



Dynamic programming using region-limiting strategies for optimization of multidimensional nonlinear processes

by Jagdish Kumar Arora

A thesis submitted to the Graduate Faculty in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY in Electrical Engineering

Montana State University

© Copyright by Jagdish Kumar Arora (1971)

Abstract:

The subject of this thesis is the development of an algorithm and a computational procedure for optimization of multidimensional, nonlinear, discrete and dynamic processes. The algorithm is based on dynamic programming, but it is free of the dimensionality problems usually associated with dynamic programming. Bounds on both state and control variables are accounted for.

The contents of the thesis are summarized as follows: First, a review of several techniques, which are described in literature and which use the method of dynamic programming, is made. Second, a description of the method of region-limiting strategies together with that of functional approximation to represent the minimal cost function is given. Third, a procedure is presented to reduce the computing effort when a quadratic polynomial is used as the approximating function. Fourth, a computer program to implement the present method is described, and the results obtained by applying the method to several different trajectory optimization problems are given. Fifth, some parallel-processing and array-processing systems are reviewed, and procedures to adapt the method of region-limiting strategies for implementation on such machines are described. And Sixth, recommendations are made to further develop the technique for a wider range of optimization problems.

DYNAMIC PROGRAMMING USING REGION-LIMITING STRATEGIES FOR  
OPTIMIZATION OF MULTIDIMENSIONAL NONLINEAR PROCESSES

by

JAGDISH KUMAR ARORA

A thesis submitted to the Graduate Faculty in partial  
fulfillment of the requirements for the degree

of

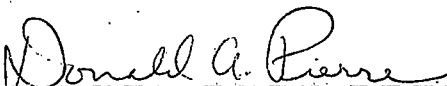
DOCTOR OF PHILOSOPHY

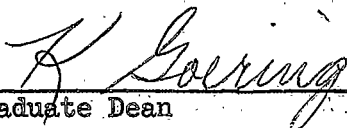
in

Electrical Engineering

Approved:

  
Head, Major Department

  
Chairman, Examining Committee

  
Graduate Dean

MONTANA STATE UNIVERSITY  
Bozeman, Montana

August, 1971

## ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to his adviser, Dr. D. A. Pierre, for his active participation and guidance during the course of this research.

The financial support provided by the Department of Electrical Engineering at Montana State University and the Office of Naval Research under Contract No. N00014-67-C-0477 is gratefully acknowledged. Also, the work described in this thesis could not have been accomplished without extensive use of the XDS-SIGMA-7 computer. The author is very thankful for the financial assistance which made its use possible.

The author is indebted to his wife, Asha, and other members of his family for their help and for their fortitude during the current phase of his studies.

## TABLE OF CONTENTS

	Page
CHAPTER 1: INTRODUCTION AND REVIEW OF PERTINENT LITERATURE . . . . .	1
1.1 Introduction . . . . .	2
1.2 The Problem Statement . . . . .	5
1.3 The Dynamic Programming Formulation . . . . .	7
1.4 A Review of Dynamic Programming Methods . . . . .	9
1.4.1 The Second Variation Methods . . . . .	10
1.4.2 Decomposition Methods . . . . .	15
1.4.3 Quasilinearization Methods . . . . .	17
1.4.4 Direct Methods . . . . .	19
1.4.5 A Comparison . . . . .	21
1.5 Other Related Research . . . . .	22
1.6 This Work--An Outline . . . . .	23
CHAPTER 2: FUNCTIONAL APPROXIMATION AND THE METHOD OF REGION-LIMITING STRATEGIES . . . . .	26
2.1 Introduction . . . . .	27
2.2 The Recurrence Relations in Dynamic Programming . . . . .	28
2.2.1 The Discrete Case . . . . .	29
2.2.2 The Continuous Case . . . . .	31
2.3 Functional Approximation . . . . .	33
2.4 The Method of Region-Limiting Strategies . . . . .	34

	Page
2.5 Quadratic Approximating Function . . . . .	36
2.5.1 Coefficients of the Quadratic Polynomial. . . . .	37
2.6 The Optimization Procedure . . . . .	39
2.6.1 Calculation of Minimal Cost Functions . . . . .	40
2.6.2 Updating the Trajectory . . . . .	42
2.7 An Initial Trajectory . . . . .	45
2.8 Convergence of the Solution . . . . .	46
2.9 Continuously Variable Control. . . . .	47
2.10 Conclusions . . . . .	52
 CHAPTER 3: ALGORITHMS AND COMPUTATIONAL RESULTS . . . . .	 54
3.1 Introduction . . . . .	55
3.2 The Computer Program . . . . .	56
3.2.1 Subroutine GRID . . . . .	63
3.2.2 Cost From the Approximating Polynomial. . . . .	68
3.2.3 Other Subroutines . . . . .	76
3.3 Computational Results . . . . .	77
3.3.1 A Discrete-Time Three-State Variable Problem . . . . .	77
3.3.2 A Continuous-Time Four-Variable Problem. . . . .	82
3.3.3 An Orbit Transfer Problem. . . . .	85
3.3.4 A Tenth-Order Problem . . . . .	91

