}{dt} = \begin{cases} x_3 - 25 & \text{if } x_3 > 25 \\ 0 & \text{if } 0 \le x_3 \le 25 \\ 1000 & \text{if } x_3 < 0. \end{cases} \tag{11.147}$$

The initial conditions for x_5 and x_6 are chosen to be zero. If constraints are violated then these auxiliary variables will become positive. We now form the augmented

performance index to be minimized:

$$J = -I + \theta_1(x_4(t_f) - 10 - s)^2 + \theta_2 x_5(t_f) + \theta_3 x_6(t_f). \tag{11.148}$$

Here we take all the penalty function factors to be equal, i.e.,

$$\theta = \theta_1 = \theta_2 = \theta_3. \tag{11.149}$$

Since the computations are very time-consuming and a small number of passes is intended, instead of after every pass, we are updating the shifting term s after every iteration according to

$$s^{j+1} = s^j - (x_4(t_f) - 10), \tag{11.150}$$

where j denotes the iteration number in the optimization procedure.

To solve this problem with IDP we chose $P = 15$ time stages of equal length, initial control policy $u^{(0)} = 11.9$, initial region size $r_{in} = 10$, region contraction factor $\gamma = 0.90$, region restoration factor $\eta = 0.80$, $R = 11$, and allowed 5 passes, each consisting of 30 iterations, with $\theta = 5$ and $\theta = 50$, for different values of the number of grid points N. As can be seen in Tables 11.7 with penalty function factor $\theta = 5$, the global optimum $I = 87.970$ was difficult to obtain because of the existence of local optima. The volume constraint is well satisfied. We have entered only those values for which the volume is not greater than 10 and for which $x_5(t_f)$ and $x_6(t_f)$ are zero, because even a small constraint violation can have a large effect on the performance index. The value of the shifting parameter was $s = -2.89$. For the volume constraint, a small amount in excess of 10 will increase the performance index substantially, as can be seen from the relatively large value of the shifting term s, where we would expect an increase in I of 2.89×10^{-3} if the volume constraint is relaxed by 10^{-4}.

The set of runs was repeated with the use of $\theta = 50$. As can be seen in Table 11.8, the penalty function factor θ does not have much influence in getting the global optimum. Here the shifting parameter obtained at the optimum was $s = -0.289$. The global optimum $I = 87.970$ was again obtained only once.

To improve the chances for getting the global optimum, we can examine the findings of Luus and Galli [15] that a larger initial region size promotes convergence to the global optimum. Therefore, the set of runs was repeated, using the same parameters, except the initial region size was doubled from 10 to 20 and 10 passes were allowed. The penalty factor $\theta = 5$ was used to get the results of Table 11.9. The global optimum $I = 87.970$ was obtained with $N = 3, 5$ and 7. The use of $N = 13$ led to the second best local optimum $I = 87.9572$, and the other two values for N gave two other local optima: $I = 87.924$ and $I = 87.920$. The control policies corresponding to these four local optima are shown in Table 11.10. Trajectories for the three constrained state variables obtained with the use of the global optimum control policy are shown in Figure 11.19. It is clear that the constraints are not violated.

Table 11.7: Performance index obtained after 5 passes as a function of the number of grid points N for Example 8 with $\theta = 5$

Number of grid points N	Performance Index I	Final volume $x_4(t_f)$
3	87.668	10.00000
5	87.920	9.99995
7	87.920	9.99998
9	87.920	9.99995
11	**87.970**	10.00000
13	87.919	9.99998

Table 11.8: Performance index obtained after 5 passes as a function of the number of grid points N for Example 8 with $\theta = 50$

Number of grid points N	Performance Index I	Final volume $x_4(t_f)$
3	87.785	9.99991
5	87.472	10.00000
7	87.667	9.999989
9	87.472	9.99997
11	87.920	10.00000
13	**87.970**	9.99999

Table 11.9: Performance index obtained after 10 passes as a function of the number of grid points N for Example 8 with $\theta = 5$ and initial region size = 20

Number of grid points N	Performance Index I	Final volume $x_4(t_f)$
3	**87.970**	10.00001
5	**87.970**	10.00000
7	**87.970**	10.00000
9	87.924	10.00000
11	87.920	9.99999
13	87.957	10.00000

Table 11.10: Piecewise constant control policies for the best four local optima for Example 8 obtained by using $P = 15$ stages

	Control policy			
Stage number	$I = 87.970$	$I = 87.957$	$I = 87.924$	$I = 87.920$
1	3.741	4.280	3.383	3.747
2	6.970	9.241	15.978	6.983
3	13.187	48.712	42.861	13.219
4	44.624	8.179	8.038	44.575
5	8.680	8.518	8.498	8.682
6	8.916	8.745	8.733	8.914
7	9.058	8.887	8.878	9.060
8	9.162	8.987	8.985	9.162
9	9.244	9.071	9.070	9.249
10	9.317	9.146	9.144	9.314
11	9.385	9.213	9.212	9.381
12	9.449	9.277	9.276	9.451
13	9.513	9.339	9.339	9.508
14	9.574	9.401	9.402	9.575
15	9.6345	9.458	9.461	9.636

To show the usefulness of the sensitivity factor $-2\theta s$ to predict the effect of changing the upper constraint on the volume, we compare the predicted effect from

$$I = 87.97 - 2\theta s(V_{max} - 10) \tag{11.151}$$

to the computed values in the range $9.5 \leq V_{max} \leq 10.5$. As is seen in Figure 11.20, the agreement is excellent.

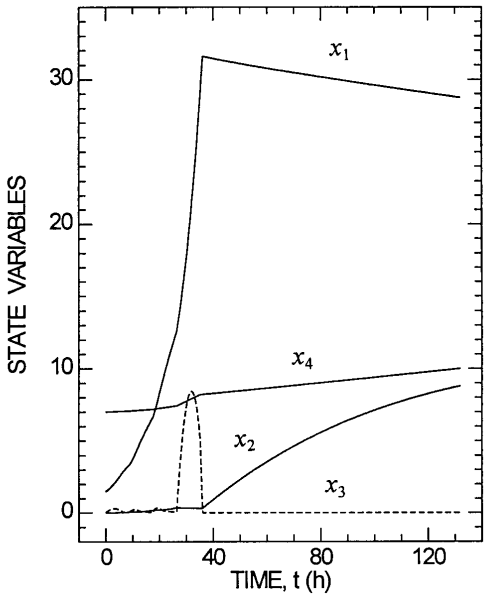

Figure 11.19: Trajectories for the three constrained state variables obtained by the optimal control policy

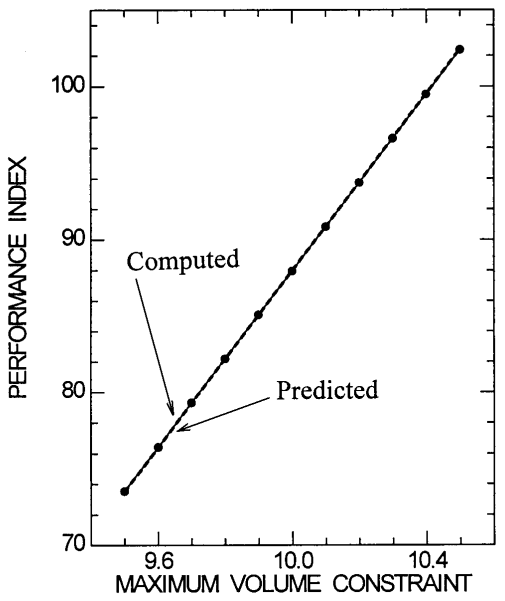

Figure 11.20: Comparison of the predicted and computed values of the performance index I when the volume constraint is changed

11.4 References

[1] Chen, G. and Mills, W.H.: "Finite elements and terminal penalization for quadratic cost optimal control problems governed by ordinary differential equations", *SIAM J. Control and Optimization* **19** (1981), 744-764.

[2] Cuthrell, J.E. and Biegler, L.T.: "Simultaneous optimization and solution methods for batch reactor control profiles", *Comput. Chem. Eng.* **13** (1989), 49-62.

[3] Dadebo, S.A. and McAuley, K.B.: "Dynamic optimization of constrained chemical engineering problems using dynamic programming", *Comput. Chem. Eng.* **19**(1995), 513-525.

[4] Goh, C.J. and Teo, L.K.: "Control parametrization: a unified approach to optimal control problems with generalized constraints", *Automatica* **24** (1988), 3-18.

[5] Gupta, Y.P.: "Semiexhaustive search for solving nonlinear optimal control problems", *Ind. Eng. Chem. Res.* **34** (1995), 3878-3884.

[6] Hull, T.E., Enright, W.D., and Jackson, K.R. *User Guide to DVERK–a Subroutine for Solving Nonstiff ODE's*. Report 100 (1976); Department of Computer Science, University of Toronto, Canada.

[7] Ko, D.Y.C. and Stevens, W.F.: "Study of singular solutions in dynamic optimization", *AIChE J.* **17** (1971), 160-166.

[8] Lapidus, L. and Luus, R.: *Optimal Control of Engineering Processes*, Blaisdell, Waltham, Mass. (1967).

[9] Lim, H.C., Tayeb, Y.J., Modak, J.M., and Bonte, P.: "Computational algorithms for optimal feed rates for a class of fed-batch fermentation: Numerical results for penicillin and cell mass production", *Biotechnol. Bioeng.* **28** (1986), 1408-1420.

[10] Luus, R.: "Application of iterative dynamic programming to state constrained optimal control problems", *Hung. J. Ind. Chem.* **19** (1991), 245-254.

[11] Luus, R.: "Optimization of fed-batch fermentors by iterative dynamic programming", *Biotechnol. Bioeng.* **41** (1993), 599-602.

[12] Luus, R.: "Optimal control of batch reactors by iterative dynamic programming", *J. Proc. Control* **4** (1994), 218-226.

[13] Luus, R.: "Handling difficult equality constraints in direct search optimization", *Hung. J. Ind. Chem.* **24** (1996), 285-290.

[14] Luus, R.: "Use of iterative dynamic programming with variable stage lengths and fixed final time", *Hung. J. Ind. Chem.* **24** (1996), 279-284.

[15] Luus, R. and Galli, M.: "Multiplicity of solutions in using dynamic programming for optimal control", *Hung. J. Ind. Chem.* **19** (1991), 55-62.

[16] Luus, R. and Hennessy, D.: "Optimization of fed-batch reactors by the Luus-Jaakola optimization procedure", *Ind. Eng. Chem. Res.* **38** (1999), 1948-1955.

[17] Luus, R., Mekarapiruk, W., and Storey, C.: "Evaluation of penalty functions for optimal control", *Proc. of International Conference on Optimization Techniques and Applications (ICOTA '98)* (1998), Perth, Western Australia, July 1-3, 1998, Curtin Printing Services, pp. 724-731.

[18] Luus, R. and Rosen, O.: "Application of dynamic programming to final state constrained optimal control problems", *Ind. Eng. Chem. Res.* **30** (1991), 1525-1530.

[19] Luus, R. and Storey, C.: "Optimal control of final state constrained systems", *Proc. IASTED International Conference on Modelling, Simulation and Control*, Singapore, Aug.11-14, 1997, 245-249.

[20] Luus, R. and Wyrwicz, R.: "Use of penalty functions in direct search optimization", *Hung. J. Ind. Chem.* **24** (1996), 273-278.

[21] Mekarapiruk, W. and Luus, R.: "Optimal control of inequality state constrained systems", *Ind. Eng. Chem. Res.* **36** (1997), 1686-1694.

[22] Mekarapiruk, W. and Luus, R.: "Optimal control of final state constrained systems", *Can. J. Chem. Eng.* **75** (1997), 806-811.

[23] Park, S. and Ramirez, W.F.: "Optimal production of secreted protein in fed-batch reactors", *AIChE J.* **34** (1988), 1550-1558.

[24] Reddy, K.V. and Husain, A.: "Computation of optimal control policy with singular subarc", *Can. J. Chem. Eng.* **59** (1981), 557-559.

[25] Sakawa, Y. and Shindo, Y.: "Optimal control of container cranes", *Automatica* **18** (1982), 257-266.

[26] Teo, K.L., Goh, C.J., and Wong, K.H.: *A Unified Computational Approach to Optimal Control Problems*, Longman Scientific & Technical, Wiley, New York (1991).

Chapter 12

Time optimal control

12.1 Introduction

There are numerous situations where time optimal control problems arise. For example, in startup of a chemical reactor, it is frequently desired to reach the desired state of operation as fast as possible. In operating a crane system, one frequently would like the trolley to reach a desired point in minimum time. In the drug displacement problem, it is desirable to reach some specified concentration of drugs in a patient's bloodstream in minimum time. In the operation of a robot arm to deliver a payload to a certain location, we encounter the time optimal control problem. Therefore, a considerable amount of research effort has gone into solving the time optimal control problem. For linear systems, linear programming may be used to determine the minimum number of time steps to reach the desired state, but for high dimensional nonlinear systems the determination of time optimal control policy is generally very difficult.

In this chapter we wish to examine several approaches that have been used with IDP to solve some time optimal control problems.

12.2 Time optimal control problem

Let us consider the system described by the differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{12.1}$$

where the initial state $\mathbf{x}(0)$ is given. The state vector $\mathbf{x}$ is an $(n \times 1)$ vector and $\mathbf{u}$ is an $(m \times 1)$ control vector bounded by

$$\alpha_j \leq u_j \leq \beta_j, \quad j = 1, 2, \cdots, m. \tag{12.2}$$

At the final time t_f the state variables must be at the desired values, i.e.,

$$h_i = x_i(t_f) - x_i^d = 0, \quad i = 1, 2, \cdots, n. \tag{12.3}$$

The time optimal control problem is to find the control policy $\mathbf{u}(t)$ in the time interval $0 \leq t < t_f$ that minimizes the performance index

$$I = t_f. \tag{12.4}$$

Numerically, it is required that in a minimum value of the final time t_f, we satisfy the constraint

$$|x_i(t_f) - x_i^d| \leq \epsilon_i, \qquad i = 1, 2, \cdots, n, \tag{12.5}$$

where ϵ_i is some tolerance, such as the measurement error.

12.3 Direct approach to time optimal control

To solve this optimal control problem, we follow the suggestion of Luus [10] to choose the augmented performance index to be minimized as

$$J = t_f + \theta \sum_{i=1}^{n} (x_i(t_f) - x_i^d - s_i)^2, \tag{12.6}$$

and update the shifting terms s_i after every pass of IDP according to

$$s_i^{q+1} = s_i^q - (x_i(t_f) - x_i^d), \tag{12.7}$$

where q is the pass number in IDP.

To measure the closeness of the final states to the desired states we use the norm of the sum of absolute deviations from the desired state, namely

$$S = \sum_{i=1}^{n} |x_i(t_f) - x_i^d|. \tag{12.8}$$

In order to solve the time optimal control problem by IDP, we transform the continuous control policy into a piecewise constant control problem by dividing the time interval $[0, t_f]$ into P stages, each of variable length,

$$v(k) = t_k - t_{k-1}, \qquad k = 1, 2, \cdots, P. \tag{12.9}$$

Computationally, we impose the condition

$$v(k) > 0, \qquad k = 1, 2, \cdots, P, \tag{12.10}$$

so that we do not deal with negative stage lengths.

As we did in Chapter 9, based on the idea of Bojkov and Luus [3], we transform the time variable through the relationship

$$dt = v(k) P d\tau \tag{12.11}$$

so that in normalized time all the time stages are of equal length

$$\tau_k = \frac{k}{P}, \qquad k = 1, 2, \cdots, P. \tag{12.12}$$

The differential equation in the time interval $\tau_{k-1} \leq \tau < \tau_k$ becomes

$$\frac{d\mathbf{x}}{d\tau} = v(k) P \mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{12.13}$$

With this transformation, at $t = t_f$, the transformed time variable $\tau = 1$, and the time to reach the desired state is

$$t_f = \sum_{k=1}^{P} v(k). \tag{12.14}$$

Minimization of the augmented performance index in Eq.(12.6) should give the minimum value to t_f and ensure that the state variables at the final time are close to the desired values. In order to test the computational viability of this approach, we consider several examples. In all the examples we used the DVERK [7] subroutine, listed in Appendix E, to integrate the differential equations.

12.4 Examples

12.4.1 Example 1: Bridge crane system

Let us consider the bridge crane system used by Moon and VanLandingham [13], where the system is described by four differential equations

$$\frac{dx_1}{dt} = x_2 \tag{12.15}$$

$$\frac{dx_2}{dt} = u \tag{12.16}$$

$$\frac{dx_3}{dt} = x_4 \tag{12.17}$$

$$\frac{dx_4}{dt} = -0.98 x_3 + 0.1 u \tag{12.18}$$

with $\mathbf{x}^T(0) = \mathbf{0}$, and the desired final state is $\mathbf{x}^d = [\ 15\ \ 0\ \ 0\ \ 0\ ]^T$. The constraints on the control are

$$-1 \leq u \leq 1. \tag{12.19}$$

It is required to find the control policy that minimizes the performance index

$$I = t_f. \tag{12.20}$$

We therefore choose the augmented performance index

$$J = t_f + \theta[(x_1(t_f) - 15 - s_1)^2 + (x_2(t_f) - s_2)^2 + (x_3(t_f) - s_3)^2 + (x_4(t_f) - s_4)^2], \quad (12.21)$$

where θ is a positive penalty function factor and the shifting terms s_i are updated after every pass of IDP according to Eq. (12.7).

To solve this problem, we made two preliminary runs with $P = 5$ time stages, each chosen initially to be of length 1.0. We chose the initial control policy to be zero, and initial region sizes for both control and stage lengths to be 1.0. The penalty function factor θ was chosen to be 10, $R = 50$, region contraction factor $\gamma = 0.80$, region restoration factor $\eta = 0.95$, and 100 passes each consisting of 30 iterations were performed. The runs were made with $N = 3$ grid points, and with $N = 5$. The convergence profiles of these runs were very similar, but a smaller value for t_f was obtained with $N = 5$, namely, 8.58012 versus $t_f = 8.58162$ with $N = 3$. The convergence profile with $N = 5$ is shown in Figure 12.1. The resulting control policy consisted of 4 stages with $u(0) = 1.0$, $u(1) = -1.0$, $u(2) = 1.0$, and $u(3) = -1.0$, with the length of the stages 2.98534, 1.30471, 1.30394, and 2.98492 respectively. There was also a very short stage of length 0.00119 with a control value of 0.2963 before the last stage. The shifting terms were $s_1 = 0.0090816$, $s_2 = -0.038988$, $s_3 = 0.054761$, and $s_4 = -0.110204$.

To refine these results, we dropped the very short stage, and used the results of this preliminary run as the initial conditions, including the values for the shifting terms, and performed a run with $R = 25$, putting the region size for control to zero, and reducing the value of the region size for stage lengths to 0.1. After 100 passes, the final time was reduced to $t_f = 8.58010$ and the sum of absolute value norm of deviations was 1.098×10^{-7}. The stage lengths converged to $v(1) = 2.98534$, $v(2) = 1.30471$, $v(3) = 1.30471$, and $v(4) = 2.98534$. It is interesting to note the symmetry where $v(1) = v(4)$ with $u(0) = 1, u(3) = -1$, and $v(2) = v(3)$ with $u(1) = -1, u(2) = 1$. The run was repeated with $\theta = 1$, and $\theta = 100$.

Convergence patterns for all the three values of θ were very similar, as is shown for $\theta = 1$ and $\theta = 10$ in Figure 12.2. Identical values were obtained for the stage lengths. However, the shifting terms, given in Table 12.1, for the three cases are different, yielding the same value for the sensitivity, as given in the fifth column. In fact, the factor $-2\theta s_i$ gives the adjoint variable at the final time, corresponding to the variable i, and should be the same for all the values of θ. It should be noted that the values for the sensitivity in the fifth column of Table 12.1 are close to the final values of the adjoint variables obtained by Moon and VanLandingham [13], given in the sixth column of Table 12.1.

The time optimal control policy is shown in Figure 12.3.

Table 12.1: Shifting terms and comparison to adjoint variables at t_f for Example 1

Variable	Shifting terms, s_i			Sensitivity	Adjoint variable at
i	$\theta = 1$	$\theta = 10$	$\theta = 100$	$-2\theta s_i$	$t_f = 8.5801$ [13]
1	0.090861	0.009086	0.0009086	-0.182	-0.182
2	-0.389797	-0.038980	-0.0038981	0.780	0.782
3	0.547945	0.054793	0.0054782	-1.096	-1.086
4	-1.102032	-0.110203	-0.0110194	2.204	2.216

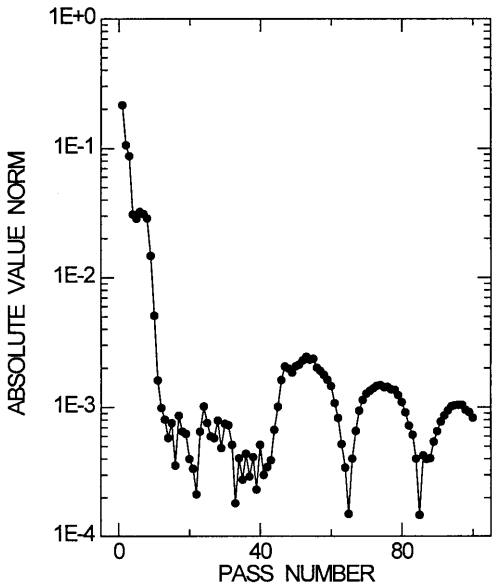

Figure 12.1: Sum of absolute deviations of the final state variables from the desired values for the preliminary run $N = 5$ grid points for Example 1

12.4.2 Example 2: Two-link robotic arm

Let us consider the minimum time control of a two-link robotic arm as considered by Weinreb and Bryson [17], Lee [8], Bojkov and Luus [1], and Luus [10], where the equations are given by

$$\frac{dx_1}{dt} = \frac{(\sin x_3)[\frac{9}{4}(\cos x_3)x_1^2 + 2x_2^2] + \frac{4}{3}(u_1 - u_2) - \frac{3}{2}(\cos x_3)u_2}{\frac{31}{36} + \frac{9}{4}\sin^2 x_3} \tag{12.22}$$

$$\frac{dx_2}{dt} = \frac{-[(\sin x_3)(\frac{7}{2}x_1^2 + \frac{9}{4}(\cos x_3)x_2^2) - \frac{7}{3}u_2 + \frac{3}{2}(\cos x_3)(u_1 - u_2)]}{\frac{31}{36} + \frac{9}{4}\sin^2 x_3} \tag{12.23}$$

$$\frac{dx_3}{dt} = x_2 - x_1 \tag{12.24}$$

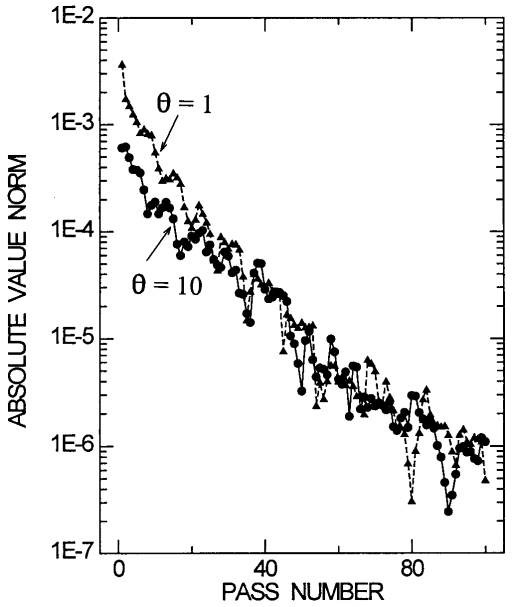

Figure 12.2: Sum of absolute deviations of the final state variables from the desired values for two values of the penalty function factor for Example 1

$$\frac{dx_4}{dt} = x_1 \tag{12.25}$$

where x_1 and x_2 are the inertial velocities of the shoulder and elbow links, and x_3 and x_4 are the angles. The initial state is $\mathbf{x}(0) = [\ 0\ \ 0\ \ 0.5\ \ 0\ ]^T$, and the desired final state is $\mathbf{x}^d = [\ 0\ \ 0\ \ 0.5\ \ 0.522\ ]^T$. The constraints on the control are

$$-1 \leq u_j \leq 1, \qquad j = 1, 2. \tag{12.26}$$

It is required to find the control policy that minimizes the performance index

$$I = t_f. \tag{12.27}$$

We therefore choose the augmented performance index

$$J = t_f + \theta[(x_1(t_f) - s_1)^2 + (x_2(t_f) - s_2)^2 + (x_3(t_f) - 0.5 - s_3)^2 + (x_4(t_f) - 0.522 - s_4)^2], \tag{12.28}$$

where θ is a positive penalty function factor and the shifting terms s_i are updated after every pass of IDP. To keep track of the convergence, we record after every pass the norm of the absolute value deviations of the final state from the desired state

$$S = |x_1(t_f)| + |x_2(t_f)| + |x_3(t_f) - 0.5| + |x_4(t_f) - 0.522|. \tag{12.29}$$

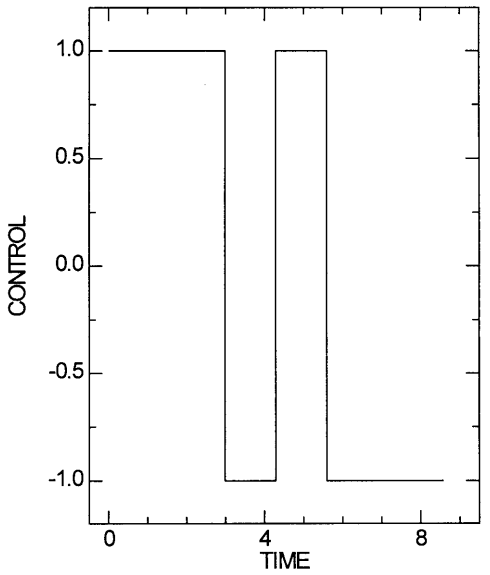

Figure 12.3: Time optimal control policy for Example 1

We chose $P = 5$ time stages each initially of length 0.5, and used the same parameters for IDP as for Example 1, namely, $\gamma = 0.80$, $\eta = 0.95$, $\theta = 10$, $R = 50$, and used 100 passes, each consisting of 30 iterations. The initial control policy was chosen as zero, and the initial region sizes for control and stage lengths were chosen as 1.0 and 0.5 respectively. As can be seen from Figure 12.4, there does not appear to be much difference between the use of $N = 3$, and $N = 5$, but the use of $N = 5$ gave a final time $t_f = 2.9822$, whereas $N = 3$ gave $t_f = 2.9824$.

