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

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

LIST OF TABLES

LIST OF FIGURES

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 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

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

The state vector $\underline{x}_k$ is an n x 1 vector at the k^{th} stage, $\underline{q}$ is an n-dimensional vector function, and $\underline{m}_k$ is an r x 1 vector control action at the k^{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 $x_1^k, x_2^k, \ldots, x_n^k$, and $m_1^k, m_2^k, \ldots, m_r^k$, respectively.

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

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

where f_k's are given real-valued stage costs. Given $\underline{x}_1$, a sequence of controls $\underline{m}_1, \underline{m}_2, \ldots, \underline{m}_{K-1}$ 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

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

for i = 1, 2, ..., n, and k - 1, 2, ..., K;

and

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

for i = 1, 2, ..., r, and k = 1, 2, ..., 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[1]:

$$B_{K-k}(\underline{x}_k) = \min_{\underline{m}_k \in U_k} \left\{ f_{k+1}\left[\underline{g}(\underline{x}_k, \underline{m}_k, k), \underline{m}_k\right] + B_{K-k-1}\left[\underline{g}(\underline{x}_k, \underline{m}_k, k)\right] \right\} \qquad (1.5)$$

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

[1] 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^{th} stage, the total number N_k of points at which B must be evaluated at the k^{th} stage is

$$N_k = \prod_{i=1}^{n} N_{i,k} \tag{1.6}$$

where $N_{i,k}$ is the number of points associated with the i^{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.

1.411 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):

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

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

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

and

$$\bar{V}_K(\bar{x}_K) = F(\bar{x}_K) \tag{1.9}$$

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

$$\underline{m}_k = \bar{\underline{m}}_k + \delta\underline{m}_k \tag{1.10}$$

$$\delta\underline{m}_k \triangleq \underline{\alpha}_k + \underline{\beta}_k^T \delta\underline{x}_k \tag{1.11}$$

and

$$\delta\underline{x}_k \triangleq \underline{x}_k - \bar{\underline{x}}_k \ . \tag{1.12}$$

Then

$$V_k(\underline{x}_k) = f_k(\underline{x}_k, \underline{m}_k) + V_{k+1}(\underline{x}_{k+1}) \tag{1.13}$$

where

$$\underline{m}_k = \bar{\underline{m}}_k + \underline{\alpha}_k + \underline{\beta}_k^T \delta\underline{x}_k \tag{1.14}$$

Also, let

$$V_k(\bar{\underline{x}}_k) \triangleq \bar{V}_k(\bar{\underline{x}}_k) + a_k \tag{1.15}$$

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

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

where $\bar{V}_k(\bar{\underline{x}}_k)$ is the cost function which results from control $\{\bar{\underline{m}}_k\}$ 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 $\{m_i\}$ and initial condition $\bar{x}_k$; then, within second-order variations, equations (1.15) and (1.16) are used to obtain

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

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

$$\begin{aligned} f_k(x_k, m_k) + \bar{V}_{k+1}(\bar{x}_{k+1}) = {} & f_k + V_{k+1}(\bar{x}_{k+1}) + a_{k+1} \\ & + \left\{\left[f_x^k\right]^T + \left[V_x^{k+1}(\bar{x}_{k+1})\right]^T(q_x^k)\right\}(\delta x_k) \\ & + \left\{\left[f_m^k\right]^T + \left[V_x^{k+1}(\bar{x}_{k+1})\right]^T(q_m^k)\right\}(\delta m_k) \\ & + \frac{1}{2}(\delta x_k)^T E(\delta x_k) + \frac{1}{2}(\delta m_k)^T F(\delta m_k) \\ & + (\delta m_k)^T G(\delta x_k) \end{aligned} \qquad (1.18)$$

where

$$E = f_{xx}^k + (q_m^k)^T V_{xx}^{k+1}(\bar{x}_{k+1})(q_x^k) + \left[V_x^{k+1}(\bar{x}_{k+1})\right]^T(q_{xx}^k) \qquad (1.19)$$

$$F = f_{mm}^k + (q_m^k)^T V_{xx}^{k+1}(\bar{x}_{k+1})(q_x^k) + \left[V_x^{k+1}(\bar{x}_{k+1})\right]^T(q_{mm}^k) \qquad (1.20)$$

and

$$G = f_{mx}^k + (q_m^k)^T V_{xx}^{k+1}(\bar{x}_{k+1})(q_x^k) + \left[V_x^{k+1}(\bar{x}_{k+1})\right]^T(q_{mx}^k) \qquad (1.21)$$