	Page
3.4 Grid Spacing and Rate of Convergence . . . . .	92
CHAPTER 4: REAL-TIME COMPUTATIONS AND PARALLEL PROCESSING . . . . .	97
4.1 Introduction . . . . .	98
4.2 Computing Systems with Multiple Processors and Their Application . . . . .	99
4.2.1 A Parallel-Processing System . . . . .	100
4.2.1.1 Instructions for a Parallel Processor . . . . .	101
4.2.1.2 Structure of a Parallel Processor . . . . .	103
4.2.1.3 Programming for a Parallel Processor . . . . .	106
4.2.1.4 Method of Region-Limiting Strategies and Parallel Processing . . . . .	107
4.3 Array Processor Systems and Their Applications . . . . .	113
4.3.1 ILLIAC IV Computer . . . . .	113
4.3.2 A Cellular-Array Computer . . . . .	117
4.3.3 Application of an Array-Processor System . . . . .	120
4.4 Suboptimal Control Trajectories . . . . .	126
CHAPTER 5: SUMMARY AND CONCLUSIONS . . . . .	130
5.1 A Summary and an Evaluation . . . . .	131
5.2 Related Domains for Further Study . . . . .	134

	Page
APPENDIX A . . . . .	137
APPENDIX B . . . . .	152
APPENDIX C . . . . .	188
REFERENCES . . . . .	190

## LIST OF TABLES

	Page
Table 3.1 Final trajectory and control . . . . .	79
Table 3.2 Final trajectory and control . . . . .	80
Table 3.3 Final trajectory and control . . . . .	81
Table 3.4 Final trajectory and control (no bounds on variables). . . . .	82
Table 3.5 Final trajectory and control (bounded case) . . . . .	82
Table 3.6 Control trajectories for the tenth- order problem . . . . .	93
Table 4.1 Assumed weights for various operations . . . . .	111
Table 4.2 Comparison of weighted operations count for various numbers of processors for the six-variable case with ten stages and twenty-one discrete controls . . . . .	113
Table 4.3 Performance measures for suboptimal and optimal trajectories . . . . .	129



## LIST OF FIGURES

	Page
Figure 3.1	Flowchart for subroutine MINMUM . . . . . 58
Figure 3.2	Flowchart for subroutine GRID . . . . . 64
Figure 3.3	Flowchart for subroutine INPOL1 . . . . . 69
Figure 3.4	Flowchart for subroutine INPOL2 . . . . . 73
Figure 3.5	Block diagram of a computer control system . 78
Figure 3.6(a)	Control trajectories for the orbit transfer problem (initial trajectory and the trajectory for $M = 500$ in the performance measure (3.13) ) . . . . . 88
Figure 3.6(b)	Intermediate control trajectories, (i) $M = 1000$ in the performance measure (3.13), and (ii) performance measure as given by (3.12). . . . . 89
Figure 3.6(c)	Final control trajectories, (i) performance measure as given by (3.14), and (ii), the trajectory obtained by McReynolds [13] . . . . . 90
Figure 3.7	State variable trajectories for the tenth-order problem . . . . . 94
Figure 4.1	A configuration for a parallel-processing system . . . . . 104
Figure 4.2	Flowchart for an algorithm to compute minimal costs at the base points . . . . . 108
Figure 4.3	ILLIAC IV system configuration . . . . . 115
Figure 4.4	Computer organizations of a matrix-oriented cellular computer . . . . . 118

		Page
Figure 4.5	The computation of $D^{-1}\underline{f}$ from $\underline{f}$ on an array computer . . . . .	122
Figure 4.6	Column vector summation for evaluating $P(\underline{z})$ on an array computer . . . . .	125
Figure A.1	Base points for quadratic approximation in two variables . . . . .	142
Figure A.2	Construction of matrix $D$ from the base points and the basis vectors for the second-order case . . . . .	142
Figure A.3	Base points of the third type for the five-variable case . . . . .	143
Figure A.4	Desired order of the elements of the vectors $\underline{f}$ and $\underline{\Phi}$ . . . . .	147
Figure A.5(a)	Matrix $D^{-1}$ for the three-variable case . . .	149
Figure A.5(b)	Vector $D^{-1}\underline{f}$ for the three-variable case . .	150

## ABSTRACT

The subject of this thesis is the development of an algorithm and a computational procedure for optimization of multidimensional, nonlinear, discrete and dynamic processes. The algorithm is based on dynamic programming, but it is free of the dimensionality problems usually associated with dynamic programming. Bounds on both state and control variables are accounted for.

The contents of the thesis are summarized as follows: First, a review of several techniques, which are described in literature and which use the method of dynamic programming, is made. Second, a description of the method of region-limiting strategies together with that of functional approximation to represent the minimal cost function is given. Third, a procedure is presented to reduce the computing effort when a quadratic polynomial is used as the approximating function. Fourth, a computer program to implement the present method is described, and the results obtained by applying the method to several different trajectory optimization problems are given. Fifth, some parallel-processing and array-processing systems are reviewed, and procedures to adapt the method of region-limiting strategies for implementation on such machines are described. And Sixth, recommendations are made to further develop the technique for a wider range of optimization problems.

CHAPTER 1

INTRODUCTION AND REVIEW  
OF PERTINENT LITERATURE

### 1.1      Introduction

The theory of optimization is a result of efforts by designers to obtain the best possible performance from their systems. Optimization is a three-step process, the first of which is the construction of a mathematical model of the system, the second is the rational specification of a measure of performance, and the last, the optimization proper, is the specification of variables to obtain optimum performance.