By using the results obtained with $N = 5$ as the initial condition, a refined run was performed with $R = 25$, yielding after 100 passes a sum of deviations of $S = 1.219 \times 10^{-7}$ and a final time of $t_f = 2.98228$. The shifting terms with $\theta = 10$ are $s_1 = -0.13434$, $s_2 = -0.09387$, $s_3 = 0.03620$, and $s_4 = 0.09739$. The time optimal control policy is given in Figure 12.5.

12.4.3 Example 3: Drug displacement problem

Let us consider the time optimal drug displacement problem, considered by Maurer and Weigand [12], and Bojkov and Luus [1], where a desired level of two drugs, warfarin and phenylbutazone, must be reached in a patient's bloodstream in minimum time. The equations expressing the changes in the concentrations of these two drugs in the patient's bloodstream are

$$\frac{dx_1}{dt} = g_1[g_4(0.02 - x_1) + 46.4x_1(u - 2x_2)] \tag{12.30}$$

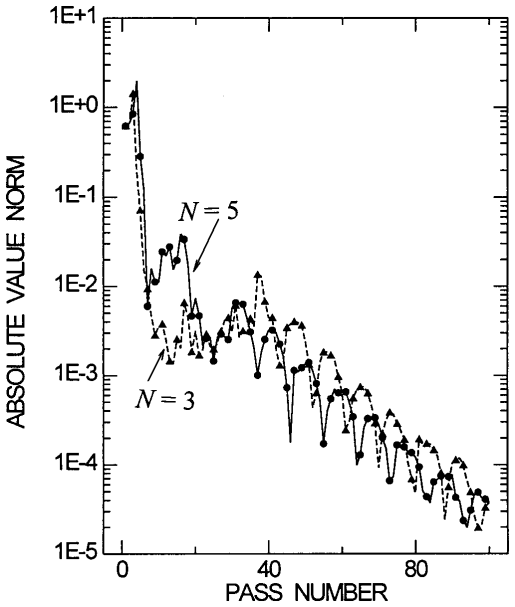

Figure 12.4: Convergence profiles for the sum of absolute deviations of the final state variables from the desired values as a function of the number of grid points for Example 2

$$\frac{dx_2}{dt} = g_1[g_3(u - 2x_2) + 46.4(0.02 - x_1)] \tag{12.31}$$

where

$$g_2 = 1 + 0.2(x_1 + x_2) \tag{12.32}$$

$$g_3 = g_2^2 + 232 + 46.4x_2 \tag{12.33}$$

$$g_4 = g_2^2 + 232 + 46.4x_1 \tag{12.34}$$

and

$$g_1 = \frac{g_2^2}{g_3 g_4 - 2152.96 x_1 x_2}. \tag{12.35}$$

The state variable x_1 denotes the concentration of warfarin, and x_2 is the concentration of phenylbutazone.

The scalar control variable u is the rate of infusion of phenylbutazone, and is bounded by

$$0 \leq u \leq 8. \tag{12.36}$$

The initial concentrations are given by

$$\mathbf{x}(0) = [\ 0.02 \quad 0\ ]^T \tag{12.37}$$

and the desired values for the concentrations at the time t_f are

$$\mathbf{x}^d = [\ 0.02 \quad 2.00\ ]^T. \tag{12.38}$$

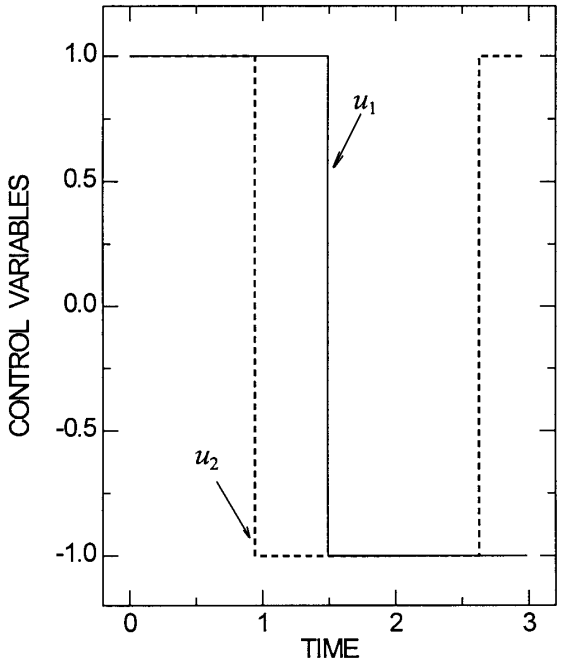

Figure 12.5: Time optimal control policy for Example 2

The objective is to find the control u such that the final time is minimized, so the performance index is

$$I = t_f. \tag{12.39}$$

To handle the final state constraints, we use the augmented performance index

$$J = t_f + \theta[(x_1(t_f) - 0.02 - s_1)^2 + (x_2(t_f) - 2 - s_2)^2] \tag{12.40}$$

where θ is a positive penalty function factor, and the shifting terms are updated after every pass by

$$s_1^{q+1} = s_1^q - (x_1(t_f) - 0.02) \tag{12.41}$$

$$s_2^{q+1} = s_2^q - (x_2(t_f) - 2), \tag{12.42}$$

where q is the pass number.

To solve this time optimal control problem by IDP, as in the two preceding examples we chose $P = 5$ time stages. We used the same parameters for IDP as before, namely $\gamma = 0.80$, $\eta = 0.95$, $R = 50$, $N = 5$, and used 100 passes, each consisting of 30 iterations. We chose $\theta = 10^6$, initial stage lengths = 30, initial control = 0, and initial region sizes for control and stage lengths of 8 and 30 respectively. As is seen in Figure 12.6, the convergence is quite regular, giving a sum of absolute deviation of 10^{-7} at the end of 100 passes. The control policy consisted of using $u = 8$ for the first 4 stages and then using $u = 0$ for the last stage of length 32.309, giving a final time $t_f = 221.46167$. The shifting terms were $s_1 = -0.0171$ and $s_2 = 0.0000338$.

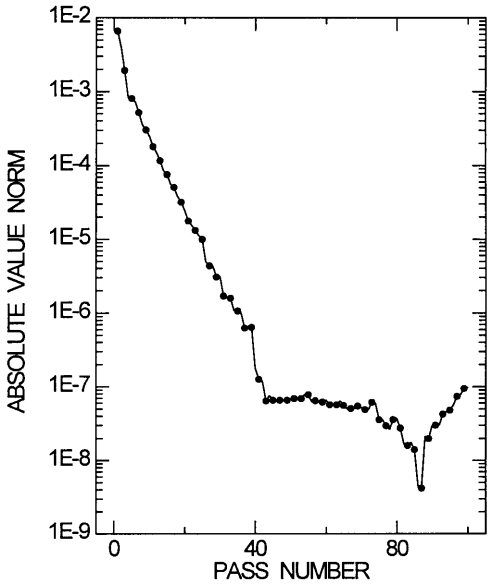

Figure 12.6: Sum of absolute deviations of the final state variables from the desired values with $N = 5$ for Example 3

The results were then refined by a run consisting of only two stages, to give the switching time at $t = 189.15319$ and the length of the last stage of 32.30708. The optimum final time t_f was reduced slightly to 221.46027, giving the sum of absolute deviations $S = 2.35 \times 10^{-10}$. The time optimal control policy is shown in Figure 12.7.

12.4.4 Example 4: Two-stage CSTR system

A very challenging problem is the determination of the time optimal control policy for the two-stage CSTR system that we considered in Chapters 10 and 11 for other types of control objectives. Let us now consider the two-stage CSTR system used for optimal control studies by Edgar and Lapidus [6], Luus [9], Nishida *et al.* [14], Rosen and Luus [15], Dadebo and McAuley [5], and by Luus *et al.* [11]. The system is described by

$$\frac{dx_1}{dt} = -3x_1 + g_1(\mathbf{x}) \tag{12.43}$$

$$\frac{dx_2}{dt} = -11.1558x_2 + g_1(\mathbf{x}) - 8.1558(x_2 + 0.1592)u_1 \tag{12.44}$$

$$\frac{dx_3}{dt} = 1.5(0.5x_1 - x_3) + g_2(\mathbf{x}) \tag{12.45}$$

$$\frac{dx_4}{dt} = 0.75x_2 - 4.9385x_4 + g_2(\mathbf{x}) - 3.4385(x_4 + 0.122)u_2 \tag{12.46}$$

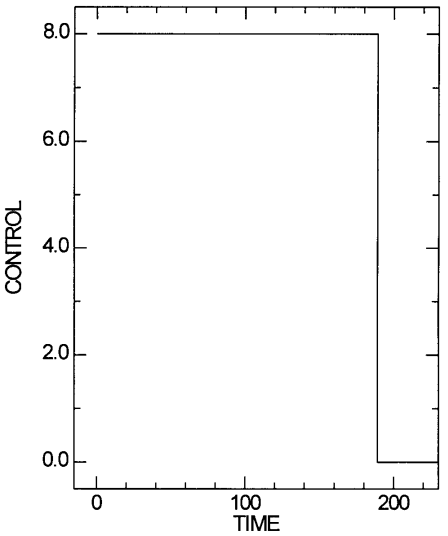

Figure 12.7: Time optimal control policy for Example 3

where the reaction term in the first tank is given by

$$\begin{aligned} g_1(\mathbf{x}) &= 1.5 \times 10^7 (0.5251 - x_1) \exp(\frac{-10}{x_2 + 0.6932}) \\ &\quad -1.5 \times 10^{10} (0.4748 + x_1) \exp(\frac{-15}{x_2 + 0.6932}) - 1.4280 \end{aligned} \tag{12.47}$$

and the reaction term in the second tank is

$$\begin{aligned} g_2(\mathbf{x}) &= 1.5 \times 10^7 (0.4236 - x_2) \exp(\frac{-10}{x_4 + 0.6560}) \\ &\quad -1.5 \times 10^{10} (0.5764 + x_3) \exp(\frac{-15}{x_4 + 0.6560}) - 0.5086. \end{aligned} \tag{12.48}$$

The initial condition is given by

$$\mathbf{x}(0) = [\ 0.1962 \quad -0.0372 \quad 0.0946 \quad 0\ ]^T \tag{12.49}$$

and the desired state is

$$\mathbf{x}^d = \mathbf{0}. \tag{12.50}$$

The constraints on the control are

$$-1 \leq u_j \leq 1, \quad j = 1, 2. \tag{12.51}$$

The state variables x_1 and x_3 are normalized concentration variables in tanks 1 and 2 respectively, and x_2 and x_4 are normalized temperature variables in tanks 1 and 2

respectively. It is required to find the control policy that minimizes the time to reach the desired state. Therefore, the performance index is

$$I = t_f. \tag{12.52}$$

As before, we introduce the augmented performance index to take into account the final state constraint

$$J = t_f + \theta \sum_{i=1}^{4} (x_i(t_f) - s_i)^2 \tag{12.53}$$

where the shifting terms are updated after every pass of IDP by

$$s_i^{q+1} = s_i^q - x_i(t_f), \qquad i = 1, 2, \cdots, 4. \tag{12.54}$$

The objective is to use IDP to minimize the augmented performance in Eq. (12.53).

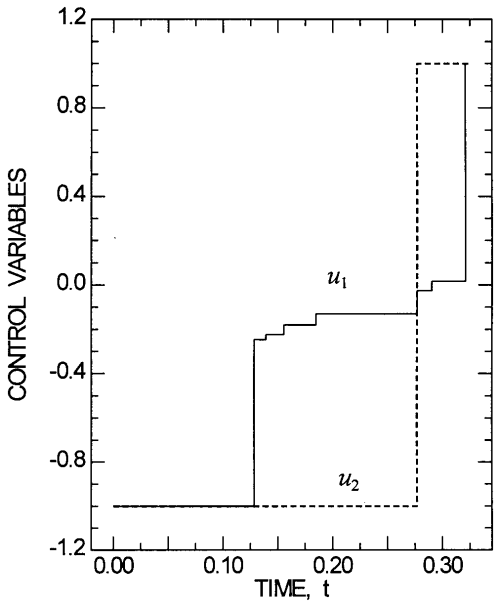

Figure 12.8: Time optimal control policy for Example 4 with the use of 8 stages, giving t_f = 0.323646 with $S = 1.2 \times 10^{-8}$

For this problem we chose the penalty function factor θ to be 100, $P = 20$ time stages, and for the initial control policy we took the optimal control policy from Chapter 11 for the quadratic performance index. We used $N = 1$ grid point, $R = 25$, and carried out 100 passes of 30 iterations each, with $\gamma = 0.80$ and $\eta = 0.95$. The initial region sizes for control were chosen as 1 and for the stage lengths, 0.001. The result was a 9-stage control policy with $t_f = 0.323654$ and $S = 1.902 \times 10^{-6}$. Two

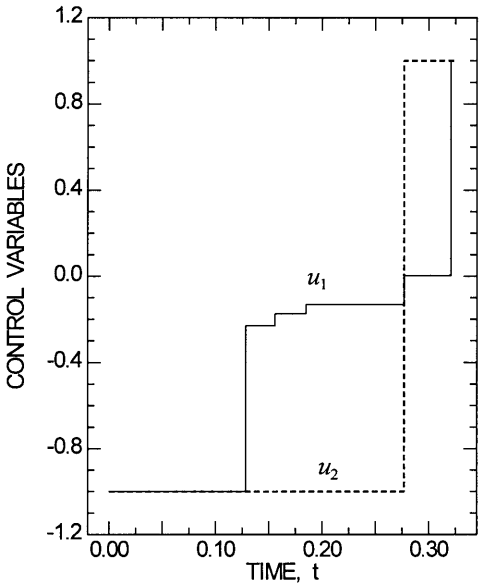

Figure 12.9: Time optimal control policy for Example 4 with the use of 6 stages, giving $t_f = 0.323650$, with $S = 1.4 \times 10^{-8}$

stages were joined and the problem was run again to give the time optimal control policy in Figure 12.8 with $t_f = 0.323646$ with the sum of absolute deviations of the final states from desired states $S = 1.18 \times 10^{-8}$. The shifting terms are $s_1 = 0.0342416$, $s_2 = -0.0001473$, $s_3 = -0.0212781$, and $s_4 = -0.0115108$. It is interesting to note the very short last stage of length 0.00260 with $u_1 = u_2 = 1$.

Stages 2 and 3, and also, 6 and 7, were combined to give a 6-stage control policy which was used as the initial control policy for another run. The resulting 6-stage control policy is given in Figure 12.9. There is a very slight increase, almost negligible, of the final time to $t_f = 0.323650$, and the final state constraints are well satisfied with $S = 1.37 \times 10^{-8}$.

12.4.5 Example 5

Let us next consider the quadruple-integral system of Chung and Wu [4], which was also considered by Bojkov and Luus [2]. The system is described by the four differential equations

$$\frac{dx_1}{dt} = x_2 \tag{12.55}$$

$$\frac{dx_2}{dt} = x_3 \tag{12.56}$$

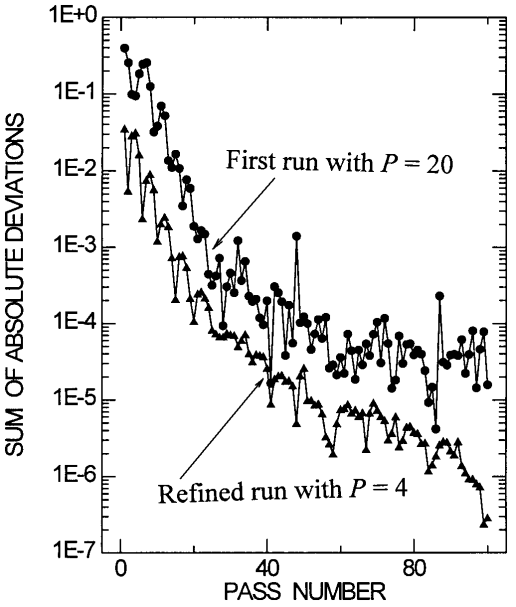

Figure 12.10: Convergence of the time optimal control for Example 5

$$\frac{dx_3}{dt} = x_4 \tag{12.57}$$

$$\frac{dx_4}{dt} = u \tag{12.58}$$

with the initial state $\mathbf{x}(0) = [\ 0.1\ \ 0.2\ \ 0.3\ \ 0\ ]^T$, and the final state $\mathbf{x}^d = \mathbf{0}$.

The constraints on the control are

$$-1 \leq u \leq 1. \tag{12.59}$$

Although there is only the scalar control, the time optimal control policy is quite difficult to obtain, as shown by Bojkov and Luus [2]. Since the goal is to reach the desired state in minimum time, we choose the augmented performance index as

$$J = t_f + \theta \sum_{i=1}^{4} (x_i(t_f) - s_i)^2 \tag{12.60}$$

where the shifting terms are updated after every pass of IDP by

$$s_i^{q+1} = s_i^q - x_i(t_f), \qquad i = 1, 2, \cdots, 4. \tag{12.61}$$

As in Example 4, we chose the penalty function factor $\theta = 100$.

By taking $P = 20$ time stages, each initially of length 0.20, and initial control policy $u_1 = -1$, and using $N = 1$, $R = 25$, $\gamma = 0.80$, $\eta = 0.95$, with initial region sizes for control of 2.0 and for stage lengths 0.05, the convergence was quite regular as

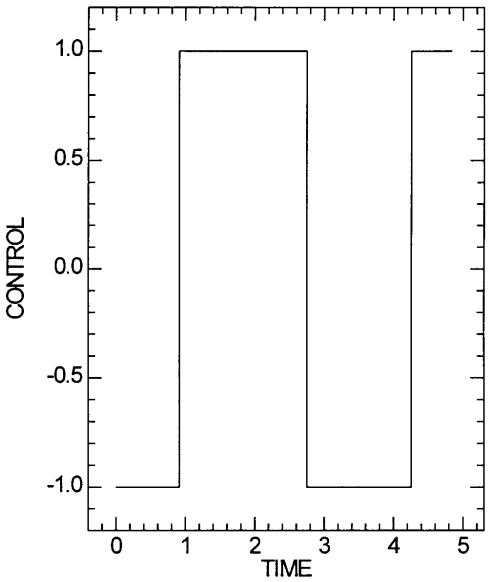

Figure 12.11: Time optimal control policy for Example 5

is shown in Figure 12.10. The resulting four stage bang-bang control policy was used then as the initial condition for the refined run which gave a final time of $t_f = 4.8319$ with the sum of absolute deviations from the final state of $S = 2.89 \times 10^{-8}$. The values of the shifting terms are $s_1 = -0.006372$, $s_2 = 0.013975$,$s_3 = -0.012346$, and $s_4 = 0.005000$.

The resulting time optimal control policy is given in Figure 12.11 and the state trajectories are shown in Figure 12.12. The computation time on a PentiumII/350 for $P = 20$ was 329 s, and for $P = 4$ the computation time was 59 s.

12.5 High dimensional systems

The use of IDP for time optimal control of high dimensional systems has not been fully explored. The approaches used by Bojkov and Luus [2] have been found very useful in the time optimal control of a 20-plate gas absorber. Here we want to just outline the main ideas and the interested reader can refer to this paper for the details. To simplify the discussion, let us assume that the variables have been transformed so that the desired state is the origin.

Rosen *et al.* [16] showed that there is a significant effect of the tolerance to which the origin is reached on the final time t_f in the time optimal control. Although mathematically it is nice to reach the origin to within the computational accuracy of the computer, this is not necessary. For the 6-plate gas absorber, for example, to get to the origin within 10^{-6} tolerance gives a minimum time of $t_f = 6$ min, whereas to get to within 10^{-3} of the origin (within the measurement error) reduces the time

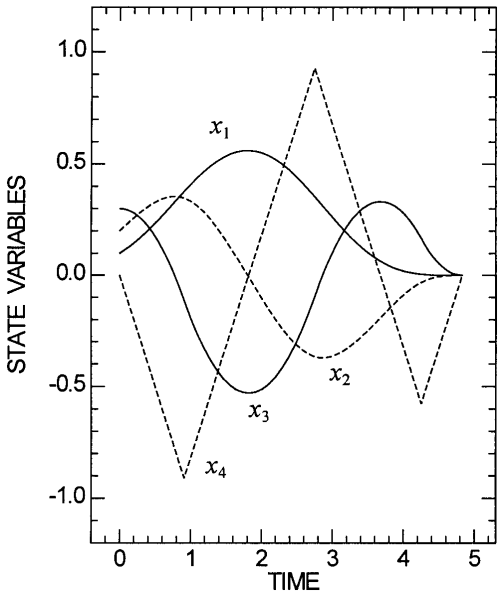

Figure 12.12: State trajectories for Example 5

to $t_f = 4.18$ min. Therefore, we would like to exploit the flexibility of the tolerance based on the accuracy of measurements to reduce the time required to reach the appropriate vicinity of the origin.