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

As given by (1.13), the right-hand members of (1.17) and (1.18) are equal. If $\delta\underline{m}_k$ is replaced by $\underline{\alpha}_k + \underline{\beta}_k^T \delta\underline{x}_k$ in (1.18) and coefficients of like powers of $\delta\underline{x}_k$ 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:

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

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

and

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

Expressions for $\underline{\alpha}_k$ and $\underline{\beta}_k$ for k = 1, 2, ..., K-1 are obtained by minimizing the right-hand side of (1.18) with respect to $\delta\underline{m}_k$. These are used to generate a new trajectory within a small region of the old trajectory. The new control is obtained using

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

The actual algorithm consists of the following steps:

1. For the given initial condition and a control sequence $\{\bar{\underline{m}}_k\}$, a nominal trajectory $\{\bar{\underline{x}}_k\}$ 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, ..., 0.

3. The change in the value of the performance criterion is calculated as

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

4. If ΔV is positive, i.e., the cost has increased, or if ΔV 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 ϵ; c is, in effect, the limiting ratio of change in performance measure to the estimated value for the specified value of ϵ (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 η 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 x 1 vectors. The state equations are assumed to be of the form

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

where A and C both are n x n matrices. Matrix A is decomposed as

$$A = T + D \tag{1.28}$$

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:

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

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

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

where the length of the $n_i \times 1$ vectors $\underline{x}^{(i)}_{k}$, $\underline{c}_i$, and $\underline{d}_i$ varies with the index n_i. n_i is the number of nonzero elements in the i^{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^{th} nonzero element in i^{th} row of T. This row consists of n_i nonzero elements in that row of T. $\underline{x}^{(i)}_{k}$ and $\underline{c}_i$ consist of those elements in $\underline{x}_k$ and i^{th} column of C, respectively, which correspond to the n_i nonzero elements in i^{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^{th} element in the product $D\underline{x}_k$.

To simplify notation, relation (1.30) is written as

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

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

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

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^{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 $\{\bar{\underline{x}}_k\}$. The system equations are then decomposed as

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

where A_k is an n x n matrix and C_k is an n x 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

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

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

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

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 Δt 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^{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

initiate the optimization process. Thus, an algorithm to obtain such an initial trajectory is also investigated.

2.2 The Recurrence Relations in Dynamic Programming

In an on-line control situation, the control should be applied in the feedback sense; at each stage, the control should be so selected as to minimize the cost from the present to the final stage. Although a forward dynamic programming procedure may be used to obtain an optimal trajectory, this approach is unrealistic for on-line control because the corresponding recurrence relations result in control at k^{th} stage being calculated to minimize the sum of costs of stages 1 to k. The backward dynamic programming technique, however, simulates the feedback situation exactly and is used in the development that follows.

The optimization problem for a nonlinear, discrete, deterministic process is defined by equations (1.1), (1.2), (1.3), and (1.4) of Chapter 1. In a control problem, values of the state variables may be specified at the initial point or the terminal point or both. It is assumed here that only the initial boundary value $\underline{x}_1$ is fixed. A scalar control action is used in the rest of this section; the relations for the vector case follow directly.

2.2.1 The Discrete Case

Let B_{K-k} denote the optimal cost function at the k^{th} stage,

$$B_{K-k}(\underline{x}_k) = \underset{m_k's \in U_k's}{\min} \left[\sum_{i=k+1}^{K} f_i(\underline{x}_i, m_{i-1}) \right] \tag{2.1}$$