In the last two decades, significant advances have been made in the development of optimization theory and its applications. Many of these have been motivated by the need to design optimal control systems; in the process, control system design has been transformed into an engineering science. The earlier cut-and-try approaches to the design of feedback compensation were adequate to settle certain questions of stability, overshoot, and steady-state response. These approaches were augmented in the 1950's by Wiener's approach to minimization of mean-squared and integral-squared error in linear constant-coefficient systems. The early techniques relied heavily on Laplace and Fourier transform theory. To effect control system optimization directly in the time domain, classical methods of the calculus of variations [1,2] have been applied with limited success because of the discontinuities encountered in most control problems.

Some of the difficulties associated with classical variational methods have been surmounted by Pontryagin's maximum principle [1], a generalization of the classical variational calculus, which gives necessary conditions for optimality, but computational schemes for solving these necessary conditions are generally unsuitable for on-line control. Pontryagin's principle was originally developed for continuous-time systems, systems with dynamics characterized by differential equations. It was later extended for discrete-time processes described by difference equations. Several authors [3,4,5] have discussed the necessary conditions to solve discrete-time problems using the maximum principle. These conditions have been summarized by Athans [6].

To optimize discrete-time control processes, Bellman applied what he termed dynamic programming [7]. The method is conceptually straightforward and is based on the principle of causality and on what has come to be known as the principle of optimality. The former assures that the control at a stage does not affect the results of preceding stages, while the latter specifies the mode of selection of optimum controls. In the dynamic programming process, a sequence of optimum controls is generated by minimizing a sequence of nonlinear cost functions. The equivalent problem in continuous-time systems involves the solution of a nonlinear partial differential equation as an initial-value problem [8].

If the solution of an optimal control problem results in control actions which are determined at each instant on the basis of the system state at that instant, the control is called a control law, and it yields a closed-loop solution. This is very desirable. If, however, a sequence of control actions is generated in terms of an initial system state, and is applied without regard to subsequent state and disturbance information, the control is called a control function and it yields an open-loop solution.

The earliest control problem solved analytically using dynamic programming was the linear-quadratic problem [9], i.e., a problem with linear dynamics and a quadratic performance criterion. In that case, the optimal closed-loop control follows from a solution of the so-called discrete Riccati equation. For nonlinear processes or problems with nonquadratic performance criteria, numerical solutions are generally required.

Developments in the field of computational algorithms for numerical optimization tend to parallel developments in the field of computers. Some computational schemes based on the maximum principle have been described for discrete-time problems [10]. But these approaches have not been favored because of the difficulty of implementing bounds on state and control variables and because of the sensitivity of solutions to the terminal boundary values.

Dynamic programming, because of its conceptual simplicity and generality, is applicable in the solution of a large class of optimal design problems. One attractive feature of the method is the ease with which constraints on variables can be handled. There are difficulties associated with its application to feedback control, however, and most developments have been to obtain optimal open-loop controls and trajectories which start from a single point in state space [11].

In this chapter a mathematical formulation of a discrete, dynamic, trajectory optimization problem is given (Sections 1.2 and 1.3). The methods currently available to solve this problem are reviewed (Section 1.4). The reason for having a multiplicity of solution techniques is that one procedure is really not suitable for treating all nonlinear problems. For each class of problems several techniques may be tried before the most appropriate one is selected. A summary of some of the work leading up to this dissertation is given in Section 1.5, and an outline of the remainder of the thesis is given in Section 1.6.

## 1.2 The Problem Statement

A  $K-1$  stage, discrete, deterministic, dynamic process may be characterized by a vector difference equation of the form

$$\underline{x}_{k+1} = \underline{q}(\underline{x}_k, \underline{m}_k, k), \quad k = 1, 2, \dots, K-1 \quad (1.1)$$



The state vector  $\underline{x}_k$  is an  $n \times 1$  vector at the  $k^{\text{th}}$  stage,  $g$  is an  $n$ -dimensional vector function, and  $\underline{m}_k$  is an  $r \times 1$  vector control action at the  $k^{\text{th}}$  stage.  $\underline{x}_{k+1}$  is a function of  $\underline{x}_k$  and  $\underline{m}_k$  and may be an explicit function of time which is represented by the integer  $k$ . Elements of  $\underline{x}_k$  and  $\underline{m}_k$  are denoted as  $x_1^k, x_2^k, \dots, x_n^k$ , and  $m_1^k, m_2^k, \dots, m_r^k$ , respectively.

The performance criterion of this  $K-1$  stage process is given by

$$J = \sum_{k=2}^K f_k(\underline{x}_k, \underline{m}_{k-1}) \quad (1.2)$$

where  $f_k$ 's are given real-valued stage costs. Given  $\underline{x}_1$ , a sequence of controls  $\underline{m}_1, \underline{m}_2, \dots, \underline{m}_{K-1}$  is desired such that (1.2) is minimized.

In addition to state-equation constraints (1.1), the state variables and control actions may be subject to constraints of the type

$$x_{i,\min}^k \leq x_i^k \leq x_{i,\max}^k; \quad x_{i,\min}^k \text{ and } x_{i,\max}^k = \text{constants} \quad (1.3)$$

for  $i = 1, 2, \dots, n$ , and  $k = 1, 2, \dots, K$ ;

and

$$m_{i,\min}^k \leq m_i^k \leq m_{i,\max}^k; \quad m_{i,\min}^k \text{ and } m_{i,\max}^k = \text{constants} \quad (1.4)$$

for  $i = 1, 2, \dots, r$ , and  $k = 1, 2, \dots, K-1$ . Of those control sequences that satisfy (1.1), (1.3), and (1.4), the optimum one yields a minimum of  $J$  in (1.2).