Thus we write the final state constraints as

$$|x_i(t_f)| \leq \epsilon_i, \qquad i = 1, 2, \cdots, n, \tag{12.62}$$

where we could have a different tolerance for each state variable. Here, for simplicity, we take all the tolerances to be the same, namely $\epsilon_1 = \epsilon_2 = \cdots = \epsilon_n = \epsilon$.

By trying different structures for the augmented performance index, Bojkov and Luus [2] found that the following structure was useful in solving a 20-plate gas absorber time optimal control problem:

$$J = \sigma \sum_{i=1}^{n} x_i^2(t_f) + \mu \sum_{i=1}^{n} (|x_i(t_f)| - \epsilon)^2 + \theta t_f, \tag{12.63}$$

where

$$\sigma = \begin{cases} 0 & \text{if all } |x_i(t_f)| \leq \epsilon \\ 1 & \text{if any } |x_i(t_f)| > \epsilon \end{cases} \tag{12.64}$$

$$\mu = \begin{cases} 0 & \text{if all } |x_i(t_f)| \leq \epsilon \\ 1 & \text{if any } |x_i(t_f)| > \epsilon \end{cases} \tag{12.65}$$

$$\theta = \begin{cases} \epsilon^2/(nQ) & \text{if all } |x_i(t_f)| \leq \epsilon \\ 0 & \text{if any } |x_i(t_f)| > \epsilon, \end{cases} \tag{12.66}$$

where Q is an estimate of the final time t_f at the beginning of the computer run.

By putting $\epsilon = 10^{-3}$, Bojkov and Luus obtained a minimum time $t_f = 4.19$ min for the 6-plate gas absorber and $t_f = 36.89$ min for the 20-plate gas absorber considered by Wong and Luus [18]. For the 20-plate gas absorber this final time is 34% less than the previously obtained value of $t_f = 56$ min.

12.6 References

[1] Bojkov, B. and Luus, R.: "Time-optimal control by iterative dynamic programming", *Ind. Eng. Chem. Res.* **33** (1994), 1486-1492.

[2] Bojkov, B. and Luus, R.: "Time optimal control of high dimensional systems by iterative dynamic programming", *Can. J. Chem. Eng.* **73** (1995), 380-390.

[3] Bojkov, B. and Luus, R.: "Optimal control of nonlinear systems with unspecified final times", *Chem. Eng. Sci.* **51** (1996), 905-919.

[4] Chung, T.S. and Wu, C.J.: "A computationally efficient numerical algorithm for the minimum-time control problem of continuous systems", *Automatica* **72** (1992), 841-847.

[5] Dadebo, S.A. and McAuley, K.B.: "Dynamic optimization of constrained chemical engineering problems using dynamic programming", *Comput. Chem. Eng.* **19** (1995), 513-525.

[6] Edgar, T.F. and Lapidus, L.: "The computation of optimal singular bang-bang control II. Nonlinear systems", *AIChE J.* **18** (1972), 780-785.

[7] Hull, T.E., Enright, W.D., and Jackson, K.R.: *User Guide to DVERK–a Subroutine for Solving Nonstiff ODE's.* Report 100 (1976), Department of Computer Science, University of Toronto, Canada.

[8] Lee, A.Y.: "Solving constrained minimum-time robot problems using sequential gradient restoration algorithm", *Optim. Control Appl. Methods* **13** (1992), 145-154.

[9] Luus, R.: "Optimal control by direct search on feedback gain matrix", *Chem. Eng. Sci.* **29** (1974), 1013-1017.

[10] Luus, R.: "Direct approach to time optimal control by iterative dynamic programming", *Proc. IASTED International Conference on Intelligent Systems and Control*, Halifax, Nova Scotia, Canada, June 1-4, 1998, pp. 121-125.

[11] Luus, R., Mekarapiruk, W., and Storey, C.: "Evaluation of penalty functions for optimal control", *Proc. of International Conference on Optimization Techniques and Applications (ICOTA '98)* (1998), Perth, Western Australia, July 1-3, 1998, Curtin Printing Services, pp. 724-731.

[12] Maurer, H. and Weigand, M.: "Numerical solution of a drug displacement problem with bounded state variables", *Optim. Control Appl. Methods* **13** (1992), 43-55.

[13] Moon, S.M. and VanLandingham, H.F.: "On the variational approach in a time optimal control problem", *Proc. IASTED International Conference CONTROL'97*, Cancun, Mexico, May 28-31, 1997, pp. 290-293.

[14] Nishida, N., Liu, Y.A., Lapidus, L., and Hiratsuka, S.: "An effective computational algorithm for suboptimal singular and/or bang-bang control", *AIChE J.* **22** (1976), 505-523.

[15] Rosen, O. and Luus, R.: "Evaluation of gradients for piecewise constant optimal control", *Comput. Chem. Eng.* **15** (1991), 273-281.

[16] Rosen, O., Imanudin, and Luus, R.: "Sensitivity of the final state specification for time-optimal control problems", *Int. J. Control* **45** (1987), 1371-1381.

[17] Weinreb, A. and Bryson, A.E.Jr.: "Optimal control of systems with hard control bounds", *IEEE Trans. Autom. Control* **AC-30** (1985), 1135-1138.

[18] Wong, K.T. and Luus, R.: "Model reduction of high-order multistage systems by the method of orthogonal collocation", *Can. J. Chem. Eng.* **58** (1980), 382-388.

Chapter 13

Nonseparable problems

13.1 Introduction

In using iterative dynamic programming to solve optimal control problems up to now, we have broken up the problem into a number of stages and assumed that the performance index could always be expressed explicitly in terms of the state variables at the last stage. This provided a scheme where we could proceed backwards in a systematic way, carrying out optimization at each stage. Suppose, however, that the performance index can not be expressed in terms of the variables at the last stage only. In other words, suppose the performance index is also a function of controls and variables at the other stages. Then we have a *nonseparable* optimal control problem. The goal of this chapter is to formulate this problem in general terms and to provide an approach that enables IDP to be used. Throughout this chapter we use difference equations rather than differential equations.

13.2 Problem formulation

Let us consider a nonlinear system described by a difference equation

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)), \quad k = 1, 2, \cdots, P-1, \tag{13.1}$$

where $\mathbf{x}$ is an $(n \times 1)$ state vector and $\mathbf{u}$ is an $(m \times 1)$ control vector. We consider problems where the initial state $\mathbf{x}(0)$ and the number of stages P are given. The performance index is a nonseparable general function of all the controls and states, which we write as

$$I = g[\mathbf{x}(0), \mathbf{x}(1), \cdots, \mathbf{x}(P), \mathbf{u}(0), \mathbf{u}(1), \cdots, \mathbf{u}(P-1)]. \tag{13.2}$$

The control variables at each stage are constrained by

$$\alpha_j \le u_j(k) \le \beta_j, \qquad j = 1, 2, \cdots, m; \;\; k = 1, 2, \cdots, P-1. \tag{13.3}$$

The optimal control problem is to find $\mathbf{u}(0), \mathbf{u}(1), \cdots, \mathbf{u}(P-1)$, such that the performance index in Eq. (13.2) is minimized.

The strategy for solving problems where the performance index is not an explicit function of the final state only was suggested by Luus and Tassone [3]. It was suggested to use for the variables upstream of calculations the best values available from the previous iteration. The downstream variables are obtained by simply solving Eq. (13.1). As the iterations proceed and the region is contracted, the effect of such approximations is expected to become negligibly small. In fact, using two examples, including the nonseparable problem of Li and Haimes [1], Luus [2] showed that this scheme works very well.

Here we illustrate the use of this scheme with two nonlinear nonseparable problems.

13.3 Examples

13.3.1 Example 1 − Luus-Tassone problem

Let us first consider the nonseparable optimization problem that we looked at in Chapter 2. This example was introduced by Luus and Tassone [3], and used to examine the convergence properties of IDP. The system is described by three difference equations:

$$x_1(k+1) = \frac{x_1(k)}{1 + 0.01u_1(k)(3 + u_2(k))} \tag{13.4}$$

$$x_2(k+1) = \frac{x_2(k) + u_1(k)x_1(k+1)}{1 + u_1(k)(1 + u_2(k))} \tag{13.5}$$

$$x_3(k+1) = \frac{x_3(k)}{1 + 0.01u_2(k)(1 + u_3(k))} \tag{13.6}$$

with the initial condition

$$\mathbf{x}(0) = [2 \quad 5 \quad 7]^T. \tag{13.7}$$

The control variables are constrained by

$$0 \le u_1(k) \le 4 \tag{13.8}$$

$$0 \le u_2(k) \le 4 \tag{13.9}$$

$$0 \le u_3(k) \le 0.5. \tag{13.10}$$

The performance index to be minimized is

$$\begin{aligned} I \;=\; & x_1^2(P) + x_2^2(P) + x_3^2(P) + [(\sum_{k=1}^{P} x_1^2(k-1) + x_2^2(k-1) + 2u_3^2(k-1)) \\ & (\sum_{k=1}^{P} x_3^2(k-1) + 2u_1^2(k-1) + 2u_2^2(k-1))]^{\frac{1}{2}} \end{aligned} \tag{13.11}$$

where P is the number of stages; Luus and Tassone [3] used $P = 20$, and Luus [2] used $P = 100$. Here we wish to examine the convergence properties for using IDP to solve this problem for three different values of P. In each case we take as the initial region size for the first pass $\mathbf{r}_{in} = [4 \;\; 4 \;\; 0.5]^T$ and the initial control policy $\mathbf{u}^{(0)} = [2 \quad 2 \quad 0.25]^T$. For all the runs we take the region contraction factor $\gamma = 0.85$, the region restoration factor $\eta = 0.5$, and allow 7 passes, each consisting of 30 iterations.

Case 1: $P = 20$

With the use of $P = 20$, the results in Table 13.1 show that the global optimum $I = 209.26937$ was readily obtained with the use of a single grid point. The convergence was quite regular, as is shown in Figure 13.1. Also the local optimum with $I = 209.27547$ was obtained three times. When the number of grid points was increased, the local optimum $I = 209.27320$ was obtained much more frequently. In the global optimum solution, there is a sudden change in the control variables at the 15th stage, as is shown in Figure 13.2.

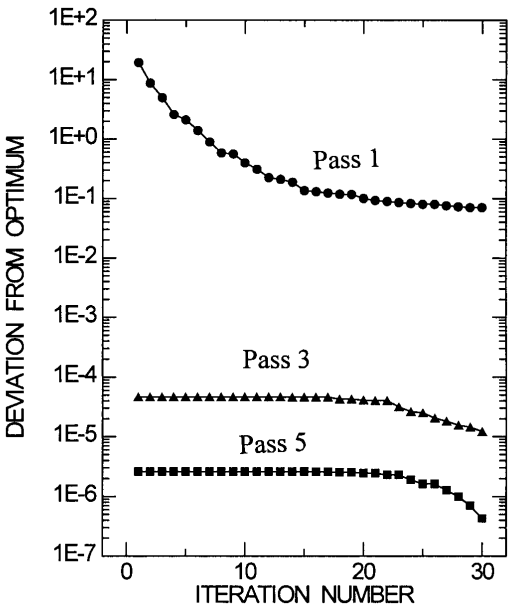

Figure 13.1: Convergence profile for P = 20, showing the deviations from $I = 209.2693696$ with $N = 1$ and $R = 7$

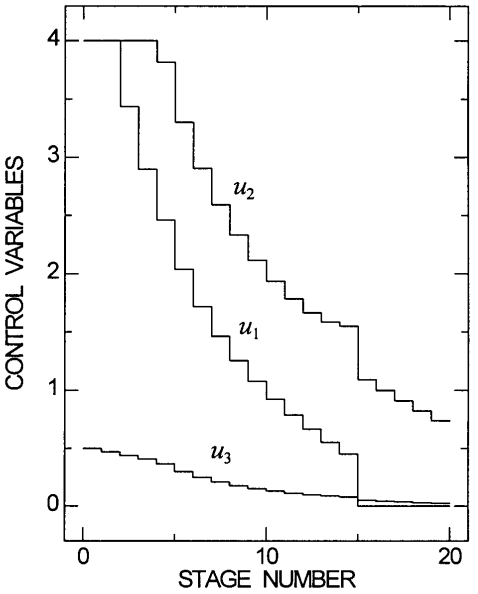

Figure 13.2: Optimal control policy for P = 20, showing the sudden change in control variables at stage 15

Case 2: $P = 50$

With $P = 50$, the global optimum $I = 240.91700$ could not be obtained with $N = 1$ grid point. However, with $N = 5$ over half of the runs resulted in the global optimum, as is shown in Table 13.2. Convergence profile as shown for $N = 3$ and $R = 7$ in Figure 13.3 is quite regular, although not monotonic. The optimal control policy, shown in Figure 13.4, shows a sudden change in the control variables at stage 36.

Case 3: $P = 100$

With $P = 100$, the global optimum $I = 258.33922$ was obtained twice with $N = 1$ grid point, as is shown in Table 13.3. It is interesting to note that with $N = 3$, the global optimum was not obtained for any of the eight runs. With $N = 5$, the global optimum was obtained in half of the runs. The convergence was found to be regular and systematic as is shown in Figure 13.5 with $N = 1$ and $R = 5$. The optimal control policy in Figure 13.6 shows a sudden change in the control variables at stage 69. It is the positioning of this sudden change that determines the value of the performance index. When the sudden change occurs at stage 68, then the local optimum gives $I = 258.33933$, and when the sudden change occurs at stage 70, the local optimum is $I = 258.33928$. In fact, the local optima in Table 13.3 can be arranged with respect to where the sudden change occurs, as is shown in Table 13.4. This is one way to assure that the global optimum has been obtained for this problem.

Table 13.1: Performance index after 7 passes, showing the effects of the number of grid points N and the number of allowable values for control R for $P = 20$

Random points	Performance index, I		
R	$N = 1$	$N = 3$	$N = 5$
3	**209.26937**	209.33307	209.27320
5	209.27547	209.27320	**209.26937**
7	**209.26937**	209.27320	209.27320
9	**209.26937**	**209.26937**	209.27320
11	**209.26937**	209.27320	209.27320
13	**209.26937**	209.27320	209.27320
15	209.27547	209.27320	209.27320
21	209.27547	209.27320	209.27320

Table 13.2: Performance index after 7 passes, showing the effects of the number of grid points N and the number of allowable values for control R for $P = 50$

Random points	Performance index, I		
R	$N = 1$	$N = 3$	$N = 5$
3	240.91808	240.91901	240.91896
5	240.91901	240.91767	240.91767
7	240.91734	**240.91700**	**240.91700**
9	240.92238	240.91767	240.91767
11	240.91734	240.91767	**240.91700**
13	240.91901	240.91767	**240.91700**
15	240.92238	240.91767	**240.91700**
21	240.91734	**240.91700**	**240.91700**

Table 13.3: Performance index after 7 passes, showing the effects of the number of grid points N and the number of allowable values for control R for $P = 100$

Random points	Performance index, I		
R	$N = 1$	$N = 3$	$N = 5$
3	258.33979	258.34023	258.33979
5	**258.33922**	258.33948	258.33979
7	258.34203	258.34017	**258.33922**
9	**258.33922**	258.33928	258.33964
11	258.34342	258.33928	**258.33922**
13	258.34096	258.33933	**258.33922**
15	258.34515	258.33928	258.33933
21	258.33933	258.33928	**258.33922**

Table 13.4: Classification of all the local optima in Table 13.3 with $P = 100$ according to the location of the sudden change in the control policy

Performance index	Location of the sudden change,
I	Stage number
258.34515	62
258.34342	63
258.34203	64
258.34096	65
258.34017	66
258.33964	67
258.33933	68
258.33922	**69**
258.33928	70
258.33948	71
258.34023	72

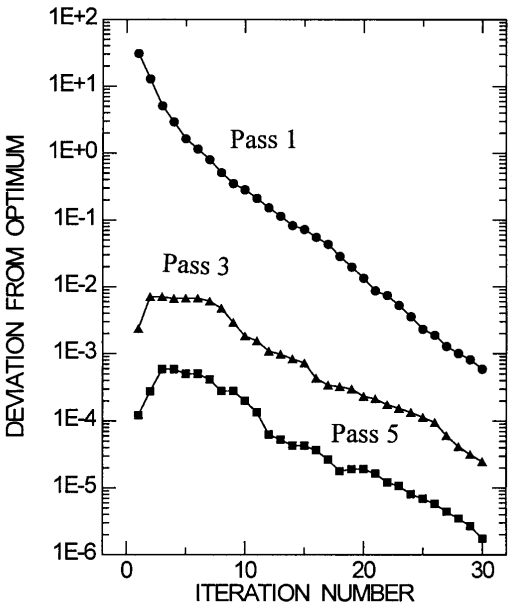

Figure 13.3: Convergence profile for P = 50, showing deviations from $I = 240.917000$ with $N = 3$ and $R = 7$

Table 13.5: Optimal solution to the Li-Haimes problem obtained after 4 passes, each consisting of 40 iterations, with the use of $R = 5$

Stage number k	control $u(k-1)$	state $x(k)$
1	-0.427163	0.314498
2	-0.098968	0.283373
3	-0.082382	0.200991

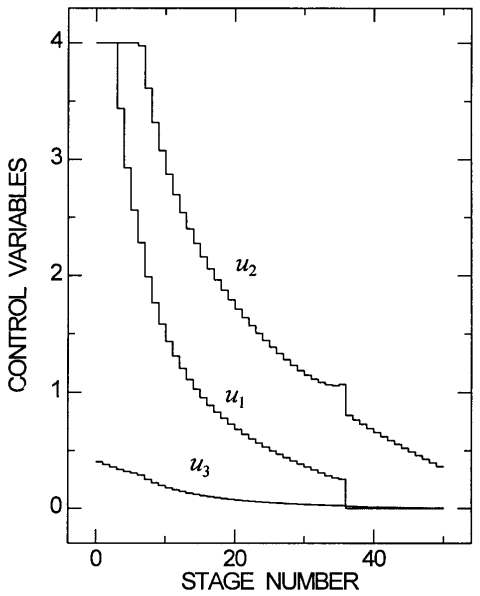

Figure 13.4: Optimal control policy for P = 50, showing the sudden changes in control variables at stage 36

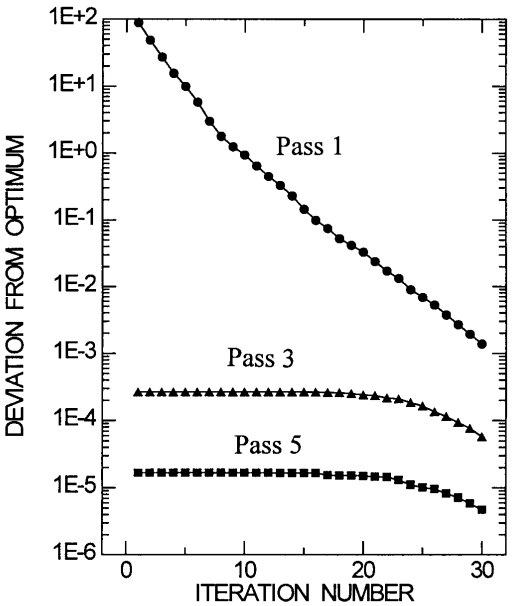

Figure 13.5: Convergence profile for P = 100, showing deviations from $I = 258.339218$ with $N = 1$ and $R = 5$

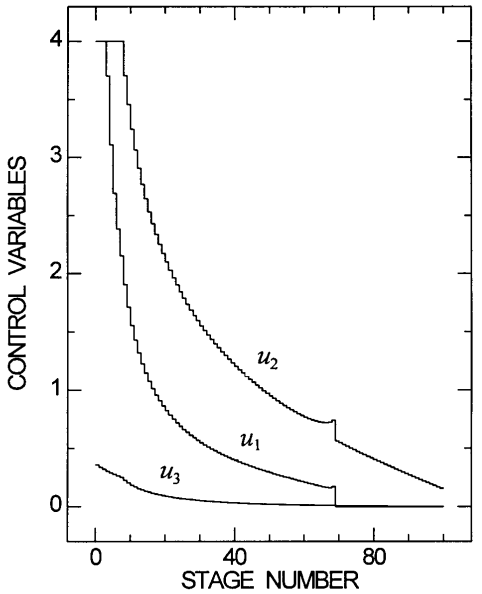

Figure 13.6: Optimal control policy for P = 100, showing sudden changes in control variables at stage 69

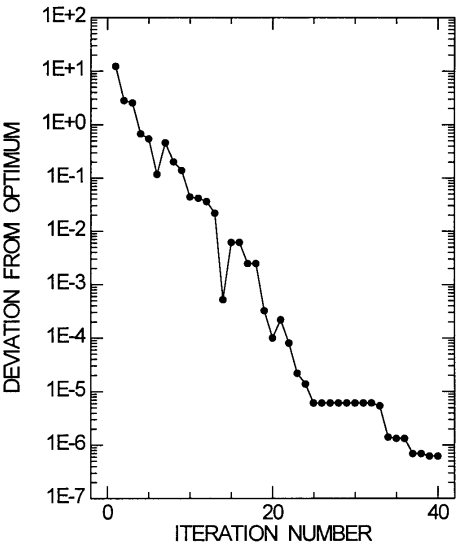

Figure 13.7: Deviation from the optimum value of the performance index $I = 1596.4796778$ during the first pass for the Li-Haimes problem

13.3.2 Example 2 – Li-Haimes problem

Let us consider the nonseparable optimization problem of Li and Haimes [1], where the system is described by the three scalar equations

$$x(1) = x(0)^{u(0)} \tag{13.12}$$

$$x(2) = [1 + u(1)]x(1) \tag{13.13}$$

$$x(3) = x(2) + u(2) \tag{13.14}$$

with the initial condition $x(0) = 15$. The control at each stage is constrained by

$$-1 \leq u(k) \leq 1, \qquad k = 0, 1, 2. \tag{13.15}$$

The performance index to be minimized is nonseparable in the form

$$I = [x^2(0)+x^2(1)+(2x^2(2)+x^2(3))\exp(x^2(1))][50+u^2(0)+(u^2(1)+u^2(2))\exp(u^2(0))]^{\frac{1}{2}}. \tag{13.16}$$

To solve this problem with IDP, we chose initial value for control of zero and the initial region size $r = 2$, $N = 1$, $R = 5$, and used 5 passes, each consisting of 40 iterations with $\gamma = 0.85$ and $\eta = 0.25$. The convergence was fast as is seen in Figure 13.7 where the convergence profile is given for the first pass. After 4 passes, convergence to the values given in Table 13.5 was obtained with $I = 1596.4796778$. The computation time for 5 passes was 0.5 s on a PentiumII/350 personal computer.

13.4 References

[1] Li, D. and Haimes, Y.Y.: "New approach for nonseparable dynamic programming problems", *JOTA* **66** (1990), 311-330.