The minimum of the performance measure J is $B_{K-1}(\underline{x}_1)$,

$$B_{K-1}(\underline{x}_1) = \underset{m_k's \in U_k's}{\min} \left[\sum_{k=2}^{K} f_k(\underline{x}_k, m_{k-1}) \right]$$

$$= \underset{m_k's \in U_k's}{\min} \left\{ f_2 \left[\underline{g}(\underline{x}_1, m_1, 1), m_1 \right] + \sum_{k=3}^{K} f_k(\underline{x}_k, m_{k-1}) \right\}$$

$$= \underset{m_1 \in U_1}{\min} \left\{ f_2 \left[\underline{g}(\underline{x}_1, m_1, 1), m_1 \right] + \underset{m_k's \in U_k's}{\min} \sum_{k=3}^{K} f_k(\underline{x}_k, m_{k-1}) \right\}$$

$$= \underset{m_1 \in U_1}{\min} \left\{ f_2 \left[\underline{g}(\underline{x}_1, m_1, 1), m_1 \right] + B_{K-2} \left[\underline{g}(\underline{x}_1, m_1, 1) \right] \right\} \tag{2.2}$$

In a similar manner,

$$B_{K-2}(\underline{x}_2) = \min_{m_2 \in U_2} \left\{ f_3 \left[\underline{q}(\underline{x}_2, m_2, 2), m_2 \right] + B_{K-3} \left[\underline{q}(\underline{x}_2, m_2, 2) \right] \right\} \tag{2.3}$$

The K-1 stage problem of (2.2) is exactly similar to the K-2 stage problem of (2.3). In general, the optimal cost function B for two consecutive stages is related by the difference equation

$$B_{K-k}(\underline{x}_k) = \min_{m_k \in U_k} \left\{ f_{k+1} \left[\underline{q}(\underline{x}_k, m_k, k), m_k \right] + B_{K-k-1} \left[\underline{q}(\underline{x}_k, m_k, k) \right] \right\} \tag{2.4}$$

for $k = K-1, K-2, \ldots, 1$. The boundary condition on B is

$$B_0(\underline{x}_K) = 0 \tag{2.5}$$

and, therefore, B_1 is obtained as

$$B_1(\underline{x}_{K-1}) = \min_{m_{K-1} \in U_{K-1}} f_K \left[\underline{q}(\underline{x}_{K-1}, m_{K-1}, K-1), m_{K-1} \right] \tag{2.6}$$

A compact form for (2.4) is

$$B_{K-k}(\underline{x}_k) = \min_{m_k \in U_k} \left[f_{k+1}(\underline{x}_{k+1}, m_k) + B_{K-k-1}(\underline{x}_{k+1}) \right] \tag{2.7}$$

2.2.2 The Continuous Case

Consider the case of continuous systems which are described by the set of differential equations

$$\dot{\underline{x}} = \underline{q}(\underline{x}, m, t) \tag{2.8}$$

and for which the performance criterion is

$$J = F\left[\underline{x}(t_f)\right] + \int_{t_0}^{t_f} L(\underline{x}, m, t)\, dt \tag{2.9}$$

where t denotes time, t_0 is initial time, and t_f is the final time; $\underline{q}$ is an n-dimensional vector of (possibly) nonlinear functions of $\underline{x}$, m, and t, and L and F are real-valued scalar functions.

For computations on a digital computer, discretization of the equations (2.8) and (2.9) is necessary [29]. Several methods for discrete approximations of continuous systems, to represent them by difference equations, are available [12,51]. A first-order, rectangular, discrete approximation is employed here and gives rise to

$$\underline{x}(t + \delta t) = \underline{x}(t) + \underline{q}(\underline{x}, m, t)\delta t \tag{2.10}$$

where δt is a small time increment. A similar approximation for the cost is

$$\int_{t}^{t+\delta t} L(\underline{x}, m, t)\, dt = L\left[\underline{x}(t), m(t), t\right]\delta t \tag{2.11}$$

If the time interval $(t_f - t_0)$ is divided into K equal intervals of width δt each, the performance criterion can be written as

$$J = F\left[\underline{x}(t_f)\right] + \sum_{k=0}^{K-1} L\left[\underline{x}(k\delta t),\ m(k\delta t),\ t_0 + k\delta t\right]\delta t \tag{2.12}$$

To simplify notation, (2.12) is expressed as

$$J = F(\underline{x}_K) + \sum_{k=0}^{K-1} L_k(\underline{x}_k,\ m_k) \tag{2.13}$$