### 1.3 The Dynamic Programming Formulation

The principle of optimality, when applied to the discrete-time system of Section 1.2, yields a recurrence relation<sup>1</sup>:

$$B_{K-k}(x_k) = \min_{m_k \in U_k} \left\{ f_{k+1} \left[ g(x_k, m_k, k), m_k \right] + B_{K-k-1} \left[ g(x_k, m_k, k) \right] \right\} \quad (1.5)$$

$B_{K-k}$  is the optimal cost of  $K-k$  stages of the process, from the  $(k+1)^{\text{th}}$  to  $K^{\text{th}}$ ; and  $B_{K-k-1}$  is the optimal cost of  $K-k-1$  stages, from the  $(k+2)^{\text{th}}$  to the final stage. The set  $U_k$  is characterized by (1.4). The minimization in (1.5), for  $k = 1, 2, \dots, K-1$ , yields a sequence of optimal controls.

For a process with linear dynamics and a quadratic performance criterion,  $B_{K-k}$  can be obtained as a quadratic function of the state vector at the  $k^{\text{th}}$  stage, provided that constraints (1.3) and (1.4) do not apply, and an analytical solution can be obtained for a control law. Otherwise, for a numerical solution to the problem, with a straightforward application of the dynamic programming technique, a knowledge of  $B$  at all the stages of the process is required. Because a given state variable can assume any value between its specified bounds, an analytical expression for  $B$  would be very helpful for the minimization

---

<sup>1</sup>This is derived in Chapter 2.

indicated in (1.5). B may, however, be discontinuous or at best a non-linear function of the state vector. B is, therefore, evaluated at a number of points at each stage, and the information so obtained is used to find approximate values of B at other points.

The number of points at which B must be evaluated for accurate approximations depends on the number of state variables. If the state vector has 4 or more components, difficulties are encountered which Bellman refers to as the "curse of dimensionality". If B is evaluated at the points formed by quantizing each state variable at the  $k^{\text{th}}$  stage, the total number  $N_k$  of points at which B must be evaluated at the  $k^{\text{th}}$  stage is

$$N_k = \prod_{i=1}^n N_{i,k} \quad (1.6)$$

where  $N_{i,k}$  is the number of points associated with the  $i^{\text{th}}$  variable. The number  $N_k$  increases exponentially with the number of state variables, and therefore, rapid-access core memory requirements tend to be very large for multi-dimensional problems. The execution time tends to be very large too. In general, three is considered to be the maximum size of state vector for which optimization using straightforward dynamic programming is realistic [12]. Several methods have been devised to circumvent these dimensionality problems.

#### 1.4 A Review of Dynamic Programming Methods

In this section, the relative merits of previously developed dynamic programming approaches to the problem of Sections 1.2 and 1.3 are examined. These numerical techniques are designed to do one or more of the following:

1. To find an alternative to solving a two-point boundary-value problem obtained by applying the maximum principle.
2. To reduce rapid-access memory requirements.
3. To reduce a problem with a large state vector to one of manageable dimensions.

In all methods, an initial trajectory satisfying the given constraints is the starting point. Most methods belong in one of the following categories:

1. The Methods of Successive Approximations
2. Quasilinearization
3. Discrete Methods

The first two categories contain iterative procedures to obtain an optimal solution. The methods of successive approximations encompass several different algorithms. Among these are the decomposition methods and the methods employing second variations of the cost function.

### 1.4.1.1 The Second Variation Methods

The successive sweep method, due to McReynolds [13], and Mayne's differential dynamic programming [14,15] both employ second variations and resemble each other closely. A development of the latter is given in this Subsection. Because these algorithms were developed for discrete approximations to continuous-time systems, the performance criterion used is given as (see Subsection 2.2.2):

$$J = F(\underline{x}_K) + \sum_{k=0}^{K-1} f_k(\underline{x}_k, \underline{m}_k) \quad (1.7)$$

A second-order Taylor series expansion of the cost function is used as an approximation in a small region around a nominal trajectory; third and higher-order terms are assumed to be negligible for the calculation of incremental cost in this region.  $\{\underline{m}_k\}$  is used to denote a control sequence,  $\{\underline{x}_k\}$  to denote a state trajectory, both starting at stage k, and  $V_k(\underline{x}_k)$  denotes the cost for a trajectory starting at  $\underline{x}_k$  with a control sequence  $\{\underline{m}_k\}$ . When both subscripts and superscripts are used, the superscript indicates the stage, and the subscript indicates partial differentiation.

A nominal trajectory  $\{\bar{\underline{x}}_k\}$  is obtained using a control sequence  $\{\bar{\underline{m}}_k\}$ . The following relations hold for points on the trajectory:

$$\bar{V}_k(\bar{\underline{x}}_k) = f_k(\bar{\underline{x}}_k, \bar{\underline{m}}_k) + \bar{V}_{k+1}(\bar{\underline{x}}_{k+1}) \quad (1.8)$$

and

$$\bar{V}_k(\bar{x}_k) = F(\bar{x}_k). \quad (1.9)$$

Let  $\{\underline{m}_k\}$  denote another policy which generates a trajectory  $\{\underline{x}_k\}$  in the neighborhood of  $\{\bar{x}_k\}$ , where