[2] Luus, R.: "Application of iterative dynamic programming to optimal control of nonseparable problems", *Hung. J. Ind. Chem.* **25** (1997), 293-297.

[3] Luus R. and Tassone V.: "Optimal control of nonseparable problems by iterative dynamic programming", *Proc. 42nd Canadian Chem. Eng. Conf.*, October 18-21, 1992, Toronto, Canada, pp. 81-82.

Chapter 14

Sensitivity considerations

14.1 Introduction

Sensitivity is a measure of a change in one variable with respect to a change in another variable. For example, a very small variation in the E/R ratio, where E is the activation energy and R is the gas constant, has a large effect on the rate of a typical chemical reaction. In fact, Luus [4] showed that by using for the gas constant the value 2.00 cal/gm mole K instead of 1.9872, the expected rate of that particular reaction under consideration was increased by 30%. Such high sensitivities must be taken into account in the design of reactors.

However, there are situations where very low sensitivities create computational difficulties. For example, in parameter estimation, Kalogerakis and Luus [2] introduced the information index which relates the effect of the changes in parameters to changes in the measured output that is used to estimate the parameters, and therefore provides the sensitivity information. If the information index is very small in some interval of time, then the data in that time interval can not be used to get reliable values for the parameters no matter what parameter estimation procedure is used. For parameter estimation, it was shown that the use of the data in the time interval where the information index is high not only leads to reliable parameter estimates, but also avoids computational difficulties.

In optimal control, the problem of low sensitivity of the control policy on the performance index was addressed by Luus [6]. Although from the practical point of view, low sensitivity is good, since it means that even if the control policy differs somewhat from the optimal, the loss is not great. However, from the point of view of trying to establish the optimal control policy, low sensitivity creates difficulties, since the iterative procedure may be terminated prematurely, or some crude approximation may be inadvertently accepted as the optimal one when the performance index is within reasonable proximity (such as 0.01%) of the optimum. Often the nonoptimal control policies exhibit highly oscillatory sections that are quite smooth in the optimal control policy. For example, the control policy obtained with accelerated simulated

annealing by Yang and Wu [10] for maximizing the yield of the fed-batch reactor of Park and Ramirez [8] that we considered in Chapter 9 exhibited a highly fluctuating initial portion of the control policy, as is shown in Figure 14.1. The corresponding performance index of 32.68319 is within 0.01% of the optimum value of 32.68602. The optimal control policy is quite smooth, as is seen in Figure 14.2. It was shown by Luus [6] that even the control policy giving $I = 32.68601$ is quite oscillatory. Often, very high oscillations have minimal effect on the performance index [4]. However, the performance index is quite sensitive to the time of switching; it was shown by Luus and Hennessy [7] that a better value of the performance index, namely $I = 32.68687$, is obtained by using $P = 45$ stages rather than 80, because the switching times are very slightly different. As we saw in Chapter 9 by using flexible stage lengths, we can improve the value further to $I = 32.6939$ with only 20 time stages, because the switching times can be obtained very accurately by having stages of flexible length.

Although there is very low sensitivity with respect to changes in control policy, the performance index is very sensitive to the choice of the final time, as was pointed out in [5], and illustrated in Chapter 9.

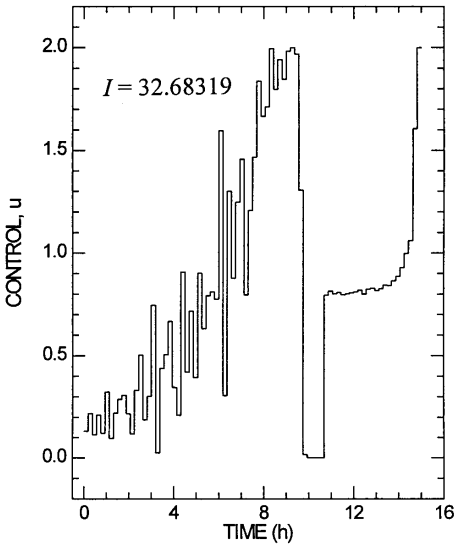

Figure 14.1: Control policy for the fed-batch reactor of Park and Ramirez [8], using 80 stages of equal length, obtained by accelerated simulated annealing by Yang and Wu [10]

Recently, Tholudur and Ramirez [9] reported getting a highly oscillatory control policy with the use of IDP, and suggested using filtering to reduce such unwanted oscillations. As we will show, the oscillatory behavior is not present in the optimal control policy. Therefore, in this chapter we wish to consider their example to illustrate the low sensitivity and to show that the optimal control policy is indeed quite

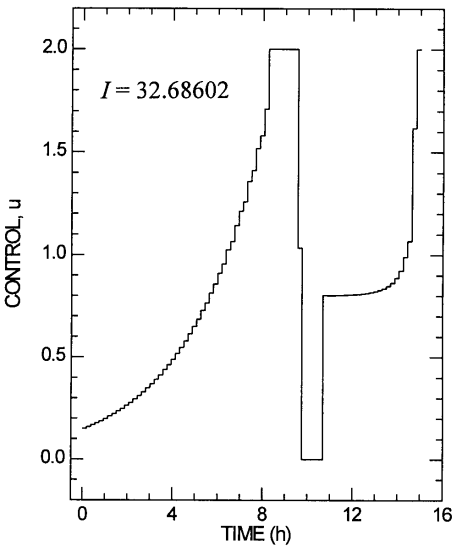

Figure 14.2: Control policy for the fed-batch reactor of Park and Ramirez [8], using 80 stages of equal length, obtained by IDP

smooth. This fed-batch reactor is interesting from the computational point of view, since there are two feed streams, and one would expect that to get a reasonably accurate answer a large number of time stages should be used. We will see that the low sensitivity allows us to use a small number of stages to get an accurate value for the profit function.

14.2 Example: Lee-Ramirez bioreactor

The bioreactor of Lee and Ramirez [3] was modified slightly by Tholudur and Ramirez [9] to give the following set of equations

$$\frac{dx_1}{dt} = u_1 + u_2 \tag{14.1}$$

$$\frac{dx_2}{dt} = g_1 x_2 - \frac{u_1 + u_2}{x_1} x_2 \tag{14.2}$$

$$\frac{dx_3}{dt} = \frac{100u_1}{x_1} - \frac{u_1 + u_2}{x_1} x_3 - \frac{g_1}{0.51} x_2 \tag{14.3}$$

$$\frac{dx_4}{dt} = R_{fp} x_2 - \frac{u_1 + u_2}{x_1} x_4 \tag{14.4}$$

$$\frac{dx_5}{dt} = \frac{4u_2}{x_1} - \frac{u_1 + u_2}{x_1} x_5 \tag{14.5}$$

$$\frac{dx_6}{dt} = -k_1 x_6 \tag{14.6}$$

$$\frac{dx_7}{dt} = k_2(1 - x_7) \tag{14.7}$$

$$\frac{dx_8}{dt} = u_2 \tag{14.8}$$

where

$$g_1 = (\frac{x_3}{14.35 + x_3(1 + x_3/111.5)})[x_6 + \frac{0.22x_7}{0.22 + x_5}] \tag{14.9}$$

$$R_{fp} = (\frac{0.233x_3}{14.35 + x_3(1 + x_3/111.5)})(\frac{0.0005 + x_5}{0.022 + x_5}) \tag{14.10}$$

$$k_1 = k_2 = \frac{0.09x_5}{0.034 + x_5}. \tag{14.11}$$

The controls are bounded by

$$0 \le u_j \le 1, \qquad j = 1, 2. \tag{14.12}$$

The performance index to be maximized is the profitability of the process described by the performance index

$$I = x_1(t_f)x_4(t_f) - 5x_8(t_f) \tag{14.13}$$

where the final time t_f is specified as 10 h.

The initial state is

$$\mathbf{x}(0) = [\ 1 \quad 0.1 \quad 40 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]^T. \tag{14.14}$$

The two control variables are the glucose feed rate u_1 and the inducer feed rate u_2. The state variables are the reactor volume x_1, the cell density x_2, the nutrient concentration x_3, the foreign protein concentration x_4, the inducer concentration x_5, the inducer shock factor x_6, the inducer recovery factor x_7, and the amount of inducer used x_8.

14.2.1 Solution by IDP

To solve this optimal control problem by iterative dynamic programming, we took initially $P = 15$ time stages of equal length, the initial control $u_1^{(0)} = u_2^{(0)} = 0.0$, initial region size $r_1^{(0)} = r_2^{(0)} = 0.5$, reduction factor $\gamma = 0.95$, $N = 1$ grid point, and used a multi-pass method with region restoration factor $\eta = 0.90$. We used 10 iterations in each pass, and allowed 100 passes. For integration, DVERK [1] with error tolerance of 10^{-7} was used. By taking 15 stages of constant length, convergence to $I = 5.56773$ with the use of $R = 15$ was systematic as is shown in Figure 14.3. Even with $R = 7$ we got good convergence rate, but $R = 5$ gave inadequate convergence rate with

the parameters that were used. The computation time on a PentiumII/350 personal computer with $R = 15$ for 100 passes was 139 s. Thus, the computational effort is quite small.

The resulting optimal control policy is given in Figure 14.4. By increasing the number of time stages, the performance index was increased, but very slightly, as is shown in Figures 14.5-14.7. The refined control policy with the use of $P = 100$ gives the profitability function $I = 5.56789$ which is only 0.003% better than $I = 5.56773$ obtained with the use of 15 stages. Therefore, this problem exhibits very low sensitivity, and an accurate optimal control policy is difficult to establish. Whether or not that small peak around $t = 8$h is included makes very little difference in the value of the performance index.

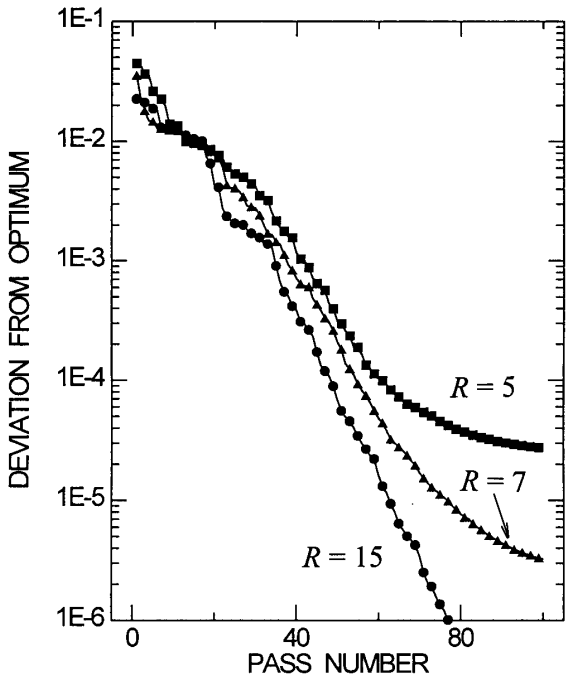

Figure 14.3: Convergence of IDP, showing the effect of the number of allowable values for control R with $N = 1$ and $P = 15$ stages of equal length; the deviations are from $I = 5.56772556$

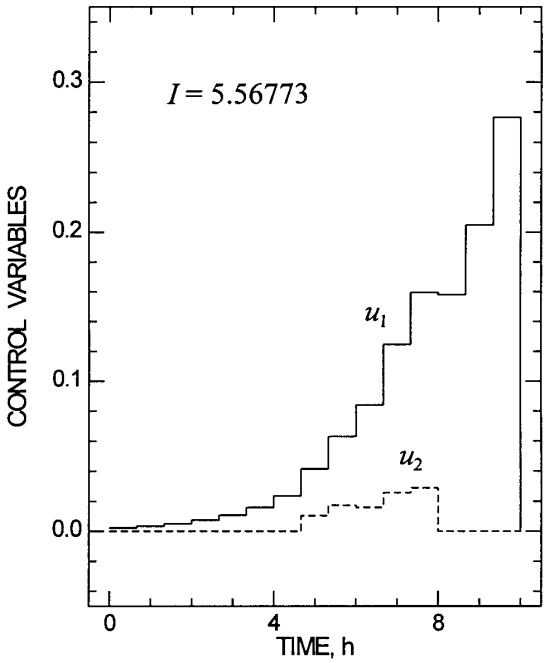

Figure 14.4: Optimal control policy with $P = 15$ stages of equal length, giving $I = 5.56773$

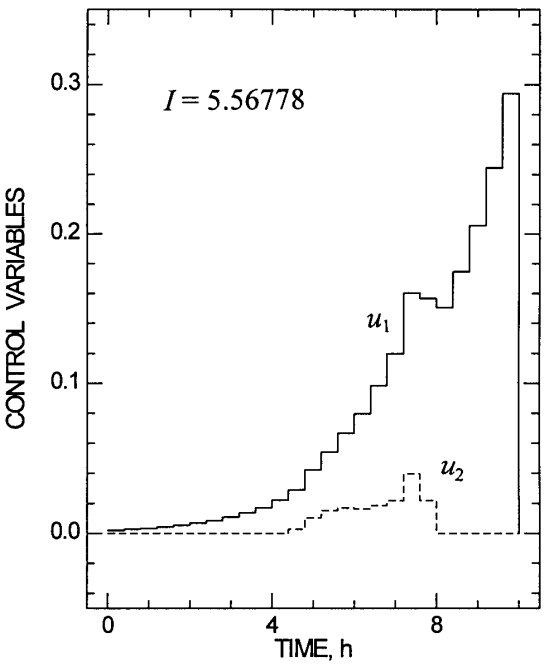

Figure 14.5: Optimal control policy with $P = 25$ stages of equal length, giving $I = 5.56778$

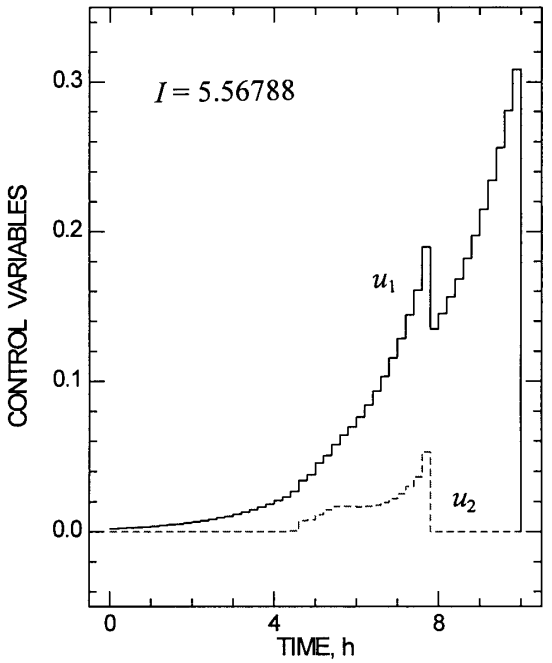

Figure 14.6: Optimal control policy with $P = 50$ stages of equal length, giving $I = 5.56788$

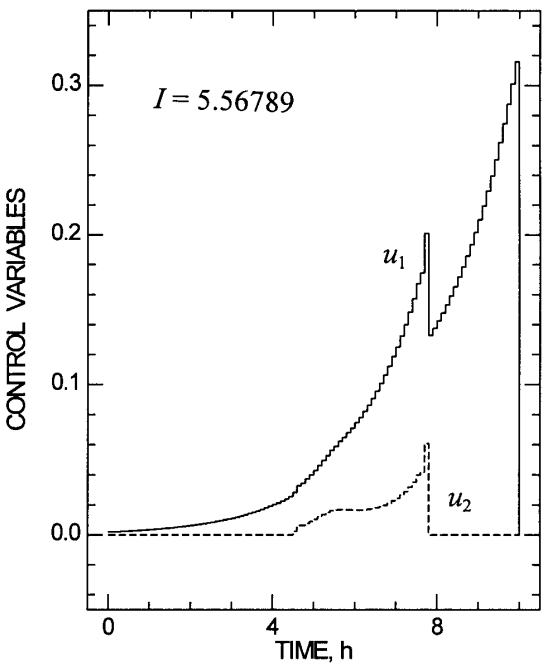

Figure 14.7: Optimal control policy with $P = 100$ stages of equal length, giving $I = 5.56789$

14.3 References

[1] Hull, T.E., Enright, W.D., and Jackson, K.R.: *User Guide to DVERK–a Subroutine for Solving Nonstiff ODE's*. Report 100 (1976), Department of Computer Science, University of Toronto, Canada.

[2] Kalogerakis, N. and Luus, R.: "Improvement of Gauss-Newton method for parameter estimation through the use of information index", *Ind. Eng. Chem. Fundam.* **22** (1983), 436-445.

[3] Lee, J. and Ramirez, W.F.: "Optimal fed-batch control of induced foreign protein production by recombinant bacteria", *AIChE J.* **40** (1994), 899-907.

[4] Luus, R.: "Some important considerations in reactor design", *AIChE J.* **18** (1972), 438-439.

[5] Luus, R.: "Effect of the choice of final time in optimal control of nonlinear systems", *Can. J. Chem. Eng.* **69** (1991), 144-151.

[6] Luus, R.: "Sensitivity of control policy on yield of a fed-batch reactor", *Proc. IASTED Int. Conf. on Modelling and Simulation*, Pittsburgh, PA, April 27-29, 1995, pp. 224-226.

[7] Luus, R. and Hennessy, D.: "Optimization of fed-batch reactors by the Luus-Jaakola optimization procedure", *Ind. Eng. Chem. Res.* **38** (1999), 1948-1955.

[8] Park, S. and Ramirez, W.F.: "Optimal production of secreted protein in fed-batch reactors", *AIChE J.* **34** (1988), 1550-1558.

[9] Tholudur, A. and Ramirez, W.F.: "Obtaining smoother singular arc policies using a modified iterative dynamic programming algorithm", *Int. J. Control* **68** (1997), 1115-1128.

[10] Yang, R.L. and Wu, C.P.: "Global optimal control by accelerated simulated annealing", Paper presented at the First Asian Control Conference, Tokyo, Japan, 1994.

Chapter 15

Toward practical optimal control

15.1 Introduction

In establishing the optimal control of chemical reactors, the control variable used most frequently is the temperature of the reacting mixture. Although the optimal control policy, consisting of the temperature profile, is useful for understanding the reactor behavior, it does not lend itself to practical use, since the temperature can not be changed directly. As such, temperature is not a practical control variable. The temperature of the reactor can be changed by withdrawing heat, or adding heat by means of flowing coolant or heating fluid around the reactor. Then the flow rates of these streams become the real practical control variables which can be manipulated to maximize the appropriate performance criterion.

Using flow rates to change the reactor temperature makes the optimal control determination much more difficult. When faced with the problem of determining the optimal flow rates of heating and cooling fluids, so that the yield of a desired product in a batch reactor is maximized, we encounter two additional differential equations to handle the heat balances for the reactor and the surrounding jacket. Furthermore, the problem is more difficult, since the constraints on the reactor temperature now become state constraints, which are more difficult to handle than simple constraints on the control. Such practical aspects of optimal control were considered by Luus and Okongwu [5], who showed that the speed of our present-day computers makes the on-line optimal control of chemical reactors now possible. Even if there is uncertainty in the heat transfer coefficient, the optimal control problem can be solved sufficiently fast to enable on-line application. To illustrate these ideas, we consider the oil shale pyrolysis problem that we considered in Chapter 9. This time, however, instead of simply determining the optimal temperature profile, we consider the problem of determining the optimal flow rates of coolant and heating fluid.

15.2 Optimal control of oil shale pyrolysis

Let us consider the oil shale pyrolysis problem used for optimal control studies by Wen and Yen [7] and by Luus [1], and for which we determined the optimal temperature profile in Chapter 9. The equations describing the chemical reactions are

$$\frac{dx_1}{dt} = -k_1 x_1 - (k_3 + k_4 + k_5) x_1 x_2 \tag{15.1}$$

$$\frac{dx_2}{dt} = k_1 x_1 - k_2 x_2 + k_3 x_1 x_2 \tag{15.2}$$

$$\frac{dx_3}{dt} = k_2 x_2 + k_4 x_1 x_2 \tag{15.3}$$

$$\frac{dx_4}{dt} = k_5 x_1 x_2 \tag{15.4}$$

where the rate constants are given by

$$k_i = \exp(a_i - \frac{b_i}{T}), \quad i = 1, 2, \cdots, 5, \tag{15.5}$$

and the kinetic parameters a_i and b_i are given in Table 9.2. The residence time or batch time t_f is not specified, but should be approximately 9.3 min. The problem is to maximize the yield of bitumen $x_2(t_f)$ when starting with $x_1(0) = 1$ and the rest of the concentrations equal to zero. By using the control policy in Figure 9.9, we showed that we could get $x_2(t_f) = 0.35382$, which is 1.8% better than the yield of 0.34763 obtained by the best isothermal policy of $T = 712.5$ K. Now let us examine this problem by using as control variables the flow rates of coolant and heating fluids.

By using a reactor volume of 1.2 m^3 and realistic parameters, Luus and Okongwu [5] obtained expressions for the heat balances for the reactor temperature $T = x_5$ and the jacket temperature x_6:

$$\frac{dx_5}{dt} = \frac{[602.4 k_1 x_1 - 0.833(UA/\rho)(x_5 - x_6)]}{C_p} \tag{15.6}$$

$$\frac{dx_6}{dt} = u_1(873 - x_6) + u_2(373 - x_6) + 0.01357 UA(x_5 - x_6) \tag{15.7}$$

where the slurry density is given by

$$\rho = 2139 x_1 + 760 x_2 + 560 x_3 + 1800 x_4, \tag{15.8}$$

and the heat capacity is given by

$$C_p = 1.16 x_1 + 0.827(1 - x_1) + [3.4 x_1 + 0.92(1 - x_1)](x_5 - 298) \times 10^{-3}. \tag{15.9}$$

The product of the heat transfer coefficient U and the heat transfer area A, namely UA, is kept as a parameter and allowed to be in the range 400 to 1500 kJ/min K.

The hot steam flow rate u_1 and the "cold" steam flow rate u_2 (m^3/min) are bounded by

$$0 \leq u_j \leq 5, \qquad j = 1, 2. \tag{15.10}$$

To avoid simultaneous flows of heating steam and cooling steam, we introduce a heat flow term w such that

$$u_1 = w, \quad u_2 = 0, \quad \text{if } w \geq 0 \tag{15.11}$$

and

$$u_1 = 0, \quad u_2 = -w, \quad \text{if } w < 0 \tag{15.12}$$

with the bounds

$$-5 \leq w \leq 5, \tag{15.13}$$

and carry out optimization by using the scalar w as the control variable. Once w is determined, the flow rates u_1 and u_2 are immediately available.

To handle the upper and lower constraints on the reactor temperature, we introduce the state constraint variables x_7 and x_8, as outlined in Section 11.3.2, through the differential equations

$$\frac{dx_7}{dt} = \begin{cases} 0 & \text{if } x_5 \geq 698.15 \\ 698.15 - x_5 & \text{if } x_5 < 698.15 \end{cases} \tag{15.14}$$

and

$$\frac{dx_8}{dt} = \begin{cases} 0 & \text{if } x_5 \leq 748.15 \\ x_5 - 748.15 & \text{if } x_5 > 748.15 \end{cases} \tag{15.15}$$

with zero initial conditions. Thus if the constraint $698.15 \leq x_5 \leq 748.15$ is not violated, the state constraint variables remain zero. If a violation occurs at the lower boundary, then x_7 becomes positive, and if the violation occurs at the upper boundary, then the variable x_8 becomes positive.