$$\underline{m}_k = \bar{m}_k + \delta m_k \quad (1.10)$$

$$\delta m_k \triangleq \alpha_k + \beta_k^T \delta x_k \quad (1.11)$$

and

$$\delta x_k \triangleq x_k - \bar{x}_k. \quad (1.12)$$

Then

$$V_k(x_k) = f_k(x_k, \underline{m}_k) + V_{k+1}(x_{k+1}) \quad (1.13)$$

where

$$\underline{m}_k = \bar{m}_k + \alpha_k + \beta_k^T \delta x_k \quad (1.14)$$

Also, let

$$V_k(x_k) \triangleq \bar{V}_k(\bar{x}_k) + a_k \quad (1.15)$$

The superscript T signifies a matrix transpose. By using a Taylor series expansion in the neighborhood of  $\{\bar{x}_k\}$ ,  $V_k(x_k)$  for a point  $x_k$  can be written as

$$V_k(x_k) = V_k(\bar{x}_k) + \left[ V_x^k(\bar{x}_k) \right]^T (\delta x_k) + \frac{1}{2} (\delta x_k)^T \left[ V_{xx}^k(\bar{x}_k) \right] (\delta x_k) + \dots \quad (1.16)$$

where  $\bar{V}_k(\bar{x}_k)$  is the cost function which results from control  $\{\bar{m}_k\}$  and

state trajectory  $\{\bar{x}_k\}$  starting from  $\bar{x}_k$ , while  $V_k(\bar{x}_k)$  is the cost function obtained by using a control sequence  $\{\bar{m}_k\}$  and initial condition  $\bar{x}_k$ ; then, within second-order variations, equations (1.15) and (1.16) are used to obtain

$$V_k(\underline{x}_k) = \bar{V}_k(\bar{x}_k) + a_k + \left[ V_x^k(\bar{x}_k) \right]^T (\delta \underline{x}_k) + \frac{1}{2} (\delta \underline{x}_k)^T \left[ V_{xx}^k(\bar{x}_k) \right]^T (\delta \underline{x}_k) \quad (1.17)$$

The right-hand member of (1.13) can be expanded in a Taylor series:

$$\begin{aligned} f_k(\underline{x}_k, \underline{m}_k) + \bar{V}_{k+1}(\bar{x}_{k+1}) &= f_k + V_{k+1}(\bar{x}_{k+1}) + a_{k+1} \\ &+ \left\{ \left[ f_x^k \right]^T + \left[ V_x^{k+1}(\bar{x}_{k+1}) \right]^T (g_x^k) \right\} (\delta \underline{x}_k) \\ &+ \left\{ \left[ f_m^k \right]^T + \left[ V_x^{k+1}(\bar{x}_{k+1}) \right]^T (g_m^k) \right\} (\delta \underline{m}_k) \\ &+ \frac{1}{2} (\delta \underline{x}_k)^T E (\delta \underline{x}_k) + \frac{1}{2} (\delta \underline{m}_k)^T F (\delta \underline{m}_k) \\ &+ (\delta \underline{m}_k)^T G (\delta \underline{x}_k) \end{aligned} \quad (1.18)$$

where

$$E = f_{xx}^k + (g_m^k)^T V_{xx}^{k+1}(\bar{x}_{k+1}) (g_x^k) + \left[ V_x^{k+1}(\bar{x}_{k+1}) \right]^T (g_{xx}^k) \quad (1.19)$$

$$F = f_{mm}^k + (g_m^k)^T V_{xx}^{k+1}(\bar{x}_{k+1}) (g_x^k) + \left[ V_x^{k+1}(\bar{x}_{k+1}) \right]^T (g_{mm}^k) \quad (1.20)$$

and

$$G = f_{mx}^k + (g_m^k)^T V_{xx}^{k+1}(\bar{x}_{k+1}) (g_x^k) + \left[ V_x^{k+1}(\bar{x}_{k+1}) \right]^T (g_{mx}^k) \quad (1.21)$$

The unspecified arguments are  $\bar{x}_k, \bar{m}_k, k = 1, 2, \dots, K-1$ .

As given by (1.13), the right-hand members of (1.17) and (1.18) are equal. If  $\delta m_k$  is replaced by  $\alpha_k + \beta_k^T \delta x_k$  in (1.18) and coefficients of like powers of  $\delta x_k$  in (1.17) and (1.18) are equated, vector-difference equations for  $a_k, V_x^k$ , and  $V_{xx}^k$  are obtained. The terminal conditions on these are:

$$a_K = 0 \tag{1.22}$$

$$V_x^K(\bar{x}_K) = f_x(\bar{x}_K) \tag{1.23}$$

and

$$V_{xx}^K(\bar{x}_K) = F_{xx}(\bar{x}_K) \tag{1.24}$$

Expressions for  $\alpha_k$  and  $\beta_k$  for  $k = 1, 2, \dots, K-1$  are obtained by minimizing the right-hand side of (1.18) with respect to  $\delta m_k$ . These are used to generate a new trajectory within a small region of the old trajectory. The new control is obtained using

$$\underline{m}_k = \bar{m}_k + \epsilon \left[ \alpha_k + \beta_k^T (\underline{x}_k - \bar{x}_k) \right], \quad 0 < \epsilon \leq 1. \tag{1.25}$$

The actual algorithm consists of the following steps:

1. For the given initial condition and a control sequence  $\{\bar{m}_k\}$ , a nominal trajectory  $\{\bar{x}_k\}$  is obtained.