We now have 8 state variables with the initial condition

$$\mathbf{x}(0) = [\ 1 \quad 0 \quad 0 \quad 0 \quad 698.15 \quad 698.15 \quad 0 \quad 0\ ]^T \tag{15.16}$$

where it is assumed that at $t = 0$, both the reactor and the jacket surrounding the reactor are at the temperature 698.15. Since the goal is to maximize the bitumen yield

$$I = x_2(t_f), \tag{15.17}$$

we introduce the augmented performance index

$$J = x_2(t_f) - \theta[x_7(t_f) + x_8(t_f)] \tag{15.18}$$

where θ is a positive penalty function factor.

By taking $P = 10$ stages of equal length of 0.93 min, $UA = 1200$, initial value for w of zero, initial region size of 5, $\theta = 0.01$, $R = 15$, $\gamma = 0.90$, $\eta = 0.85$, and allowing 15

Table 15.1: Performance index I after 15 passes with the use of $P = 10$ stages, each of length 0.93 min

Number of grid points	Performance index	
N	$R = 15$	$R = 25$
1	0.34962	0.34924
5	0.35164	0.35037
9	0.34872	0.35211
13	0.35184	0.35168
17	0.35173	0.35165
21	0.35208	0.35173
41	0.35210	0.35211

Table 15.2: Performance index I and the final time t_f after 20 passes with the use of $P = 10$ stages of flexible stage lengths

Number of grid points	Performance index	Final time
N	I	t_f
1	0.35086	8.504
3	0.35165	9.626
5	0.35156	9.435
7	0.35180	9.337
9	0.35193	9.254
11	0.35197	9.459
13	0.35211	9.157
17	0.35218	9.173
21	0.35213	9.216

passes, each consisting of 20 iterations, we found that the results were dependent on the number of grid points that were used. In Table 15.1 it is clear that more than five grid points are required to get convergence to I greater than 0.3517 from the chosen starting policy. The effect of increasing the number of randomly chosen candidates from $R = 15$ to $R = 25$ is not great, but the use of the larger number of points enables the control policy yielding $I = 0.35211$ to be obtained twice. In each case, at the final time both x_7 and x_8 were zero, showing that the temperature of the reactor did not violate the bounds. The corresponding control policy is shown in Figure 15.1. In the first half of the time interval, heating is used, then rapid cooling is initiated, followed by a section of relatively little cooling to keep the reactor temperature at its lower bound.

Since the time of switching from heating to cooling is important, a set of runs

Table 15.3: Performance index I and the final time t_f obtained with the use of $P = 5$ stages of flexible lengths as a function of the heat transfer term

Heat transfer term UA	Performance index I	Final time t_f
1500	0.35228	9.144
1400	0.35226	9.165
1300	0.35224	9.187
1200	0.35222	9.206
1100	0.35219	9.247
1000	0.35216	9.258
900	0.35212	9.341
800	0.35207	9.508
700	0.35201	9.535
600	0.35192	9.646
500	0.35183	9.764
400	0.35169	9.946

was performed with the use of flexible stage lengths. Using the same parameters as before with $R = 25$, and choosing the initial region size for the stage lengths of 0.1, but keeping the stage lengths constant for the first five passes, we found that the maximum value for I was 0.35218, obtained with the use of 17 grid points. The results were quite erratic, as can be seen in Table 15.2, especially with respect to the final time that was obtained. It was interesting to note that in each case, the resulting control policy indicated full heating ($w = 5$) for the first few stages. When the control policy in Figure 15.1 was used as the initial control policy, with $N = 13$ grid points, convergence was readily obtained. The control for the first stage became 5, and the resulting optimal control policy consisted of 5 time stages, as is shown in Figure 15.2, giving the optimal value $J = I = 0.35222$ with $t_f = 9.206$ min.

By using the optimal control policy for $UA = 1200$, convergence for $UA = 1100$ and $UA = 1300$ was readily obtained with $N = 13$ grid points. Using such a continuation procedure, the optimal control policies for other values for UA were obtained, as is shown in Table 15.3. It is interesting to note the regular decrease in t_f and increase in I as UA is increased. This regularity of the increase in I with UA is seen more clearly in Figure 15.3. In each case, the control policy consisted of 5 time stages: the first stage involved full heating, second stage involved full cooling, and the remaining three stages involved intermediate cooling, with the last stage very close to zero. As UA is increased, the first two stages became shorter in length, as is expected. Comparison of the optimal control policy for $UA = 400$ in Figure 15.4, with the optimal control policy for $UA = 1200$ given in Figure 15.2 earlier, shows this effect.

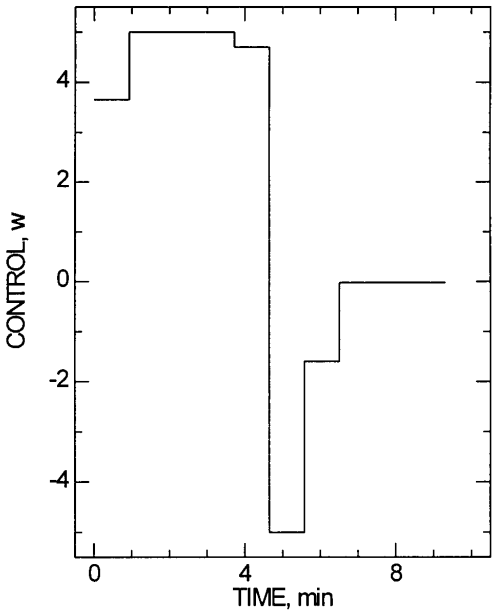

Figure 15.1: Optimal control policy obtained with $P = 10$ stages, each of length 0.93 min, giving $I = 0.35211$

The reactor and jacket temperature profiles for $UA = 1500$, giving $I = 0.35228$, are shown in Figure 15.5. As can be seen, the reactor temperature does not violate the upper and lower bounds that were specified. This yield is only 0.4% below the theoretical optimum of 0.35382 and is 1.3% above the best isothermal yield of 0.34763 mol/L. Even with $UA = 400$, the yield of 0.35169 is 1.2% above the yield expected from the best isothermal operation.

Suppose that while applying the optimal control policy for some value for UA, from measurements in the first two minutes it becomes clear that the heat transfer term UA is different from the one used. This creates no problems, since for all the control policies the initial portion is $w = 5$; so, the only change is the length of time that this control policy is kept, followed by the recalculated control policy for the remaining stages. Since a good initial control policy is already available, such a calculation can be done with a single grid point and no more than 2 or 3 passes. Such a calculation can be performed in less than 1 minute of computation time, which gives adequate time for implementation of the optimal control policy "on-line". The computational aspects with stages of equal length have been illustrated by Luus and Okongwu [5].

15.3 Future directions

Although the inclusion of additional differential equations to handle the flow rates as control increases the computational effort quite significantly, the computation time

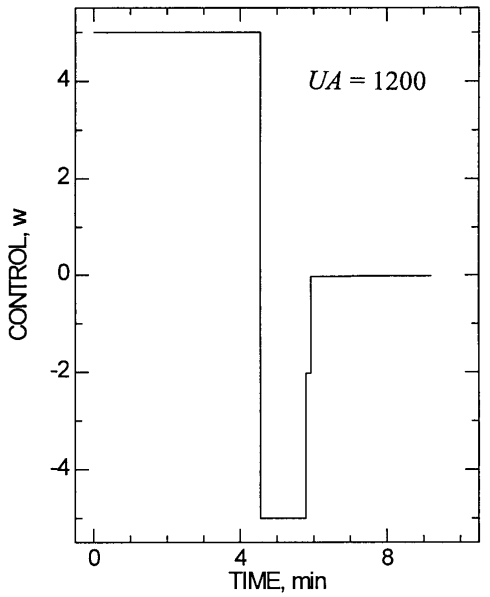

Figure 15.2: Optimal control policy obtained with $P = 5$ stages of varying length, giving $I = 0.35222$

with IDP is not excessive. The effect of the heat transfer coefficient is quite important, and, therefore, accurate modelling is required when implementation of optimal control is considered in practice. However, even with inaccurate modelling, the parameters can be updated while running the reactor. With the updated parameters reoptimization can be carried out over the rest of the batch time even with complex systems.

Since personal computers now are very fast and will be even faster in the future, the optimal control approach as outlined here appears to be quite practical. Although the resulting control policy is open loop in nature, during the course of reaction the parameters can be updated and the optimization problem can be solved repeatedly. This means that appropriate changes can be made during the operation of the reactor. In essence then the optimal control will be feedback in nature, where the control action is determined from the knowledge of the state of the system at any particular time.

Iterative dynamic programming is well suited for solving complex optimal control problems, such as the ones encountered by chemical engineers [2]. The main advantage lies in its robustness in obtaining the global optimum. Although the possibility of getting a local optimum exists, changing the parameters in IDP such as initial region size, number of grid points and the number of randomly chosen points per iteration usually allows one to obtain the global optimum. It must be remembered that IDP is not an algorithm that can be applied directly to solve any optimal control problem; IDP is an approach in applying the principle of optimality in dynamic programming in an iterative manner to obtain the optimal control policy to complex systems.

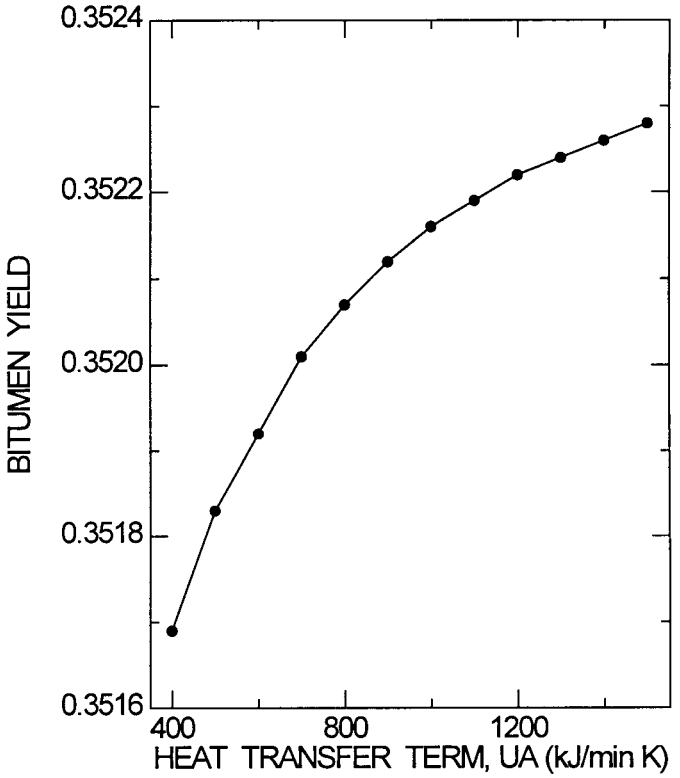

Figure 15.3: Effect of the heat transfer term UA on the performance index

Several algorithms, to illustrate the procedure, are given in the Appendix D. There is no doubt that readers will be able to improve the convergence characteristics of the procedure for many classes of problems by developing more efficient algorithms. Just as with the LJ optimization procedure where a systematic procedure was developed to determine region sizes for the variables at the beginning of each pass [3, 4], such schemes can be introduced into IDP as well. In fact, by using such an adaptive region size determination in IDP, Mekarapiruk and Luus [6] showed that, for two typical chemical engineering optimal control problems, the convergence rate is increased and there is a factor of two improvement in the computational efficiency. There is much room for research, especially in the areas of handling state constraints and dealing with systems that exhibit discontinuities. Further developments in the improvement of algorithms used to apply IDP to very complex systems are continuing.

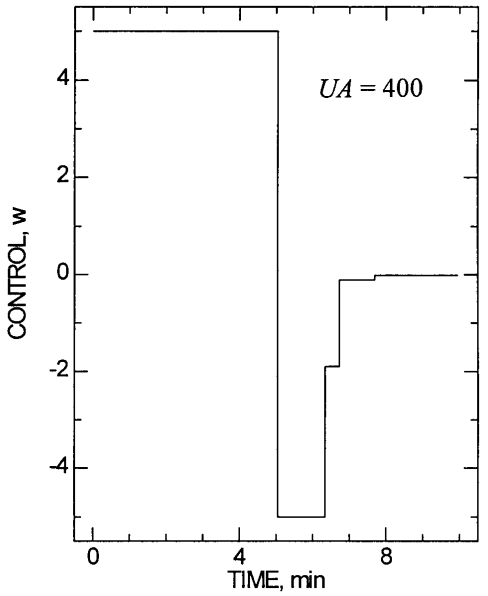

Figure 15.4: Optimal control policy obtained with $P = 5$ stages of varying length with $UA = 400$ kJ/min K, giving $I = 0.35169$

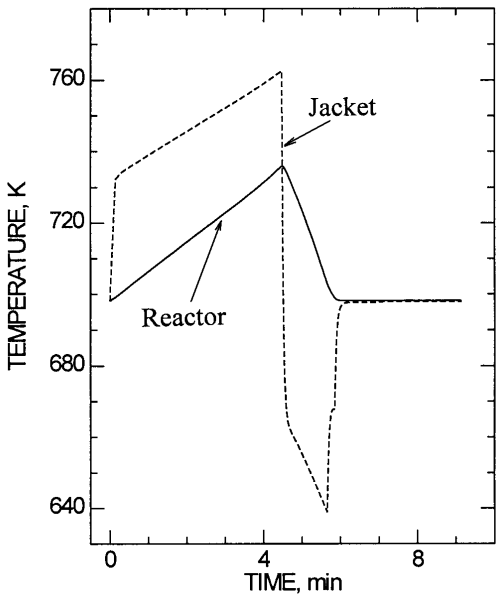

Figure 15.5: Temperature profiles for the reactor and the jacket for $UA = 1500$ kJ/min K, giving $I = 0.35228$

15.4 References

[1] Luus, R.: "On the optimization of oil shale pyrolysis", *Chem. Eng. Sci.* **33** (1978), 1403-1404.

[2] Luus, R.: "Optimal control of batch reactors by iterative dynamic programming", *J. Process Control* **4** (1994), 217-226.

[3] Luus, R.: "Determination of the region sizes for LJ optimization procedure", *Hung. J. Ind. Chem.* **26** (1998), 281-286.

[4] Luus, R. and Hennessy, D.: "Optimization of fed-batch reactors by the Luus-Jaakola optimization procedure", *Ind. Eng. Chem. Res.* **38** (1999), 1948-1955.

[5] Luus, R. and Okongwu, O.N.: "Towards practical optimal control of batch reactors", *Chem. Eng. J.* **75** (1999), 1-9.

[6] Mekarapiruk, W. and Luus, R.: "Iterative dynamic programming with adaptive scheme for region size determination", *Hung. J. Ind. Chem.* **27** (1999), 235-240.

[7] Wen, C.S. and Yen, T.F.: "Optimization of oil shale pyrolysis", *Chem. Eng. Sci.* **32** (1977), 346-349.

Appendix A

Nonlinear algebraic equation solver

Here we include the entire program, written in FORTRAN, to solve a set of nonlinear algebraic equations as entered into the FUN subroutine. The program includes the subroutine RLEQ for the Gaussian elimination.

A.1 Program listing

```
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION A(20,20),ZZ(20),R(20),X(20),F(20),DX(20),DFDX(20,20)
      DIMENSION Z(20),G(20),Y(20),REG(20),XP(20),XS(20),FACT(20)
      DIMENSION GG(20),YY(20),ZB(20)
      OPEN (UNIT=3,FILE='EXAMP106.OUT')
      CALL GETTIM(IH,IM,IS,I100)
      TIMEI = I100+100*IS+6000*IM+360000*IH
      N = 2
      NV = 3
      NIT = 25
      X(1) = 0.4D0
      X(2) = 1600.0D0
      WRITE(3,51) (X(I),I=1,N)
51    FORMAT(1X,'INTL VALUES OF X ARE',3F12.5)
      DO 41 IT = 1,NIT
      CALL FUN(F,X,N)
      TEST = 0.0D0
      DO 43 I=1,N
43    TEST = TEST + DABS(F(I))
      IF(TEST.LT.1.0D-12) GO TO 42
      DO 52 I=1,N
      DX(I) = 1.0D-6*X(I)
      IF(DABS(X(I)).LT.1.0D-6) DX(I) = 1.0D-6
      FACT(I) = 0.5D0/DX(I)
      ZB(I) = X(I)-DX(I)
52    Z(I) = X(I)+DX(I)
      DO 61 I=1,N
      DO 62 K=1,N
      Y(K) = X(K)
      YY(K) = X(K)
      IF(K.EQ.I) Y(K) = Z(K)
      IF(K.EQ.I) YY(K) = ZB(K)
62    CONTINUE
      CALL FUN(G,Y,N)
      CALL FUN(GG,YY,N)
      DO 63 J=1,N
      DFDX(J,I) = (G(J)-GG(J))*FACT(I)
63    CONTINUE
61    CONTINUE
      DO 31 J=1,N
31    R(J) = - F(J)
```

```
      WRITE(3,55) IT,(F(I),I=1,N),TEST
      WRITE(6,55) IT,(F(I),I=1,N),TEST
55    FORMAT(1X,'AT ITN ',I5,2X,'F =',4D13.5)
      CALL RLEQ(DFDX,R,ZZ,N)
29    FORMAT(10X,' X =',3D15.7)
      DO 32 J=1,N
32    X(J) = X(J) + ZZ(J)
      WRITE(3,29) (X(I),I=1,N)
      WRITE(6,29) (X(I),I=1,N)
41    CONTINUE
42    CONTINUE
      CALL FUN(F,X,N)
      WRITE(3,161) IT, TEST
161   FORMAT(1X,' ITN =',I3,' NORM=',D13.5)
      WRITE(3,59) (X(I),I=1,NV)
59    FORMAT(1X,' X ARE =',4F12.6)
      CALL GETTIM(IH,IM,IS,I100)
      TIMEF = I100+100*IS+6000*IM+360000*IH
      CPU = (TIMEF-TIMEI)*1.0D-2
      WRITE(3,21) CPU
      WRITE(6,21) CPU
21    FORMAT(1X,'CPU TIME =',F10.5,' SECONDS')
      STOP
      END
      SUBROUTINE FUN(F,X,N)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION F(20),X(20)
      X(3) = DEXP(4.23D4/X(2) - 24.2D0 + 0.17D0*DLOG(X(2)))
      F(1) = (0.91D0 - 0.5D0*X(1))/(9.1D0 - 0.5D0*X(1)) -
     1  (X(1)/(1.0D0 - X(1)))**2/X(3)
      F(2) = X(2)*(1.84D0*X(1) + 77.3D0) - 4.326D4*X(1) - 1.05128D5
      RETURN
      END
      SUBROUTINE RLEQ(A,D,X,N)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION A(20,20),D(20),X(20)
      NQ = N - 1
      DO 5 K=1,NQ
      DEN = 1.0D0/A(K,K)
      D(K) = DEN*D(K)
      KK = K + 1
      DO 6 J = KK,N
6     A(K,J) = DEN*A(K,J)
      DO 7 I=KK,N
      D(I) = D(I) - D(K)*A(I,K)
      DO 7 J=KK,N
7     A(I,J) = A(I,J) - A(K,J)*A(I,K)
5     CONTINUE
      X(N) = D(N)/A(N,N)
      DO 8 I=1,NQ
      K = N - I
      X(K) = D(K)
      DO 9 J=1,I
      L = K+J
9     X(K) = X(K) - A(K,L)*X(L)
 8    CONTINUE
      RETURN
      END
```

A.2 Output of the program

```
INTL VALUES OF X ARE     0.40000  1600.00000
AT ITN     1  F =  0.66243D-01  0.24256D+04  0.24257D+04
          X =  0.6619458D+00  0.1704247D+04
AT ITN     2  F = -0.51594D+00  0.50245D+02  0.50761D+02
          X =  0.6085199D+00  0.1676305D+04
AT ITN     3  F = -0.17430D+00  0.27468D+01  0.29211D+01
          X =  0.5645052D+00  0.1653721D+04
AT ITN     4  F = -0.48890D-01  0.18290D+01  0.18779D+01
          X =  0.5397944D+00  0.1641012D+04
AT ITN     5  F = -0.82689D-02  0.57786D+00  0.58613D+00
          X =  0.5336781D+00  0.1637861D+04
AT ITN     6  F = -0.37506D-03  0.35461D-01  0.35836D-01
          X =  0.5333736D+00  0.1637704D+04
AT ITN     7  F = -0.86980D-06  0.87963D-04  0.88833D-04
          X =  0.5333729D+00  0.1637703D+04
AT ITN     8  F = -0.47065D-11  0.48021D-09  0.48492D-09
          X =  0.5333729D+00  0.1637703D+04
 ITN =  9 NORM=  0.83267D-16
 X ARE =    0.533373 1637.703229   17.940020
CPU TIME =   0.11000 SECONDS
```

Appendix B

Listing of linear programming program

Here we include the entire program, written in FORTRAN, to accomplish the 5 food diet problem with linear programming. The program, written in single precision, is divided into the main program and 3 subroutines. This computer program is the only one used in this book that is written in single precision.

B.1 Main program for the diet problem

```
      DIMENSION A(10,20),B(10),C(20),CC(20),Z(21),ZZ(21)
      DIMENSION TITLE(20),FOOD(7,5)
      COMMON CC,ZZ,Z,FOOD
      CALL GETTIM(IH,IM,IS,I100)
      STTIME = I100 + 100*IS + 6000*IM + 360000*IH
      CALL INPUT(A,N,M,B,C,M1,TITLE)
      CALL SIMP(A,N,M,B,C,M1)
      CALL OUTPUT(A,N,M,B,C,M1,TITLE)
      CALL GETTIM(IH,IM,IS,I100)
      ENDTIME = I100 + 100*IS + 6000*IM + 360000*IH
      TIME = (ENDTIME-STTIME)*0.01
      WRITE(6,197) TIME
      WRITE(3,197) TIME
197   FORMAT(5X,'CPU TIME =',F12.5,' SECONDS')
      STOP
      END
```

B.2 Input subroutine

```
      SUBROUTINE INPUT(A,N,M,B,C,M1,TITLE)
      DIMENSION A(10,20),B(10),C(20),CC(20),Z(21),ZZ(21)
      DIMENSION TITLE(20),FOOD(7,5),CON(20)
      COMMON CC,ZZ,Z,FOOD
      OPEN( UNIT=1,FILE='FOOD5.DAT')
      DO 50 I=1,9
      DO 50 J=1,16
 50   A(I,J) = 0.0
 30   FORMAT(20A4)
      READ(1,30) (TITLE(I),I=1,20)
      KT=1
      READ(1,125) NV,NCLT,NCGT,NS
      IF(NS.GT.0) KT=NV+1
 125  FORMAT(4I5)
      NT = NCLT + NCGT + NS
      READ(1,126) (B(I),I=KT,NT)
 126  FORMAT(16F5.2)
 127  FORMAT(10X,'* * * CONSTRAINTS * * * ',10F9.2)
      NN = NV+1
      NF = NN + NCLT + NS + 2*NCGT
      NL = NCLT + 1 + NS
      NU = NL + NCGT - 1
      DO 129 I=NL,NU
      K= NV-NCLT-NS+I
      L= NV+ NCGT +I
      A(I,K) = -1.0
      C(K) = 0.0
      A(I,L) = 1.0
      C(L) = -1000.
 129  CONTINUE
  39  FORMAT(13X,'FOOD          UTILITY       COST      CAL    PROTEIN
     1IRON      MAX')
      DO 36 I=1,NV
      READ(1,37) (FOOD(I,J),J=1,5),C(I),(A(K,I),K=KT,NT),CON(I)
  36  CONTINUE
      OPEN (UNIT=3,FILE='LP5D.OUT')
      WRITE(3,31) (TITLE(I),I=1,20)
 31   FORMAT(10X,20A4)
      WRITE(3,39)
      DO 136 I=1,NV
      WRITE(3,38) I,(FOOD(I,J),J=1,5),C(I),(A(K,I),K=KT,NT),CON(I)
 136  CONTINUE
  37  FORMAT(5A4,11F5.2,F3.1,F2.0)
  38  FORMAT(1X,I2,1X,5A4,11F9.2,F5.1,1X,F3.0)
 128  FORMAT(3X,2I10)
      WRITE(3,127) (B(I),I=KT,NT)
      DO 130 I=1,NCLT
      K= NV + NCGT + NS +I
      J= I + NV
      C(K) = 0.0
 130  A(J,K) = 1.0
      IF(NS.EQ.0) GO TO 124
      DO 51 I=1,NV
      B(I) = CON(I)
      A(I,I) = 1.0
      K= NV + I + NCGT
      A(I,K) = 1.0
 51   C(K) = 0.0
 124  N=NT
      M=NF
      M1=M-1
      RETURN
      END
```

B.3 Subroutine for maximization

```
      SUBROUTINE SIMP(A,N,M,B,C,M1)
      DIMENSION A(10,20),B(10),C(20),CC(20),Z(21),ZZ(21)
      COMMON CC,ZZ,Z
      DO 8 I=1,N
      NN = M1-N+I
 8    CC(I) = C(NN)
      DO 39 KK=1,100
      IF(KK.EQ.1) GO TO 14
      TEST = Z(1)
      LC=1
      DO 30 I=2,M1
      IF(Z(I).GT.TEST) GO TO 30
      TEST = Z(I)
      LC = I
30    CONTINUE
      IF(TEST.GT.-0.001) GO TO 1000
      TEST = 1.0E7
      DO 31 I=1,N
      IF(A(I,LC).LT.0.001) GO TO 31
      RAT = B(I)/A(I,LC)
      IF(RAT.GT.TEST) GO TO 31
      TEST = RAT
      LR = I
31    CONTINUE
      AA = 1.0/A(LR,LC)
      B(LR) = B(LR)*AA
      DO 32 I=1,M1
 32   A(LR,I) = A(LR,I) *AA
      LR1=LR-1
      IF(LR1.EQ.0) GO TO 35
      LR2 = LR+1
      DO 33 I=1,LR1
      AA = A(I,LC)
      B(I) = B(I) - AA*B(LR)
      DO 33 J=1,M1
33    A(I,J) = A(I,J) -AA*A(LR,J)
      IF(LR2.GT.N) GO TO 37
      DO 34 I=LR2,N
      AA = A(I,LC)
      B(I) = B(I) -AA*B(LR)
      DO 34 J=1,M1
34    A(I,J) = A(I,J) - AA*A(LR,J)
      GO TO 37
 35   CONTINUE
      DO 36 I=2,N
      AA = A(I,LC)
      B(I) = B(I) - AA*B(LR)
      DO 36 J=1,M1
 36   A(I,J) = A(I,J) - AA*A(LR,J)
 37   CONTINUE
      CC(LR) = C(LC)
 14   CONTINUE
      DO 15 J=1,M
 15   ZZ(J) = 0.0
      DO 17 J=1,M1
      DO 16 I=1,N
 16   ZZ(J) = ZZ(J)+CC(I)*A(I,J)
 17   Z(J) = ZZ(J) - C(J)
      DO 18 I=1,N
 18   ZZ(M) = ZZ(M) + CC(I)*B(I)
      Z(M) = ZZ(M)
 39   CONTINUE
1000  CONTINUE
      RETURN
      END
```

B.4 Output subroutine

```
      SUBROUTINE OUTPUT(A,N,M,B,C,M1,TITLE)
      DIMENSION A(10,20),B(10),C(20),CC(20),Z(21),ZZ(21)
      DIMENSION TITLE(20),FOOD(7,5)
      COMMON CC,ZZ,Z,FOOD
      WRITE(3,33)
33    FORMAT(20X,'SOLUTION TO THE FOOD PROBLEM:')
      DO 51 I=1,N
      IF(CC(I).EQ.0.) GO TO 51
      DO 50 J=1,M1
      IF(C(J).EQ.CC(I)) GO TO 52
      GO TO 50
52    CONTINUE
      WRITE(3,43) J,B(I),(FOOD(J,KK),KK=1,5)
50    CONTINUE
51    CONTINUE
43    FORMAT(10X,'X(',I1,') = ',F5.2,5X,5A4)
      WRITE(3,45) ZZ(M)
45    FORMAT(10X,' MAXIMUM UTILITY =',F12.5)
      DO 60 I=1,M1
      IF(ABS(Z(I)).LT.0.001) GO TO 60
      WRITE(3,61) I,Z(I)
60    CONTINUE
61    FORMAT(10X,'SHADOW PRICE OF X(',I2,') = ',F10.3)
      RETURN
      END
```

Appendix C

LJ optimization programs

C.1 Five food diet problem

```
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(10),REG(10),XP(10),XS(10),YS(10),BETA(5),REGIN(5)
      OPEN (UNIT=3,FILE='fd51j03.OUT')
      RED=0.95
      ISEED = 37
      N = 5
      EPS = 1.0D-6
      NO = 0
      LIM = 20
      NPASSES = 20
      NR = 150
      LW=LIM
      NIT = 101
      DO 37 I=1,N
      XP(I) = 0.5D0
      YS(I) = XP(I)
      XS(I) = XP(I)
37    REG(I) =2.0D0
      CALL GETTIM(IH,IM,IS,I100)
      STTIME = I100 +100*IS+6000*IM +360000*IH
      WRITE(3,301) NR,NPASSES,RED
301   FORMAT(1X,'NR =',I5,' NPASSES =',I5,' RED =',F10.5)
      TEST = - 1.D25
      BETA(1) = 2.0D0
      BETA(2) = 3.0D0
      BETA(3) = 2.0D0
      BETA(4) = 1.0D0
      BETA(5) = 1.0D0
      DO 200 IPASS = 1,NPASSES
      LW = LIM
      WRITE(3,205) IPASS,(REG(I),I=1,N)
      WRITE(6,205) IPASS,(REG(I),I=1,N)
205   FORMAT(1X,'PASS =',I5,' REG =',5F12.7)
```

```
      DO 100 IT = 1,NIT
      DO 94 J=1,NR
      DO 52 I=1,N
      RN = URAND(ISEED)
      RAN = 2.0D0*(RN - 0.5)
      IF(J.EQ.1) RAN = 0.0
      X(I) = XP(I) + RAN*REG(I)
      IF(X(I).LT.0.0D0) X(I) = 0.0D0
      IF(X(I).GT.BETA(I)) X(I) = BETA(I)
52    CONTINUE
      T1 = 1.25D0*X(1)+0.9D0*X(2)+0.99D0*X(3)+1.1D0*X(4)+0.79D0*X(5)
      IF(T1.GT.4.0D0) GO TO 30
      T2 = 1.5D2*X(1)+1.85D2*X(2)+2.45D2*X(3)+2.7D2*X(4)+9.0D1*X(5)
      IF(T2.GT.8.0D2) GO TO 30
      T3 = X(1)+7.0D0*X(2)+2.1D1*X(3)+8.6D0*X(4)+2.0D0*X(5)
      IF(T3.LT.25.0D0) GO TO 30
      T4 = 0.7D0*X(2)+2.7D0*X(3)+1.3D0*X(4)+X(5)
      IF(T4.LT.3.5D0) GO TO 30
      P =1.0D2*X(1)+5.0D1*X(2)+2.5D1*X(3)+2.0D1*X(4)+3.5D1*X(5)
      IF(P.LT.TEST) GO TO 30
      NO = NO + 1
      TEST =P
      DO 43 I=1,N
43    XS(I) = X(I)
30    CONTINUE
94    CONTINUE
      IF(LW.LT.LIM) GO TO 51
      WRITE(3,73) IT,NO,(XS(I),I=1,N),TEST
 73   FORMAT(1X,2I7,5F12.7/10X,3F12.7)
      LW=0
51    LW = LW+1
171   FORMAT(1X,5F13.7)
172   FORMAT(1X,'PASS NUMBER =', I5, ' P = ',D16.7)
      DO 95 I=1,N
      REG(I) =REG(I)*RED
      XP(I) = XS(I)
 95   CONTINUE
100   CONTINUE
      WRITE(6,171) (XS(I),I=1,N)
      WRITE(3,171) (XS(I),I=1,N)
      WRITE(6,172) IPASS,TEST
      WRITE(3,172) IPASS,TEST
      CALL GETTIM(IH,IM,IS,I100)
      ENDTIME = I100 + 100*IS + 6000*IM + 360000*IH
      TIME = (ENDTIME-STTIME)*0.01
      WRITE(6,197) TIME
      WRITE(3,197) TIME
      DO 202 I=1,N
      REG(I) = DABS(YS(I) - XS(I))
      IF(REG(I).LT.EPS) REG(I) = EPS
      YS(I) = XS(I)
202   CONTINUE
200   CONTINUE
197   FORMAT(5X,'CPU TIME =',F12.5,' SECONDS')
      STOP
      END
```

C.2 Model reduction problem

```
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER(GAMMA=0.95D0,ETA = 0.95D0,LIM=50,N=7,NR=25,NPOINTS=100)
      PARAMETER(FACT=1.10D0,WIN=0.01D0,NPASSES=4000,NIT=50)
      DIMENSION X(15),REG(15),XP(15),XS(15),CON(15),YS(15),XX(15)
      DIMENSION REGIN(10),A(100),B(100),C(100),D(100),PLOT(10000)
      COMPLEX*16 G,GN,GD,GRED,GRN,GRD,S
      OPEN (UNIT=3,FILE='EX2NEW04.OUT')
      OPEN (UNIT=4,FILE='EX2NEW04.DAT')
      NO = 0
      LW = 1
      EPS = 1.0D-5
      XP(1) = 0.37923D0
      XP(2) = 1.4090D0
      XP(3) = 0.68093D0
      XP(4) = 1.33333D0
      XP(5) = 0.33333D0
      XP(6) = 1.0D0
      XP(7) = 0.5D0
      DO 37 I=1,N
      XS(I) = XP(I)
37    REGIN(I) =0.1d0*XP(I)/ETA
      W = WIN
      DO 50 I = 1,NPOINTS
      S = DCMPLX(0.0D0,W)
      GN = 194480.0D0+S*(482964.0D0+ S*(511812.0D0 + S*(278376.0D0 +
     1S*(82402.0D0+ S*(13285.0D0 + S*(1086.0D0 + S*35.0D0))) ))
      GD = 9600.0D0 + S*(28880.0D0 + S*(37492.0D0 + S*(27470.0D0 +
     1S*(11870.0D0 + S*(3017.0D0+ S*(437.0D0+S*(33.0D0 + S))) ))))
      G = GN/GD
      A(I) = DREAL(G)
      B(I) = DIMAG(G)
      W = W*FACT
50    CONTINUE
      ISEED = 37
      WRITE(3,21) NR,NPOINTS,GAMMA,ETA,EPS
21    FORMAT(1X,'RANDOM PTS =',2I7,2X,'RED =',2F10.5,1X,'EPS =',D12.4)
      CALL GETTIM(IH,IM,IS,I100)
      STTIME = I100 +100*IS+6000*IM +360000*IH
      TEST =1.D40
      DO 1000 NPASS = 1,NPASSES
      IF(NPASS.GT.2000) EPS = 1.0D-6
      IF(NPASS.GT.3000) EPS = 1.0D-7
      FETA= ETA**NPASS
      DO 1001 I = 1,N
      REG(I) = REGIN(I)*FETA
      IF(NPASS.GT.2) REG(I) = YS(I)
1001  CONTINUE
      DO 100 IT=1,NIT
      DO 94 NPT = 1,NR
      DO 52 I=1,N
      RN = URAND(ISEED)
      RAN = RN - 0.5D0
      IF(NPT.EQ.1) RAN = 0.0D0
      X(I) = XP(I) + 2.0D0*RAN*REG(I)
      IF(X(I).LT.0.0) GO TO 30
```

```
52    CONTINUE
      W = WIN
      SM = 0.0D0
      DO 60 I = 1,NPOINTS
      S = DCMPLX(0.0D0,W)
      GRN = 20.2583D0*(1.0d0 + x(1)*S)*(1.0D0+S*(X(2) + X(3)*S))
      GRD = (1.0D0+S*(X(4)+X(5)*S))*(1.0D0+S*(X(6)+X(7)*S))
      GRED =GRN/GRD
      C(I) = DREAL(GRED)
      D(I) = DIMAG(GRED)
      SM = SM+(C(I)-A(I))**2 + (D(I)-B(I))**2
60    W = W*FACT
      NO = NO + 1
      F = SM
      IF(F.GT.TEST) GO TO 30
      TEST = F
      FM = F
      DO 43 I=1,N
43    XS(I) = X(I)
30    CONTINUE
94    CONTINUE
      IF(LW.LT.LIM) GO TO 51
      WRITE(3,73) NPASS,IT,NO,(XS(I),I=1,N),FM
      WRITE(6,73) NPASS,IT,NO,(XS(I),I=1,N),FM
73    FORMAT(1X,2I5,I11,4F13.8/10X,4F15.8)
      LW=0
51    LW = LW+1
      DO 95 I=1,N
      REG(I) =REG(I)*GAMMA
95    XP(I) = XS(I)
100   CONTINUE
      DO 101 I=1,N
      YS(I) = DABS(XS(I)-XX(I))
      XX(I) = XS(I)
      IF(YS(I).LT.EPS) YS(I) = EPS
101   CONTINUE
      PLOT(NPASS) = FM
1000  CONTINUE
      W = WIN
      SM = 0.0D0
      DO 70 I = 1,NPOINTS
      SM = SM+(C(I)-A(I))**2 + (D(I)-B(I))**2
      A0 = DSQRT(A(I)**2+B(I)**2)
      A0 = 20.0D0*DLOG10(A0)
      AR = DSQRT(C(I)**2 + D(I)**2)
      AR = 20.0D0*DLOG10(AR)
      WRITE(3,22) I,W,A(I),B(I),A0,C(I),D(I),AR,SM
70    W = W*FACT
22    FORMAT(1X,I5,5F13.7/5X,3F13.7)
      DO 203 I = 1,NPASSES
203   WRITE(4,222) I,PLOT(I)
222   FORMAT(1X,I5,D16.8)
      CALL GETTIM(IH,IM,IS,I100)
      ENDTIME = I100 + 100*IS + 6000*IM + 360000*IH
      TIME = (ENDTIME-STTIME)*0.01
      WRITE(6,197) TIME
      WRITE(3,197) TIME
197   FORMAT(5X,'CPU TIME =',F12.4,' SECONDS')
      STOP
      END
```

C.3 Geometric problem

```
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER(GAMMA=0.95D0,N=3,EPS=1.0D-6,NR=25,NPASSES=100,NIT=5)
      PARAMETER(LIM=4,THETA = 0.05D0,NC =2,ETA = 0.70D0,IREG=5)
      DIMENSION X(N),REG(N),XP(N),XS(N),YS(N),S(NC),H(NC),PERF(NPASSES)
      DIMENSION HS(NC),REGIN(N)
      OPEN (UNIT=3,FILE='GEOM1001.OUT')
      OPEN (UNIT=4,FILE='GEOM1001.DAT')
      ISEED = 37
      NO = 0
      LW=LIM
      DO 5 I=1,NC
5     S(I) = 0.0D0
      DO 37 I=1,N
      X(I) = 0.0D0
      XP(I) = X(I)
      YS(I) = X(I)
      XS(I) = XP(I)
37    REGIN(I) =10.0D0
      CALL GETTIM(IH,IM,IS,I100)
      STTIME = I100 +100*IS+6000*IM +360000*IH
      WRITE(3,301) NR,NIT,GAMMA,THETA,EPS
301   FORMAT(1X,'NR =',I5,' NIT =',I5,' GAMMA =',F7.2,' THETA =',
     1F7.3,' EPS=',D12.4)
      DO 200 IPASS = 1,NPASSES
      TEST = -1.D25
      IF(IPASS.GT.IREG) GO TO 298
      DO 299 I=1,N
299   REG(I) = REGIN(I)*ETA**(IPASS-1)
298   CONTINUE
      LW = LIM
      WRITE(3,205) IPASS,(REG(I),I=1,N)
      WRITE(6,205) IPASS,(REG(I),I=1,N)
205   FORMAT(1X,'PASS =',I5,' REG =',5F12.7)
      DO 100 IT = 1,NIT
      DO 94 J=1,NR
      DO 52 I=1,N
      RN = URAND(ISEED)
      RAN = 2.0D0*(RN - 0.5)
      IF(J.EQ.1) RAN = 0.0
      X(I) = XP(I) + RAN*REG(I)
52    CONTINUE
      H(1) = 4.0D0*(X(1)-0.5D0)**2 + 2.0D0*(X(2)-0.2)**2 + X(3)**2 +
     10.1D0*X(1)*X(2) + 0.2D0*X(2)*X(3) - 16.0D0
      H(2) = 2.0D0*(X(1)**2 - X(3)**2) + X(2)**2  - 2.0D0
      P = X(1)**2 + X(2)**2 + X(3)**2
      AJ = P - THETA*((H(1) - S(1))**2 + (H(2) - S(2))**2)
      IF(AJ.LT.TEST) GO TO 30
      NO = NO + 1
```

```
      TEST = AJ
      FM = P
      HS(1) = H(1)
      HS(2) = H(2)
      ANORM = DABS(H(1)) + DABS(H(2))
      DO 43 I=1,N
      XS(I) = X(I)
 43   CONTINUE
30    CONTINUE
94    CONTINUE
      IF(LW.LT.LIM) GO TO 51
      WRITE(3,73) IT,NO,(XS(I),I=1,N),FM,TEST
 73   FORMAT(1X,2I7,5F12.7)
      LW=0
51    LW = LW+1
171    FORMAT(1X,5F13.7)
172    FORMAT(1X,'PASS  =', I5, ' P = ',D16.7,' SHIFTS =',2F12.6,' NORM
     1=',D13.5)
      DO 95 I=1,N
      REG(I) =REG(I)*GAMMA
      XP(I) = XS(I)
 95   CONTINUE
100   CONTINUE
177    CONTINUE
      WRITE(6,171) (XS(I),I=1,N)
      WRITE(3,171) (XS(I),I=1,N)
      WRITE(6,172) IPASS,FM,S(1),S(2),ANORM
      WRITE(3,172) IPASS,FM,S(1),S(2),ANORM
      CALL GETTIM(IH,IM,IS,I100)
      ENDTIME = I100 + 100*IS + 6000*IM + 360000*IH
      TIME = (ENDTIME-STTIME)*0.01
      WRITE(6,197) TIME
      WRITE(3,197) TIME
      DO 202 I=1,N
      REG(I) = DABS(YS(I) - XS(I))
      IF(REG(I).LT.EPS) REG(I) = EPS
      YS(I) = XS(I)
202   CONTINUE
      PERF(IPASS) = ANORM
      DO 275 I=1,NC
275   S(I) = S(I) - HS(I)
200   CONTINUE
      DO 300 I=1,NPASSES
      WRITE(4,310) I,PERF(I)
310   FORMAT(1X,I5,D13.5)
300   CONTINUE
197   FORMAT(5X,'CPU TIME =',F12.5,' SECONDS')
      STOP
      END
```