2.  $a_k$ ,  $V_x(\bar{x}_k)$ , and  $V_{xx}(\bar{x}_k)$  are calculated, starting at  $k = K-1$ , for  $k = K-1, K-2, \dots, 0$ .
3. The change in the value of the performance criterion is calculated as

$$\Delta V = \sum_{k=1}^{K-1} \left[ f_k(x_k, m_k) - f_k(\bar{x}_k, \bar{m}_k) \right] + F(x_K) - F(\bar{x}_K) \quad (1.26)$$

4. If  $\Delta V$  is positive, i.e., the cost has increased, or if  $\Delta V$  is negative but  $\frac{\Delta V}{|\epsilon(1-\epsilon/2)a_0|} < c$ ,  $0 \leq c \leq 1$ , the procedure is repeated with a smaller value of  $\epsilon$ ;  $c$  is, in effect, the limiting ratio of change in performance measure to the estimated value for the specified value of  $\epsilon$  ( $a_0$  is calculated for  $\epsilon = 1$ ). A value of  $c = 0.5$  is recommended by Mayne.
5. However, if  $V$  is negative and  $\frac{\Delta V}{|\epsilon(1-\epsilon/2)a_0|} \geq c$ , the new trajectory and control sequence are stored.
6. Computations are halted when  $|a_0(x_0)| \leq \eta$ , where  $\eta$  is a small positive quantity.

It is to be noted that in the case of problems with linear dynamics and quadratic performance criteria, the algorithm is quadratic convergent, which is not the case if the differential calculus approach [16] is used. The latter also involves the solution of a larger number of difference equations. Problems involving bounds on state and control variables are not treated by this method except as they may be incorporated by use of Lagrange multipliers or penalty functions.

#### 1.4.2 Decomposition Methods

In an effort to solve optimization problems with state vectors of very large dimensions, Collins [17] presented recently the method of diagonal decomposition. This was followed by the structural decomposition method by Collins and Lew [18].

Both of the methods deal with problems in which both  $\underline{x}$  and  $\underline{m}$  are  $n \times 1$  vectors. The state equations are assumed to be of the form

$$\underline{x}_{k+1} = A\underline{x}_k + C\underline{m}_k \quad (1.27)$$

where A and C both are  $n \times n$  matrices. Matrix A is decomposed as

$$A = T + D \quad (1.28)$$

In the diagonal decomposition method, T includes only the diagonal elements of A; while in structural decomposition, T also includes some other dominating elements. D is composed of the remaining elements of A, and C is assumed to be a diagonal matrix. The following presentation assumes structural decomposition of A.

If  $\{\bar{\underline{x}}_k\}$  is a nominal trajectory, the minimization is effected around this trajectory using a pseudo-state equation:

$$\underline{x}_{k+1} = T\underline{x}_k + C\underline{m}_k + D\bar{\underline{x}}_k \quad (1.29)$$

At each stage, the minimization is effected for one component of  $\underline{x}$  at

a time. If the method of diagonal decomposition were to be used,  $T$  would be a diagonal matrix, and the state equations would be effectively decoupled for all components of  $\underline{x}$ . In the method of structural decomposition, this decoupling effect can be obtained by rewriting (1.29), in the form

$${}^{(i)}\underline{x}_{k+1} = T_i {}^{(i)}\underline{x}_k + \underline{c}_i m_i^k + \underline{d}_i \quad (1.30)$$

where the length of the  $n_i \times 1$  vectors  ${}^{(i)}\underline{x}_k$ ,  $\underline{c}_i$ , and  $\underline{d}_i$  varies with the index  $n_i$ .  $n_i$  is the number of nonzero elements in the  $i^{\text{th}}$  row of  $T$ ; the  $n_i \times n_i$  matrix  $T_i$  has rows that correspond to these nonzero elements. The matrix  $T_i$  is constructed like an identity matrix except in the row which corresponds to the  $i^{\text{th}}$  nonzero element in  $i^{\text{th}}$  row of  $T$ . This row consists of  $n_i$  nonzero elements in that row of  $T$ .  ${}^{(i)}\underline{x}_k$  and  $\underline{c}_i$  consist of those elements in  $\underline{x}_k$  and  $i^{\text{th}}$  column of  $C$ , respectively, which correspond to the  $n_i$  nonzero elements in  $i^{\text{th}}$  row of  $T$ . The only nonzero element of  $\underline{d}_i$  is the row in which  $T_i$  differs from an identity matrix, and this element is the  $i^{\text{th}}$  element in the product  $D\underline{x}_k$ .

To simplify notation, relation (1.30) is written as

$$\underline{x}_{i,k+1} = \underline{q}_i(\underline{x}_{i,k}, m_i^k, k) \quad (1.31)$$

The recurrence relation for minimization with respect to  $m_i^k$  is then

$$B_i^{K-k}(\underline{x}_{i,k}) = \min_{m_i^k} \epsilon \min_{U_i^k} \left\{ f_i^{k+1} \left[ g_i(\underline{x}_{i,k}, m_i^k, k), m_i^k \right] + B_i^{K-k-1} \left[ g_i(\underline{x}_{i,k}, m_i^k, k) \right] \right\} \quad (1.32)$$

Interpolation in  $n_i$  components of  $\underline{x}$  is required to calculate the value of  $B_i$  at the next stage, thus offsetting the problem of dimensionality to some extent. If diagonal decomposition were possible, only scalar interpolation would be necessary.