Appendix D

Iterative dynamic programming programs

D.1 CSTR with piecewise constant control

```
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
EXTERNAL CSTR
DIMENSION X(7),DX(7),XG(7,20,15)
DIMENSION US(2,20,15),UR(2,20),REGIN(2),U(2),UP(2)
DIMENSION XIN(7),FACT1(15),UIN(2,20)
COMMON U
OPEN(UNIT=3,FILE='CSTR006.OUT')
OPEN(UNIT=4,FILE='CSTR006.DAT')
CALL GETTIM(IH,IM,IS,I100)
TIMEI=I100+100*IS+6000*IM+360000*IH
ISEED = 37
NRAND = 15
NIT = 20
NU = 1
N=3
N1 = N-1
IB = 1
RED = 0.80D0
ETA = 0.50D0
NPASSES = 5
XIN(1) = 0.09D0
XIN(2) = 0.09D0
XIN(3) = 0.0D0
NT = 13
ANT = DBLE(NT)
NGX = 1
NINT = 9
AINT = DBLE(NINT)
DT = 0.78D0/(ANT*AINT)
```

```
      DO 49 I=1,NT
      UIN(1,I) = 3.0D0
49    CONTINUE
      REGIN(1) = 2.0D0/ETA
851   FORMAT(1X,I5,4F12.7)
      WRITE(6,771)RED,ETA,NRAND,ISEED
      WRITE(3,771)RED,ETA,NRAND,ISEED
771   FORMAT(1X,'RED =',2F7.3,2X,'R =',I7,'  SEED =',I7)
      DO 1000 IPASS=1,NPASSES
      DO 25 I=1,N
25    X(I) = XIN(I)
      T = 0.0D0
      DO 123 IJ=1,NU
123   REGIN(IJ) = REGIN(IJ)*ETA
      DO 50 I=1,NT
      DO 110 LU =1,NU
110   U(LU) = UIN(LU,I)
      DO 119 JK=1,N
119   XG(JK,I,1) =X(JK)
      DO 11 INT=1,NINT
      CALL RUNGK(T,X,N,DT,DX,CSTR)
      T = T + DT
11    CONTINUE
      WRITE(6,56) I,T,(X(J),J=1,N),(U(L),L=1,NU)
      WRITE(3,56) I,T,(X(J),J=1,N),(U(L),L=1,NU)
50    CONTINUE
      WRITE(6,102) NT,NGX,NRAND
      WRITE(3,102) NT,NGX,NRAND
102   FORMAT(1X,' STAGES =',I5,' NGX =',I5,' NRAND=',I5)
      DO 801 I=1,NT
      DO 801 J=1,NU
      UR(J,I) = REGIN(J)
801   CONTINUE
      IF(NGX.EQ.1) GO TO 500
      NGXX =NGX/2+1
      ANP = NGX-1
      DO 41 I=1,NGX
      A2 = 2*(I-1)
41    FACT1(I) =A2/ANP
      FACT1(NGXX)=FACT1(1)
500   CONTINUE
      DO 99 ITER = 1,NIT
      WRITE(6,100) IPASS,ITER,PERFS,X(N)
      WRITE(3,100) IPASS,ITER,PERFS,X(N)
100   FORMAT(1X,'PASS =',I5,'  ITN  =',I5,' PERF =',2F13.8)
      IF(NGX.EQ.1) GO TO 501
      DO 52 I=2,NGX
      DO 117 II=1,N
117   X(II)=XIN(II)
      T=0.D0
      DO 51 K=1,NT
      DO 57 JK = 1,NU
      U(JK) = UIN(JK,K)-UR(JK,K) +FACT1(I)*UR(JK,K)
57    CONTINUE
      DO 159 KJ = 1,N
159   XG(KJ,K,I) = X(KJ)
      DO 59 INT=1,NINT
      CALL RUNGK(T,X,N,DT,DX,CSTR)
      T = T + DT
59    CONTINUE
```

```
51    CONTINUE
52    CONTINUE
501   CONTINUE
56    FORMAT(1X,I3,F6.3,7F10.6/10X,2F10.6)
      K=NT
      DO 60 I=1,NGX
      TESTP = 1.0D20
      DO 60 L1=1,NRAND
      DO 160 J=1,N
160   X(J) =XG(J,K,I)
      DO 121 J=1,NU
      RAN = URAND(ISEED)
      RN = RAN - 0.5D0
      IF(L1.EQ.1) RN = 0.0D0
      U(J) = UIN(J,K) + RN*UR(J,K)
      UP(J) = U(J)
121   CONTINUE
      T = DBLE(NT-1)
      T = DT*T*AINT
      DO 12 INT=1,NINT
      CALL RUNGK(T,X,N,DT,DX,CSTR)
      T = T +DT
12    CONTINUE
      PERF = X(N)
      IF(PERF.GT.TESTP) GO TO 30
      PERFS = PERF
      DO 122 J=1,NU
      US(J,NT,I) = UP(J)
122   CONTINUE
      TESTP = PERF
30    CONTINUE
60    CONTINUE
      K1 = NT-2
      DO 63 LL=1,K1
      K=NT-LL
      KL=K+1
      DO 61 I=1,NGX
      TESTP = 1.0D20
      DO 64 L1=1,NRAND
      DO 164 J=1,N
164   X(J) = XG(J,K,I)
      T = DBLE(K-1)
      T = DT*T*AINT
      DO 132 J=1,NU
      RAN = URAND(ISEED)
      RN = RAN - 0.5D0
      IF(L1.EQ.1) RN = 0.0D0
      U(J) = UIN(J,K) + RN*UR(J,K)
      UP(J) = U(J)
132   CONTINUE
      DO 13 INT=1,NINT
      CALL RUNGK(T,X,N,DT,DX,CSTR)
      T = T + DT
13    CONTINUE
      DO 33 KK=KL,NT
      IF(NGX.EQ.1) GO TO 502
      TESTD = 3000.D10
      DO 62 II=1,NGX
      T1 =0.D0
      DO 162 JJ=1,N
```

```
162   T1 = T1 +(X(JJ)-XG(JJ,KK,II))**2
      IF(T1.GT.TESTD) GO TO 31
      TESTD =T1
      IB=II
31    CONTINUE
62    CONTINUE
502   CONTINUE
      T = KK-1
      T = T*DT*AINT
      DO 114 JK=1,NU
114   U(JK) = US(JK,KK,IB)
      DO 14 INT = 1,NINT
      CALL RUNGK(T,X,N,DT,DX,CSTR)
      T = T + DT
14    CONTINUE
33    CONTINUE
      PERF = X(N)
      IF(PERF.GT.TESTP) GO TO 32
      PERFS = PERF
      TESTP=PERF
      DO 125 J=1,NU
      US(J,K,I) = UP(J)
125   CONTINUE
32    CONTINUE
64    CONTINUE
61    CONTINUE
24    FORMAT(1X,11F10.5)
63    CONTINUE
      K=1
      KL=2
      TESTP= 1.0D20
      DO 74 L1=1,NRAND
      DO 163 IN = 1,N
163   X(IN) = XIN(IN)
      T=0.D0
      DO 126 J=1,NU
      RAN = URAND(ISEED)
      RN = RAN - 0.5D0
      IF(L1.EQ.1) RN = 0.0D0
      U(J) = UIN(J,K)+ RN*UR(J,K)
      UP(J) = U(J)
126   CONTINUE
      DO 15 INT=1,NINT
      CALL RUNGK(T,X,N,DT,DX,CSTR)
      T =T +DT
15    CONTINUE
      DO 93 KK=KL,NT
      IF(NGX.EQ.1) GO TO 503
      TESTD = 3000.D12
      DO 92 II=1,NGX
      T1 = 0.D0
      DO 192 JJ=1,N
192   T1 = T1 + (X(JJ)-XG(JJ,KK,II))**2
      IF(T1.GT.TESTD) GO TO 91
      TESTD=T1
      IB=II
91    CONTINUE
92    CONTINUE
503   CONTINUE
      T = DBLE(KK-1)
```

```
      T=T*DT*AINT
      DO 127 J=1,NU
      U(J) = US(J,KK,IB)
127   CONTINUE
      DO 16 INT=1,NINT
      CALL RUNGK(T,X,N,DT,DX,CSTR)
      T = T +DT
16    CONTINUE
93    CONTINUE
      PERF =X(N)
      IF(PERF.GT.TESTP) GO TO 94
      PERFS = PERF
      DO 128 J=1,NU
      US(J,K,1) = UP(J)
128   CONTINUE
      TESTP = PERF
94    CONTINUE
74    CONTINUE
      DO 118 IN = 1,N
118   X(IN) = XIN(IN)
      T= 0.D0
      DO 191 LU = 1,NU
      U(LU) = US(LU,1,1)
      UIN(LU,K) = U(LU)
191   CONTINUE
      DO 18 INT=1,NINT
      CALL RUNGK(T,X,N,DT,DX,CSTR)
      T = T+DT
18    CONTINUE
      DO 96 K=2,NT
      IF(NGX.EQ.1) GO TO 505
      TESTD=3000.D12
      DO 97 II=1,NGX
      T1 = 0.D0
      DO 197 JJ=1,N
197   T1 = T1+(X(JJ)-XG(JJ,K,II))**2
      IF(T1.GT.TESTD) GO TO 98
      TESTD = T1
      IB=II
98    CONTINUE
97    CONTINUE
505   CONTINUE
      T = K-1
      T = T*DT*AINT
      DO 1190 KJ=1,N
1190  XG(KJ,K,1) =X(KJ)
      DO 1191 IU = 1,NU
      U(IU)=US(IU,K,IB)
      UIN(IU,K) = U(IU)
1191  CONTINUE
      DO 19 INT=1,NINT
      CALL RUNGK(T,X,N,DT,DX,CSTR)
      T = T +DT
19    CONTINUE
96    CONTINUE
      DO 10 I=1,NT
      DO 10 LU = 1,NU
      UR(LU,I)= UR(LU,I)*RED
10    CONTINUE
99    CONTINUE
```

```
101   CONTINUE
      CALL GETTIM(IH,IM,IS,I100)
      TIMEF = I100 +100*IS+6000*IM+360000*IH
      TIME = (TIMEF-TIMEI)*.01
      WRITE(3,111) TIME
      WRITE(6,111) TIME
111   FORMAT(5X,'TIME =',F10.3,' SECONDS')
1000  CONTINUE
      WRITE(6,518) (X(I),I=1,N)
      WRITE(3,518) (X(I),I=1,N)
      T = 0.0D0
      DO 507 I=1,NT
      WRITE(4,508)I,T,(UIN(J,I),J=1,NU)
      T = T + DT*AINT
507   CONTINUE
518   FORMAT(1X,6F12.6)
508   FORMAT(1X,I5,4F12.7)
      STOP
      END
      SUBROUTINE CSTR(N,T,X,DX)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(7),DX(7),U(2)
      COMMON U
      AA = (X(2)+0.5D0)*DEXP(25.00D0*X(1)/(X(1)+2.0D0))
      DX(1) = AA - (X(1)+0.25D0)*(2.0D0+U(1))
      DX(2) = 0.5D0-X(2) - AA
      DX(3) = X(1)**2 + X(2)**2 + 0.1D0*U(1)**2
      RETURN
      END
      SUBROUTINE RUNGK(TSTART,Y,N,H,DY,DIFEQS)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION Y(N),DY(N)
      DIMENSION AK0(10),AK1(10),AK2(10),AK3(10),Y1(10),Y2(10),Y3(10)
      T = TSTART
      CALL DIFEQS(N,T,Y,DY)
      DO 10 I=1,N
      AK0(I) = H*DY(I)
10    Y1(I) = Y(I) + 0.5D0*AK0(I)
      T1 = T + 0.5D0*H
      CALL DIFEQS(N,T1,Y1,DY)
      DO 14 I=1,N
      AK1(I) = H*DY(I)
14    Y2(I) = Y(I) + 0.5D0*AK1(I)
      CALL DIFEQS(N,T1,Y2,DY)
      DO 15 I=1,N
      AK2(I) = H*DY(I)
15    Y3(I) = Y(I) + AK2(I)
      T2 = T + H
      CALL DIFEQS(N,T2,Y3,DY)
      DO 16 I=1,N
      AK3(I) = H*DY(I)
      Y(I) = Y(I) +(AK0(I)+2.0D0*(AK1(I)+AK2(I)) +AK3(I))/6.0D0
16    CONTINUE
      RETURN
      END
```

D.2 IDP program for piecewise linear control

```
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL CSTR
      DIMENSION X(252),DX(252),XG(252,21),US(251,21),U1S(251)
      DIMENSION XIN(252),UIN(250,21),UR(250,21),XPLOT(3,40)
      DIMENSION U1(250),U(250),GRAD(250),U2(250),PI(100)
      CHARACTER *64 FNAME
      COMMON U1,GRAD,T1,NU,N2
      FNAME = 'NAG2508.OUT'
      LIM = 70
      iseed = 37
      N=252
      NINT = 10
      ANINT =DBLE(NINT)
      NU = N-2
      N2 = N-1
      START = 0.0D0
      REGIN = 250.0D0
      GAMMA = 0.95D0
      ETA =0.90D0
      OPEN (UNIT=3,FILE=FNAME,ACCESS='APPEND')
23    FORMAT(//1X,'INITIAL CONDITIONS ARE',4F10.5/1X,7F10.5/1X,7F10.5
     1/1X,7F10.5/1X,7F10.5/1X,7F10.5)
      NIT = 70
      NT = 11
      NT1=NT + 1
      NGU = 40
      WRITE(6,332) NGU,REGIN,GAMMA,ETA
      WRITE(3,332) NGU,REGIN,GAMMA,ETA
332   FORMAT(1x,'R =',I5,' INIT REG =',F7.2,'  RED =',2F10.5)
      DT = 1.0D0/DBLE(NT)
      DELT = DT/ANINT
      DO 310 I=1,NU
      XIN(I) = DBLE(I)
      DO 310 J=1,NT1
310   UIN(I,J) = START
      XPLOT(1,1) = XIN(10)
      XPLOT(2,1) = XIN(125)
      XPLOT(3,1) = XIN(240)
      XIN(N2) = 0.0D0
      XIN(N) = 0.0D0
      WRITE(6,23) T,(XIN(I),I=1,N)
      WRITE(3,23) T,(XIN(I),I=1,N)
      CLOSE(3)
      CALL GETTIM(IH,IM,IS,I100)
      TIMEI = I100 + 100*IS + 6000*IM + 360000*IH
      DO 999 IPASS =1,LIM
      OPEN(UNIT=3,FILE=FNAME,ACCESS='APPEND')
      FACT = ETA**(IPASS -1)
      REGIN2 = REGIN* FACT
      DO 25 I=1,N
25    X(I) =XIN(I)
      T = 0.0D0
      WRITE(3,129) IPASS,GAMMA,ETA
129   FORMAT(3X,'PASS NO = ', I3, ' RED FACTORS = ',2F10.5)
```

```
      DO 49 I=1,NT1
      DO 49 J=1,NU
      UR(J,I) = REGIN2
49    CONTINUE
      DO 50 I=1,NT
      I1 = I+1
      DO 112 J=1,NU
      U1(J) = UIN(J,I)
112   U2(J) = UIN(J,I1)
      DO 111 JK=1,N
111   XG(JK,I) =X(JK)
      IND = 1
      TEND = T + DT
      T1=T
      DO 113 J=1,NU
113   GRAD(J) = (U2(J) - U1(J))/DT
      DO 114 IK = 1,NINT
      CALL RUNGK(T,X,N,DELT,DX,CSTR)
      T = T + DELT
114   CONTINUE
      T=TEND
57    FORMAT(10X,6F11.4/10X,6F11.4/10X,6F11.4/10X,6F11.4/
     110X,6F11.4/10X,6F11.4)
50    CONTINUE
      PERF = 0.0D0
      DO 1168 J=1,NU
1168  PERF = PERF + X(J) * X(J)
      PERF = 10.0D0*PERF + X(N2)
      WRITE(3,159) X(N2),PERF
      WRITE(6,159) X(N2),PERF
159   FORMAT(10X,'INITIAL VALUE OF XN1',F17.3,2x,'PERF =',F17.3)
      DO 99 LOOP = 1,NIT
100   FORMAT(//10X,'ITERATION NUMBER =',I5)
110   FORMAT(3X,'P',4X,'T',6X,'X1',8X,'X2',8X,'X3',8X,'U',5X,'I',7X,'J')
      K=NT
      TESTP=3000.D030
      DO 60 L=1,NGU
      DO 160 J=1,N
160   X(J) =XG(J,K)
      DO 161 J=1,NU
      RN = URAND(ISEED)
      RAN = (RN-0.5)*2.0
      IF(L.EQ.1) RAN = 0.0
      U1(J) = UIN(J,K) + RAN*UR(J,K)
      RN = URAND(ISEED)
      RAN = (RN-0.5)*2.0
      IF(L.EQ.1) RAN = 0.0
161   U2(J) = UIN(J,NT1)+ RAN*UR(J,NT1)
      T = DBLE(NT-1)*DT
      IND = 1
      TEND = T + DT
      T1 = T
      DO 162 J=1,NU
162   GRAD(J) = (U2(J)-U1(J))/DT
      DO 621 IL = 1,NINT
      CALL RUNGK(T,X,N,DELT,DX,CSTR)
      T = T + DELT
621   CONTINUE
      T = TEND
      PERF = 0.0D0
```

```
      DO 163 J=1,NU
163   PERF = PERF + X(J)*X(J)
      PERF = 10.0D0*PERF + X(N2)
      IF(PERF.GT.TESTP) GO TO 30
      DO 164 J=1,NU
      US(J,NT) = U1(J)
      US(J,NT1) = U2(J)
164   CONTINUE
      TESTP = PERF
      PERFS = PERF
30    CONTINUE
60    CONTINUE
      K1 = NT-1
      DO 63 LL=1,K1
      K=NT-LL
      KL=K+1
      TESTP = 3000.D30
      DO 64 L=1,NGU
      DO 165 J=1,N
165   X(J) = XG(J,K)
      T = DBLE(K-1)*DT
      DO 166 J=1,NU
      RN = URAND(ISEED)
      RAN = (RN-0.5)*2.0
      IF(L.EQ.1) RAN=0.0
      U1(J) = UIN(J,K) + RAN*UR(J,K)
      U1S(J) = U1(J)
      U2(J) = US(J,KL)
      GRAD(J) = (U2(J)-U1(J))/DT
166   CONTINUE
      T1=T
      IND = 1
      TEND = T + DT
      DO 661 IL=1,NINT
      CALL RUNGK(T,X,N,DELT,DX,CSTR)
      T = T + DELT
661   CONTINUE
      T=TEND
      DO 33 KK=KL,NT
      T = DBLE(KK-1)*DT
      K2 = KK + 1
      T1 = T
      DO 167 J=1,NU
      U1(J) = US(J,KK)
      U2(J) = US(J,K2)
167   GRAD(J) = (U2(J)-U1(J))/DT
      IND = 1
      TEND = T + DT
      DO 761 IL=1,NINT
      CALL RUNGK(T,X,N,DELT,DX,CSTR)
      T = T + DELT
761   CONTINUE
      T = TEND
33    CONTINUE
      PERF = 0.0D0
      DO 168 J=1,NU
168   PERF = PERF + X(J) * X(J)
      PERF = 10.0D0*PERF + X(N2)
      IF(PERF.GT.TESTP) GO TO 32
      TESTP=PERF
```

```
      PERFS = PERF
      DO 169 J=1,NU
169   US(J,K) = U1S(J)
32    CONTINUE
64    CONTINUE
63    CONTINUE
      DO 774 JX=1,N
774   X(JX)=XIN(JX)
      DO 96 K=1,NT
      T = DBLE(K-1)*DT
      DO 119 KJ=1,N
119   XG(KJ,K) =X(KJ)
      K2 = K+1
      T1 = T
      DO 177 J=1,NU
      U1(J) = US(J,K)
      U2(J) = US(J,K2)
      UIN(J,K) = U1(J)
177   GRAD(J) = (U2(J)-U1(J))/DT
      IND = 1
      TEND = T + DT
      DO 771 IL=1,NINT
      CALL RUNGK(T,X,N,DELT,DX,CSTR)
      T = T + DELT
771   CONTINUE
      T=TEND
      IF(LOOP.LT.NIT) GO TO 196
      XPLOT(1,K2) = X(10)
      XPLOT(2,K2) = X(125)
      XPLOT(3,K2) = X(240)
196   CONTINUE
96    CONTINUE
      WRITE(3,197) IPASS,LOOP,PERFS
      WRITE(6,197) IPASS,LOOP,PERFS
197   FORMAT(1X,'PASS NUMBER = ',I3,' ITERATION = ',I3,
     1'  PERFORMANCE INDEX = ',F20.5)
      DO 179 J=1,NU
179   UIN(J,NT1) = US(J,NT1)
      DO 10 I=1,NT1
      DO 10 J=1,NU
      UR(J,I) = UR(J,I)*GAMMA
10    CONTINUE
99    CONTINUE
101   CONTINUE
      CALL GETTIM(IH,IM,IS,I100)
      TIMEF = I100 + 100*IS + 6000*IM + 360000*IH
      TIME =(TIMEF-TIMEI)/100.
      WRITE(3,124) TIME
      WRITE(6,124) TIME
      CLOSE(3)
      PI(IPASS) = PERFS
999   CONTINUE
124   FORMAT(2X,'TIME = ',F17.5,' SECONDS')
      OPEN(UNIT=4,FILE='NAG2508.DAT')
      T = 0.0D0
      DO 181 I=1,NT1
      WRITE(4,182) T,UIN(10,I),UIN(125,I),UIN(240,I),(XPLOT(J,I),J=1,3)
      T = T + DT
181   CONTINUE
      CLOSE(4)
```

```
      OPEN(UNIT=7,FILE='NAPI2508.DAT')
      IPASS1 = IPASS-1
      DO 281 I=1,IPASS1
      DEV = PI(I) - 1.31030D7
      WRITE(7,282) I,PI(I),DEV
281   CONTINUE
282   FORMAT(1X,I5,F20.4,D15.5)
182   FORMAT(1X,7F11.5)
      STOP
      END
      SUBROUTINE CSTR(N,T,X,DX)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(252),DX(252),U1(250),GRAD(250),U(250)
      COMMON U1,GRAD,T1,NU,N2
      DO 5 J=1,NU
5     U(J) = U1(J) + GRAD(J)*(X(N)-T1)
      AK = 1.0D0
      TS = 0.0D0
      DO 7 I=1,NU
      AI = DBLE(I)
      I1 = I + 1
      DX(I) = X(I1) +U(I)
      TS = TS +AK* AI*X(I)
      AK = - AK
7     CONTINUE
      DX(NU) = TS  + U(NU)
      SUM = 0.0d0
      DO 6 J=1,NU
6     SUM = SUM + X(J)*X(J) + U(J) * U(J)
      DX(N2) = SUM
      DX(N) = 1.0D0
      RETURN
      END
      SUBROUTINE RUNGK(TSTART,Y,N,H,DY,DIFEQS)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION Y(N),DY(N)
      DIMENSION AK0(252),AK1(252),AK2(252),AK3(252)
      DIMENSION Y1(252),Y2(252),Y3(252)
      T = TSTART
      CALL DIFEQS(N,T,Y,DY)
      DO 10 I=1,N
      AK0(I) = H*DY(I)
10    Y1(I) = Y(I) + 0.5d0*AK0(I)
      T1 = T + 0.5d0*H
      CALL DIFEQS(N,T1,Y1,DY)
      DO 14 I=1,N
      AK1(I) = H*DY(I)
14    Y2(I) = Y(I) + 0.5d0*AK1(I)
      CALL DIFEQS(N,T1,Y2,DY)
      DO 15 I=1,N
      AK2(I) = H*DY(I)
15    Y3(I) = Y(I) + AK2(I)
      T2 = T + H
      CALL DIFEQS(N,T2,Y3,DY)
      DO 16 I=1,N
      AK3(I) = H*DY(I)
      Y(I) = Y(I) +(AK0(I)+2.0d0*(AK1(I)+AK2(I)) +AK3(I))/6.0d0
16    CONTINUE
      RETURN
      END
```