The preceding methods can be extended for application to certain nonlinear processes. A similar method, the dynamic programming successive approximations technique [19], has been described by Larson and Korsak, who also give convergence proofs for some special classes of problems [20]. Yet another decomposition procedure using the state transition operator as a shift operator has been presented by Wong [21]. He claims that the number of degrees of freedom in the state transitions of an  $n^{\text{th}}$  order system is more nearly the dimension of the control vector than of the state vector [22] and, therefore, that the high-speed memory requirements can be reduced. In all of these methods, bounds on state and control variables can be treated by using Lagrange multiplier or penalty function techniques.

#### 1.4.3 Quasilinearization Methods

The technique of successive approximations in policy space is used in this method. Baldwin and Sims-Williams [23] describe an al-

gorithm which is applicable when system equations are not explicitly time-dependent and the criterion function is quadratic. The system of state equations (1.1) is used to generate an initial trajectory  $\{\bar{x}_k\}$ . The system equations are then decomposed as

$$\underline{x}_{k+1} = A_k \underline{x}_k + C_k \underline{m}_{k-k} + \underline{q}_1(\bar{x}, \bar{m}, k) \quad (1.33)$$

where  $A_k$  is an  $n \times n$  matrix and  $C_k$  is an  $n \times r$  matrix. The nonlinear terms are included in  $\underline{q}_1$ . The linearized model (1.33) is used to obtain a new nominal control sequence; a solution of discrete Riccati equations is used to generate this control sequence. This procedure is repeated iteratively to obtain an optimal control sequence. But constraints of the form (1.3) and (1.4) are not readily incorporated in the method. The authors state that the practicability of the scheme in individual cases must be investigated.

Another quasilinearization approach, based on the fact that a closed form of solution of a system of linear difference equations is readily generated, has been described by Lee [24]. In this method, the points on a trajectory, obtained after  $i+1$  iterations, are related to those on the previous trajectory by

$$\begin{aligned} \underline{x}_{k+1}^{i+1} &= \underline{q}(\underline{x}_k^i, \underline{m}_k^i, k) + (\underline{q}_x)^T (\underline{x}_k^{i+1} - \underline{x}_k^i) + (\underline{q}_m)^T (\underline{m}_k^{i+1} - \underline{m}_k^i) \\ &= (\underline{q}_x)^T (\underline{x}_k^{i+1}) + \{ \underline{q}(\underline{x}_k^i, \underline{m}_k^i, k) - (\underline{q}_x)^T \underline{x}_k^i + (\underline{q}_m)^T (\underline{m}_k^{i+1} - \underline{m}_k^i) \} \end{aligned} \quad (1.34)$$

where higher-order terms are ignored in this approximation. This can also be written as

$$\underline{x}_{k+1}^{i+1} = A_k \underline{x}_k^{i+1} + \underline{p}_k^{i+1} \quad (1.35)$$

Initially, a nominal control sequence is assumed. Then, at each iteration, the point  $\underline{x}_k$  can be obtained as a sum of a vector independent of control and a convolution-summation vector which varies with control. At each stage, the change in the latter is calculated as a function of the control action. Thus, if the performance criterion is only a function of  $n_p$  of the  $n$  state variables, the dimensionality problem can be overcome in many cases. The constraints on control variables can be treated easily, but not those on the state variables which lose their identity in the process of transformations involved. The author reportedly obtained convergence for several chemical engineering problems.

#### 1.4.4 Direct Methods

The direct methods, while using the dynamic programming technique of determining the optimal cost at a number of points in state space, seek to tailor the rapid-access core memory requirements to the available facilities. In the method of state increment dynamic programming [25], the system dynamic equations are discretized using a variable sampling interval. The allowed state-space region is divided

into a number of hyper-rectangular blocks. Time constitutes one coordinate of these blocks, and the length of each block is  $\Delta t$  in this direction. To compute the minimum cost at a given point, the time interval, for each control, is chosen such that one or more variables, including time, change by one increment, and no variables change by more than one increment. Therefore, only values of the optimal cost at points adjacent to a block are required for interpolation; the number of points and the size of the block are preassigned.

In a paper by Wong and Luenberger [26], a slightly different approach is used. The state space is divided into regions. While calculating the optimal cost at points within a region, only those controls which keep the point at the next stage within a preferred region are allowed. Thus, this method also eliminates the need to store the optimal cost in the whole of allowed state space at the next stage in rapid-access memory.

In both the methods, once the optimal cost at the selected points has been calculated at all the stages, the optimal solution can be obtained, for any initial point, in one iteration. However, the interpolation between neighboring points entails repeated calculation of the coefficients of the approximating function. Also, the optimal cost is calculated over all of the admissible state space. Thus, though the rapid-access core memory requirements are not excessive, computation time can be.

1.4.5    A Comparison

The second-variation methods by-pass the problem of dimensionality in an effective manner. In the case of direct methods, while the rapid-access memory requirements are manageable, the methods are not automatically applicable to problems with large state vectors. Savings in computation time depend on proper division of the state space into regions and proper ordering of the regions for processing. In both of the direct methods, no solutions to problems with state-vector dimension larger than four have been cited.

The decomposition methods are applicable to a special class of problems in which the matrix  $C$  of (1.29) is diagonal. If  $C$  is a square matrix with distinct eigenvalues, it is possible to transform the system equations into this special mold. If  $C$ , too, is partitioned into two matrices, one with diagonal elements and the other with off-diagonal elements of  $C$ , the matrix containing off-diagonal elements of  $C$  could be grouped with the nominal control sequence during the iterative minimization, but the convergence of this method to an optimal solution is questionable. In fact, this latter method was used by this writer to solve a second-order two-stage optimization problem that has been solved by Kuo [33] using discrete matrix Riccati equations. Convergence to the given solution was not possible for this problem.