D.3 IDP program for variable stage lengths

```
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL CSTR, DVERK
      PARAMETER (NX=6, NU=2, NP=15, TOL=1.D-6, NPASSES=50, NIT=40)
      PARAMETER (GAMMA=0.95D0, ETA = 0.85D0, NGX = 1, NR = 5)
      PARAMETER (THETA = 1.0D2, TF = 15.0D0, NSTAB = 7)
      DIMENSION X(NX),DX(NX),XG(NX,NP,NGX),UG(NU,NP,NR),US(NU,NP,NGX)
      DIMENSION XIN(NX),FACT1(NGX),UIN(NX,NP),UR(NU,NP),UMAX(3),UMIN(NU)
      DIMENSION U(NU),C(24),W(NX,9)
      COMMON /CONT/ U,ANP,ASTAB
      OPEN (UNIT=3,FILE='RAMZ1501.OUT')
      OPEN (UNIT=4,FILE='RAMZ1501.DAT')
      CALL GETTIM(IH,IM,IS,I100)
      TIMEI = I100+100*IS+6000*IM+360000*IH
      iseed = 37
      ASTAB = DBLE(NSTAB)
      IB = 1
      ANP = DBLE(NP)
      SHIFT = 0.0D0
      UMAX(1) = 2.0D0
      UMIN(1) = 0.0d0
      UMAX(2) = 15.0D0
      UMIN(2) = 0.0001d0
      DT = 1.0D0/ANP
      XIN(1) = 0.0D0
      XIN(2) = 0.0D0
      XIN(3) = 1.0D0
      XIN(4) = 5.0D0
      XIN(5) = 1.0D0
      XIN(6) = 0.0D0
      DO 499 I=1,NP
      UIN(1,I) = 0.5D0
      UIN(2,I) = TF/ANP
499   CONTINUE
      FACT1(1) = 0.0D0
      IF(NGX.EQ.1) GO TO 411
      NGXX =NGX/2+1
      AANP = NGX-1
      DO 41 I=2,NGX
      A2 = 2*(I-1)
41    FACT1(I) =A2/AANP
411   CONTINUE
      DO 998 IPASS = 1,NPASSES
      T = 0.0D0
      DO 25 I=1,NX
 25   X(I)=XIN(I)
      WRITE(6,23) IPASS,NGX,NR,T,(XIN(I),I=1,NX),SHIFT,THETA
      WRITE(3,23) IPASS,NGX,NR,T,(XIN(I),I=1,NX),SHIFT,THETA
23    FORMAT(1X,'PASS=',I3,' NGX=',I3,' NR=',I5,'INTL COND:',4F10.5/
```

```
     15X,7F10.5)
      DO 49 I=1,NP
      UR(1,I) = 1.0D0*ETA**(IPASS-1)
49    UR(2,I) = 0.5D0*ETA**(IPASS-1)
      DO 50 I=1,NP
      DO 111 JK=1,NX
111   XG(JK,I,1) =X(JK)
      DO 501 J=1,NU
501   U(J) = UIN(J,I)
      TIN = T
      DO 801 JH = 1,NSTAB
      TEND = T + DT
      IND = 1
      CALL DVERK(NX,CSTR,T,X,TEND,TOL,IND,C,NX,W)
801   T=TEND
      T = TIN + DT
      WRITE(6,56) I,T,(X(J),J=1,NX),(U(J),J=1,NU)
      WRITE(3,56) I,T,(X(J),J=1,NX),(U(J),J=1,NU)
50    CONTINUE
      DO 99 ITN = 1,NIT
      WRITE(6,100) IPASS,ITN,THETA,GAMMA,ETA,SHIFT
      WRITE(3,100) IPASS,ITN,THETA,GAMMA,ETA,SHIFT
100   FORMAT(X,'PASS =',I3,' ITN =',I3,' THETA =',D10.3,
     1' GAMMA =',F5.3,' ETA =',F5.3, ' SHIFT =', F8.5)
      IF(NGX.EQ.1) GO TO 522
      DO 52 I=2,NGX
      DO 117 II=1,NX
117   X(II)=XIN(II)
      T=0.D0
      DO 51 K=1,NP
      DO 502 J=1,NU
      U(J) = UIN(J,K)-UR(J,K) +FACT1(I)*UR(J,K)
      IF(U(J).GT.UMAX(J)) U(J) = UMAX(J)
      IF(U(J).LT.UMIN(J)) U(J) = UMIN(J)
502   CONTINUE
      DO 159 KJ =1,NX
159   XG(KJ,K,I)=X(KJ)
      TIN = T
      DO 802 JH=1,NSTAB
      IND = 1
      TEND = T + DT
      CALL DVERK(NX,CSTR,T,X,TEND,TOL,IND,C,NX,W)
802   T=TEND
      T = TIN + DT
51    CONTINUE
52    CONTINUE
522   CONTINUE
      DO 53 K=1,NP
      DO 533 JJ=1,NU
533   UG(JJ,K,1) = UIN(JJ,K)
      DO 53 J=2,NR
      DO  503 JU=1,NU
      RAN = URAND(ISEED)
      RN = 2.0D0*(RAN - 0.5D0)
      UG(JU,K,J) = UIN(JU,K) + RN*UR(JU,K)
      IF(UG(JU,K,J).LT.UMIN(JU)) UG(JU,K,J) = UMIN(JU)
      IF(UG(JU,K,J).GT.UMAX(JU)) UG(JU,K,J) = UMAX(JU)
503   CONTINUE
53    CONTINUE
56    FORMAT(1X,I3,F7.2,6F11.5/5X,5F14.5)
```

```
      K=NP
      DO 60 I=1,NGX
      TESTP=-3000.D23
      DO 60 L=1,NR
      DO 160 J=1,NX
160   X(J) =XG(J,K,I)
      DO 504 JU=1,NU
      U(JU) = UG(JU,NP,L)
504   CONTINUE
      T = DBLE(NP-1)*DT
      TIN = T
      DO 803 JH=1,NSTAB
      IND = 1
      TEND = T + DT
      CALL DVERK(NX,CSTR,T,X,TEND,TOL,IND,C,NX,W)
803   T = TEND
      T = TIN + DT
      PERF = X(1)*X(5) - THETA*(X(NX) - TF - SHIFT)**2
      IF(PERF.LT.TESTP) GO TO 30
      DO 505 JU = 1, NU
505   US(JU,NP,I) = UG(JU,NP,L)
      TESTP = PERF
      PERFS = PERF
30    CONTINUE
60    CONTINUE
      K1 = NP-2
      DO 63 LL=1,K1
      K=NP-LL
      KL=K+1
      DO 61 I=1,NGX
      TESTP = -3000.D023
      DO 64 L=1,NR
      DO 164 J=1,NX
164   X(J) = XG(J,K,I)
      T = DBLE(K-1)*DT
      DO 506 JU=1,NU
      U(JU) = UG(JU,K,L)
506   CONTINUE
      TIN = T
      DO 804 JH = 1,NSTAB
      IND = 1
      TEND = T + DT
      CALL DVERK(NX,CSTR,T,X,TEND,TOL,IND,C,NX,W)
804   T=TEND
      T = TIN + DT
      DO 33 KK=KL,NP
      IF(NGX.EQ.1) GO TO 622
      TESTD = 3000.D23
      DO 62 II=1,NGX
      T1 =0.D0
      DO 162 JJ=1,NX
162   T1 = T1 +(X(JJ)-XG(JJ,KK,II))**2
      IF(T1.GT.TESTD) GO TO 31
      TESTD =T1
      IB=II
31    CONTINUE
62    CONTINUE
622   CONTINUE
      T = DBLE(KK-1)
      DO 507 JU=1,NU
```

```
      U(JU) = US(JU,KK,IB)
507   CONTINUE
      TIN = T
      DO 806 JH=1,NSTAB
      IND = 1
      TEND = T + DT
      CALL DVERK(NX,CSTR,T,X,TEND,TOL,IND,C,NX,W)
806   T = TEND
      T = T + DT
33    CONTINUE
      PERF = X(1)*X(5) - THETA*(X(NX) - TF - SHIFT)**2
      IF(PERF.LT.TESTP) GO TO 32
      TESTP=PERF
      PERFS = PERF
      DO 508 JU=1,NU
      US(JU,K,I) = UG(JU,K,L)
508   CONTINUE
32    CONTINUE
64    CONTINUE
61    CONTINUE
63    CONTINUE
      K =1
      KL=2
      TESTP = -3000.D23
      DO 74 L=1,NR
      DO 763 JX=1,NX
763   X(JX)=XIN(JX)
      T=0.D0
      DO 509 JU=1,NU
      U(JU) = UG(JU,K,L)
509   CONTINUE
      TIN = T
      DO 807 JH =1,NSTAB
      IND = 1
      TEND = T + DT
      CALL DVERK(NX,CSTR,T,X,TEND,TOL,IND,C,NX,W)
807   T=TEND
      T = TIN + DT
      DO 93 KK=KL,NP
      IF(NGX.EQ.1) GO TO 922
      TESTD = 3000.D7
      DO 92 II=1,NGX
      T1 = 0.0D0
      DO 192 JJ=1,NX
192   T1 = T1 + (X(JJ)-XG(JJ,KK,II))**2
      IF(T1.GT.TESTD) GO TO 91
      TESTD=T1
      IB=II
91    CONTINUE
92    CONTINUE
922   CONTINUE
      T = DBLE(KK-1)
      DO 510 JU=1,NU
      U(JU) = US(JU,KK,IB)
510   CONTINUE
      TIN = T
      DO 808 JH =1,NSTAB
      IND = 1
      TEND = T + DT
      CALL DVERK(NX,CSTR,T,X,TEND,TOL,IND,C,NX,W)
```

```
808   T = TEND
      T = TIN + DT
93    CONTINUE
      PERF = X(1)*X(5) - THETA*(X(NX) - TF - SHIFT)**2
      IF(PERF.LT.TESTP) GO TO 94
      TESTP = PERF
      PERFS = PERF
      DO 511 JU=1,NU
      US(JU,K,1)=UG(JU,K,L)
511   CONTINUE
94    CONTINUE
74    CONTINUE
      DO 774 JX=1,NX
774   X(JX)=XIN(JX)
      T= 0.0d0
      DO 512 JU=1,NU
      U(JU) = US(JU,1,1)
512   UIN(JU,K) = U(JU)
      TIN = T
      DO 809 JH=1,NSTAB
      IND = 1
      TEND = T + DT
      CALL DVERK(NX,CSTR,T,X,TEND,TOL,IND,C,NX,W)
809   T=TEND
      T = TIN + DT
      PERF = X(1)*X(5) - THETA*(X(NX) - TF - SHIFT)**2
      WRITE(6,56) K,T,(X(I),I=1,NX),(U(JU),JU=1,NU),PERF
      WRITE(3,56) K,T,(X(I),I=1,NX),(U(JU),JU=1,NU),PERF
      DO 96 K=2,NP
      IF(NGX.EQ.1) GO TO 977
      TESTD=3000.
      DO 97 II=1,NGX
      T1 = 0.
      DO 197 JJ=1,NX
197   T1 = T1+(X(JJ)-XG(JJ,K,II))**2
      IF(T1.GT.TESTD) GO TO 98
      TESTD = T1
      IB=II
98    CONTINUE
97    CONTINUE
977   CONTINUE
      T = DBLE(K-1)*DT
      DO 119 KJ=1,NX
119   XG(KJ,K,1) =X(KJ)
      DO 513 JU=1,NU
      U(JU) = US(JU,K,IB)
513   UIN(JU,K) = U(JU)
      TIN = T
      DO 810 JH=1,NSTAB
      IND = 1
      TEND = T + DT
      CALL DVERK(NX,CSTR,T,X,TEND,TOL,IND,C,NX,W)
810   T=TEND
      T = TIN + DT
      PERF = X(1)*X(5) - THETA*(X(NX) - TF - SHIFT)**2
      AA = X(1)*X(5)
      WRITE(6,56) K,T,(X(I),I=1,NX),(U(JU),JU=1,NU),PERF,AA
      WRITE(3,56) K,T,(X(I),I=1,NX),(U(JU),JU=1,NU),PERF,AA
96    CONTINUE
      DO 10 I=1,NP
```

```
      DO 514 JU=1,NU
514   UR(JU,I) = UR(JU,I)*GAMMA
10    CONTINUE
99    CONTINUE
      CALL GETTIM(IH,IM,IS,I100)
      TIMEF = I100+100*IS+6000*IM+360000*IH
      TIME = (TIMEF-TIMEI)*0.01
      WRITE(3,124) TIME
      WRITE(6,124) TIME
      SHIFT = SHIFT -(X(NX) - TF)
998   CONTINUE
124   FORMAT(2X,'TIME = ',F17.5,' SECONDS')
      TT = 0.0D0
      DO 275 I=1,NP
      WRITE(4,276) TT, UIN(1,I)
      TT = TT + UIN(2,I)
      WRITE(4,276) TT, UIN(1,I)
275   CONTINUE
276   FORMAT(1X,6F12.5)
      STOP
      END
      SUBROUTINE CSTR(N,T,X,DX)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(N),DX(N),U(2)
      COMMON /CONT/ U,ANP,ASTAB
      G3 = 21.87D0*X(4)/((X(4) + 0.4D0)*(X(4) + 62.5D0))
      G2 = X(4)*DEXP(-5.0D0*X(4))/(0.1D0 + X(4))
      G1 = 4.75D0*G3/(0.12D0 + G3)
      UP = U(2)*ANP/ASTAB
      FACT = U(1)/X(5)
      DX(1) = UP*(G1*(X(2) - X(1)) - FACT*X(1))
      DX(2) = UP*(G2*X(3) - FACT*X(2))
      DX(3) = UP*(G3*X(3) - FACT*X(3))
      DX(4) = UP*(-7.3D0*G3*X(3) + FACT*(20.0D0-X(4)))
      DX(5) = UP*U(1)
      DX(6) = UP
      RETURN
      END
```

Appendix E

Listing of DVERK

E.1 DVERK

Here we include, with the permission of the authors, the subroutine DVERK written in FORTRAN by Hull, Enright and Jackson, to integrate nonstiff ordinary differential equations.

```
      SUBROUTINE DVERK(N,FCN,X,Y,XEND,TOL,IND,C,NW,W)
      INTEGER N,IND,NW,K
      DOUBLE PRECISION X,Y(N),XEND,TOL,C(1),W(NW,9),TEMP
      EXTERNAL FCN
      IF (IND.LT.1.OR.IND.GT.6) GO TO 500
      GO TO (5,5,45,1111,2222,2222), IND
5     IF (N.GT.NW .OR. TOL.LE.0.0D0) GO TO 500
      IF (IND.EQ. 2) GO TO 15
      DO 10 K = 1, 9
      C(K) = 0.0D0
10    CONTINUE
      GO TO 35
15    CONTINUE
      DO 20 K = 1, 9
      C(K) = DABS(C(K))
20    CONTINUE
      IF (C(1).NE.4.0D0 .AND. C(1).NE.5.0D0) GO TO 30
      DO 25 K = 1, N
      C(K+30) = DABS(C(K+30))
25    CONTINUE
30    CONTINUE
35    CONTINUE
      C(10) = 2.d0**(-56)
      C(11) = 1.D-35
      C(20) = X
      DO 40 K = 21, 24
      C(K) = 0.0D0
40    CONTINUE
      GO TO 50
45    IF (C(21).NE.0.0D0 .AND.
     +  (X.NE.C(20) .OR. XEND.EQ.C(20))) GO TO 500
      C(21) = 0.0D0
      GO TO 50
50    CONTINUE
9999  CONTINUE
```

```
      IF (C(7).EQ.0.0D0 .OR. C(24).LT.C(7)) GO TO 100
      IND = -1
      RETURN
100   CONTINUE
      IF (IND .EQ. 6) GO TO 105
      CALL FCN(N, X, Y, W(1,1))
      C(24) = C(24) + 1.0D0
105   CONTINUE
      C(13) = C(3)
      IF (C(3) .NE. 0.0D0) GO TO 165
      TEMP = 0.0D0
      IF (C(1) .NE. 1.0D0) GO TO 115
      DO 110 K = 1, N
      TEMP = DMAX1(TEMP, DABS(Y(K)))
110   CONTINUE
      C(12) = TEMP
      GO TO 160
115   IF (C(1) .NE. 2.0D0) GO TO 120
      C(12) = 1.0D0
      GO TO 160
120   IF (C(1) .NE. 3.0D0) GO TO 130
      DO 125 K = 1, N
      TEMP = DMAX1(TEMP, DABS(Y(K))/C(2))
125   CONTINUE
      C(12) = DMIN1(TEMP, 1.0D0)
      GO TO 160
130   IF (C(1) .NE. 4.0D0) GO TO 140
      DO 135 K = 1, N
      TEMP = DMAX1(TEMP, DABS(Y(K))/C(K+30))
135   CONTINUE
      C(12) = DMIN1(TEMP, 1.0D0)
      GO TO 160
140   IF (C(1) .NE. 5.0D0) GO TO 150
      DO 145 K = 1, N
      TEMP = DMAX1(TEMP, DABS(Y(K))/C(K+30))
145   CONTINUE
      C(12) = TEMP
      GO TO 160
150   CONTINUE
      DO 155 K = 1, N
      TEMP = DMAX1(TEMP, DABS(Y(K)))
155   CONTINUE
      C(12) = DMIN1(TEMP, 1.0D0)
160   CONTINUE
      C(13) = 10.0D0*DMAX1(C(11),C(10)*DMAX1(C(12)/TOL,DABS(X)))
165   CONTINUE
      C(15) = C(5)
      IF (C(5) .EQ. 0.0D0) C(15) = 1.0D0
      IF (C(6).NE.0.0D0 .AND. C(5).NE.0.0D0)
     + C(16) = DMIN1(C(6), 2.0D0/C(5))
      IF (C(6).NE.0.0D0 .AND. C(5).EQ.0.0D0) C(16) = C(6)
      IF (C(6).EQ.0.0D0 .AND. C(5).NE.0.0D0) C(16) = 2.0D0/C(5)
      IF (C(6).EQ.0.0D0 .AND. C(5).EQ.0.0D0) C(16) = 2.0D0
      IF (C(13) .LE. C(16)) GO TO 170
      IND = -2
      RETURN
170   CONTINUE
      IF (IND .GT. 2) GO TO 175
      C(14) = C(4)
      IF (C(4) .EQ. 0.0D0) C(14) = C(16)*TOL**(1./6.)
```

```
      GO TO 185
175   IF (C(23) .GT. 1.0D0) GO TO 180
      TEMP = 2.0D0*C(14)
      IF (TOL .LT. (2.0D0/.9D0)**6*C(19))
     +  TEMP = .9D0*(TOL/C(19))**(1./6.)*C(14)
      C(14) = DMAX1(TEMP, .5D0*C(14))
      GO TO 185
180   CONTINUE
      C(14) = .5D0*C(14)
185   CONTINUE
      C(14) = DMIN1(C(14), C(16))
      C(14) = DMAX1(C(14), C(13))
      IF (C(8) .EQ. 0.0D0) GO TO 1111
      IND = 4
      RETURN
1111  CONTINUE
      IF (C(14) .GE. DABS(XEND - X)) GO TO 190
      C(14) = DMIN1(C(14), .5D0*DABS(XEND - X))
      C(17) = X + DSIGN(C(14), XEND - X)
      GO TO 195
190   CONTINUE
      C(14) = DABS(XEND - X)
      C(17) = XEND
195   CONTINUE
      C(18) = C(17) - X
      TEMP = C(18)/1398169080000.0D0
      DO 200 K = 1, N
      W(K,9) = Y(K) + TEMP*W(K,1)*233028180000.0D0
200   CONTINUE
      CALL FCN(N, X + C(18)/6.0D0, W(1,9), W(1,2))
      DO 205 K = 1, N
      W(K,9) = Y(K) + TEMP*(   W(K,1)*74569017600.0D0
     +  + W(K,2)*298276070400.0D0  )
205   CONTINUE
      CALL FCN(N, X + C(18)*(4.0D0/15.0D0), W(1,9), W(1,3))
      DO 210 K = 1, N
      W(K,9) = Y(K) + TEMP*(   W(K,1)*1165140900000.0D0
     +  - W(K,2)*3728450880000.0D0
     +  + W(K,3)*3495422700000.0D0 )
210   CONTINUE
      CALL FCN(N, X + C(18)*(2.0D0/3.0D0), W(1,9), W(1,4))
      DO 215 K = 1, N
      W(K,9) = Y(K) + TEMP*( - W(K,1)*3604654659375.0D0
     +  + W(K,2)*12816549900000.0D0
     +  - W(K,3)*9284716546875.0D0
     +  + W(K,4)*1237962206250.0D0 )
215   CONTINUE
      CALL FCN(N, X + C(18)*(5.0D0/6.0D0), W(1,9), W(1,5))
      DO 220 K = 1, N
      W(K,9) = Y(K) + TEMP*(   W(K,1)*3355605792000.0D0
     +  - W(K,2)*11185352640000.0D0
     +  + W(K,3)*9172628850000.0D0
     +  - W(K,4)*427218330000.0D0
     +  + W(K,5)*482505408000.0D0  )
220   CONTINUE
      CALL FCN(N, X + C(18), W(1,9), W(1,6))
      DO 225 K = 1, N
      W(K,9) = Y(K) + TEMP*( - W(K,1)*770204740536.0D0
     +  + W(K,2)*2311639545600.0D0
     +  - W(K,3)*1322092233000.0D0
```

```
     +   - W(K,4)*453006781920.0D0
     +   + W(K,5)*326875481856.0D0  )
225      CONTINUE
         CALL FCN(N, X + C(18)/15.0D0, W(1,9), W(1,7))
         DO 230 K = 1, N
         W(K,9) = Y(K) + TEMP*(   W(K,1)*2845924389000.0D0
     +   - W(K,2)*9754668000000.0D0
     +   + W(K,3)*7897110375000.0D0
     +   - W(K,4)*192082660000.0D0
     +   + W(K,5)*400298976000.0D0
     +   + W(K,7)*201586000000.0D0  )
230      CONTINUE
         CALL FCN(N, X + C(18), W(1,9), W(1,8))
         DO 235 K = 1, N
         W(K,9) = Y(K) + TEMP*(   W(K,1)*104862681000.0D0
     +   + W(K,3)*545186250000.0D0
     +   + W(K,4)*446637345000.0D0
     +   + W(K,5)*188806464000.0D0
     +   + W(K,7)*15076875000.0D0
     +   + W(K,8)*97599465000.0D0   )
235      CONTINUE
         C(24) = C(24) + 7.0D0
         DO 300 K = 1, N
         W(K,2) = (   W(K,1)*8738556750.0D0
     +   + W(K,3)*9735468750.0D0
     +   - W(K,4)*9709507500.0D0
     +   + W(K,5)*8582112000.0D0
     +   + W(K,6)*95329710000.0D0
     +   - W(K,7)*15076875000.0D0
     +   - W(K,8)*97599465000.0D0)/1398169080000.0D0
300      CONTINUE
         TEMP = 0.0D0
         IF (C(1) .NE. 1.0D0) GO TO 310
         DO 305 K = 1, N
         TEMP = DMAX1(TEMP,DABS(W(K,2)))
305      CONTINUE
         GO TO 360
310      IF (C(1) .NE. 2.0D0) GO TO 320
         DO 315 K = 1, N
         TEMP = DMAX1(TEMP, DABS(W(K,2)/Y(K)))
315      CONTINUE
         GO TO 360
320      IF (C(1) .NE. 3.0D0) GO TO 330
         DO 325 K = 1, N
         TEMP = DMAX1(TEMP, DABS(W(K,2))
     +   / DMAX1(C(2), DABS(Y(K))) )
325      CONTINUE
         GO TO 360
330      IF (C(1) .NE. 4.0D0) GO TO 340
         DO 335 K = 1, N
         TEMP = DMAX1(TEMP, DABS(W(K,2))
     +   / DMAX1(C(K+30), DABS(Y(K))) )
335      CONTINUE
         GO TO 360
340      IF (C(1) .NE. 5.0D0) GO TO 350
         DO 345 K = 1, N
         TEMP = DMAX1(TEMP, DABS(W(K,2)/C(K+30)))
345      CONTINUE
         GO TO 360
350      CONTINUE
```

```
      DO 355 K = 1, N
      TEMP = DMAX1(TEMP, DABS(W(K,2))
     +/ DMAX1(1.0D0, DABS(Y(K))) )
355   CONTINUE
360   CONTINUE
      C(19) = TEMP*C(14)*C(15)
      IND = 5
      IF (C(19) .GT. TOL) IND = 6
      IF (C(9) .EQ. 0.0D0) GO TO 2222
      RETURN
2222  CONTINUE
      IF (IND .EQ. 6) GO TO 410
      X = C(17)
      DO 400 K = 1, N
      Y(K) = W(K,9)
400   CONTINUE
      C(22) = C(22) + 1.0D0
      C(23) = 0.0D0
      IF (X .NE. XEND) GO TO 405
      IND = 3
      C(20) = XEND
      C(21) = 1.0D0
      RETURN
405   CONTINUE
      GO TO 420
410   CONTINUE
      C(23) = C(23) + 1.0D0
      IF (C(14) .GT. C(13)) GO TO 415
      IND = -3
      RETURN
415   CONTINUE
420   CONTINUE
      GO TO 9999
500   CONTINUE
      WRITE(6,505) IND, TOL, X, N, C(13), XEND, NW, C(16), C(20),
     +C(22), C(23), C(24), (Y(K), K = 1, N)
505   FORMAT( /// 1H0, 58HCOMPUTATION STOPPED IN DVERK WITH THE FOLLOWIN
     +G VALUES -
     +/ 1H0, 5HIND =, I4, 5X, 6HTOL  =, 1PD13.6, 5X, 11HX         =,
     +       1PD22.15
     +/ 1H , 5HN   =, I4, 5X, 6HHMIN =, 1PD13.6, 5X, 11HXEND      =,
     +       1PD22.15
     +/ 1H , 5HNW  =, I4, 5X, 6HHMAX =, 1PD13.6, 5X, 11HPREV XEND =,
     +       1PD22.15
     +/ 1H0, 14X, 27HNO OF SUCCESSFUL STEPS    =, 0PF8.0
     +/ 1H , 14X, 27HNO OF SUCCESSIVE FAILURES =, 0PF8.0
     +/ 1H , 14X, 27HNO OF FUNCTION EVALS      =, 0PF8.0
     +/ 1H0, 23HTHE COMPONENTS OF Y ARE
     +// (1H , 1P5D24.15)
      STOP
      END
```