The first quasilinearization method also is hindered by problems of convergence and has, therefore, limited application. It does not appear to have been applied to practical multidimensional problems. For systems with linear dynamics, Lee's method seems highly applicable and apparently suffers from no convergence problems. None of these quasilinearization methods, however, attempt to deal directly with the problems of bounds on the state variables, though some problems with bounds on control variables have been solved.

#### 1.5 Other Related Research

An early alarm about the dimensionality difficulties, and possible excessive computation requirements when dynamic programming is used for dynamical optimization, was sounded by Bellman [8,12]. For computational feasibility, the technique of polynomial approximation and successive approximations were advocated by him. Larson's state increment dynamic programming was one attempt at a systematic procedure for application of dynamic programming to multidimensional problems. But it has limited use for the solution of problems having more than four variables [25]. Among the methods employing successive approximations, differential dynamic programming has been successfully applied [52] to solve a third-order orbit transfer problem. The only related technique to be applied to a problem with a large state vector, one of eighth-order, is that of functional approximation described by Durling [27]. He uses Legendre polynomials for the functional approximation.

The optimization of continuous-time control problems with state-variable inequality constraints has received considerable attention [53,54,55]. In one such problem [53], the constraints are on the state variables that do not explicitly involve the control variable. The necessary conditions for an extremal solution to the general  $q^{\text{th}}$  order inequality constraint problem, where constraints are adjoined to the performance index are given in [53]; solutions to two analytical examples are presented. Another method is that of converting the problem to an unconstrained one by adjoining the state-variable constraints together with respective penalty functions to the performance index. A transformation technique involving slack variables, which effectively increases the problem dimension, is given by Jacobson and Lele [54]. However, the control variable is assumed to be unconstrained. Lee [24], in a treatment of the discrete-time problem, uses the quasilinearization technique to overcome dimensionality difficulties; he has noted at the same time the difficulty of accounting for inequality constraints on state variables.

#### 1.6 This Work--An Outline

In this work, the task of devising an optimization technique for multidimensional systems with inequality constraints is undertaken. The technique is based on principles of dynamic programming. No equivalent procedure of this type is known to the author.

In concurrence with current advances in array-processing computing [45,50], it is desirable that appropriate algorithms be developed to take advantage of parallelism in solution approaches for difficult computational problems. The algorithms that are developed along these lines may even make possible the real-time on-line computation of general optimal control. An evaluation of the present method for this purpose is part of this work.

The main emphasis is placed on resolving the difficulties relating to dimensionality and state-variable constraints, which are encountered when the method of dynamic programming is used. To impart computational feasibility to the method, the cost function at each stage is approximated by a quadratic polynomial. It is realized that one quadratic polynomial cannot, in general, be prescribed to accurately approximate the cost function in whole of state space; the method of region-limiting strategies is, therefore, employed. The regions in which a particular polynomial should be used can be restricted to be of a size judged appropriate to reduce the error in the approximation. The details of the technique are presented in Chapter 2. The strategy to be used when the state variables are bounded is also described in Chapter 2.

The essence of the technique is the approximation scheme, the salient features of which are covered in Appendix A. The number

of points at which the cost function must be evaluated is only of the order of the square of the state vector dimension. These points are so selected that a minimal amount of computation is necessary to evaluate the coefficients of the approximating polynomial.

A computer program to effect the optimization has been written. The details of the computer routines are set forth in Chapter 3. The routines, as listed in Appendix B, are applicable for third- to tenth-order problems. But there is no such restriction on the use of the technique itself.

The characteristics of some array-processing and parallel-processing systems are briefly described in Chapter 4. Ways of exploiting parallelism in the technique of region-limiting strategies, and possible reductions in the processing time, form part of the subject matter in Chapter 4.

It should be noted that certain limitations are imposed on the problem as formulated in Section 1.2. These relate to control action--it is assumed to be scalar and is necessarily bounded. Also, only discretized values of the control are used in the computation. And finally, the control trajectories obtained for nonlinear problems are suboptimal in the sense that all such trajectories obtained by iterative computational procedures are suboptimal when a termination criterion is established to discontinue the optimization process if improvements in performance become nominal.

CHAPTER 2

FUNCTIONAL APPROXIMATION AND  
THE METHOD OF REGION-LIMITING STRATEGIES

## 2.1 Introduction

A combination of the direct method of dynamic programming and the technique of functional approximation [27] is presented in the method of this chapter. The approach retains much of the advantage of the direct method of dynamic programming in dealing with problems with bounds on the state and control variables; the functional approximation technique alleviates the dimensionality problems that hinder other direct methods. A polynomial in the normalized state vector  $\underline{z}$  is chosen as the approximating function for the optimal cost function. For accuracy, an approximating polynomial should include terms up to the order of significant partial derivatives of the original function [28]. However, in the absence of knowledge about the derivatives of the optimal cost function, a quadratic polynomial suffices as the approximating function if the domain of approximation is limited to a suitably small region of state space. The approximation is used in the neighborhood of a nominal trajectory about which the incremental cost is accurately approximated by terms of first- and second-order.

Emphasis is placed on an orderly and simple method to obtain the approximating polynomial. At the same time, any bounds imposed on the state variables are incorporated directly in the method. These bounds must be satisfied by an initial trajectory which is required to



























































































































































































































































































































































