The recurrence relation (2.7) assumes the form

$$B_{K-k}(\underline{x}_k) = \min_{m_k \in U_k} \left[L_k(\underline{x}_k,\ m_k) + B_{K-k-1}(\underline{x}_{k+1})\right] \tag{2.14}$$

for k = K-1, K-2, ..., 1. The starting B function is

$$B_0(\underline{x}_K) = F(\underline{x}_K) \tag{2.15}$$

The recurrence relations for cost on a trajectory, not necessarily optimal, can be similarly obtained. The cost function is denoted by $V_k(\underline{x}_k)$, as used in Section 1.4.1, and is the cost from stage k to the final stage K:

$$V_k(\underline{x}_k) = \left[F(\underline{x}_K) + \sum_{i=k}^{K-1} L_i(\underline{x}_i,\ m_i)\right] \tag{2.16}$$

A recurrence relation for V is:

$$V_k(\underline{x}_k) = \left[L_k(\underline{x}_k, m_k) + V_{k+1}(\underline{x}_{k+1})\right] \tag{2.17}$$

for k = K-1, K-2, ..., 0; with

$$V_k(\underline{x}_K) = F(\underline{x}_K) \tag{2.18}$$

2.3 Functional Approximation

In the absence of an analytical solution to the problem posed by (1.1) through (1.4), an approximate solution can be obtained by generating and storing values of the optimal cost function at a grid of points in state space. For every one of these points, the minimization indicated in (2.7) may be affected by the use of differential calculus. The difficulties encountered in doing this, however, are numerous because of constraints (1.3) and (1.4) and because of the dimensionality problem, as explained in Chapter 1. One way out of this dilemma is to use a functional approximation of the optimum cost function for the purpose of minimization. That is, consider an approximate version of equation (2.7):

$$B_{K-k}(\underline{x}_k) = \min_{m_k \in U_k} \left[f_{k+1}(\underline{x}_{k+1}, m_k) + \tilde{P}_{k+1}(\underline{x}_{k+1})\right] \tag{2.19}$$

where

$$\tilde{P}_{k+1}(\underline{x}_{k+1}) = B_{K-k-1}(\underline{x}_{k+1}) - \tilde{E}_{k+1}(\underline{x}_{k+1}) \tag{2.20}$$

$\tilde{E}_{k+1}$ is the error term, and $\tilde{P}_{k+1}$ is the functional approximation. The ideal error, of course, is zero.

Some of the desirable features that an approximating function should possess are:

a. It should show reasonable convergence to the optimal cost function.

b. It should not require excessive computer storage space.

c. The program to evaluate it should be computationally feasible.

Little advance information may be known about the optimal cost function for a nonlinear problem. Therefore, it is essential that a strategy be devised to make the approximation fit the optimal cost closely, especially in the neighborhood of the optimal trajectory, and to keep the error term negligible. The method of region-limiting strategies is presented to achieve these goals.

2.4 The Method of Region-Limiting Strategies

A version of this method using forward dynamic programming is described by Pierre [1]. However, in keeping with the object of developing a procedure for feedback application, the description of the method for the backward procedure is given here. To initiate the procedure, a control sequence $\{m_k^0\}$ is specified, and a nominal trajectory $\{\underline{x}_k^0\}$ is obtained therewith. The superscript here refers to the iteration number; the initial values belong to the 0^{th} iteration.

In general, the trajectory obtained after the i^{th} iteration is $\{\underline{x}_k^i\}$. Regions R_k^i are selected around the points $\underline{x}_k^i$, k = 2, 3, ..., K-1, of the trajectory. These regions are hyper-rectangular in shape. This is the strategy phase. Within each region, a representative number of points is specified. The number of such points depends on the available rapid-access memory. Each region size can be adjusted so that the points selected are indeed representative in the sense that a suitable polynomial approximation of the cost function is accurate in each region.

A dynamic programming phase is next. It starts from the final point of the current trajectory; B functions are calculated at an array of points S_k^i for k = 2, 3, ..., K-1. The point set S_k^i lies in the corresponding region R_k^i at each stage. Relation (2.7) assumes the form

$$B_{K-k}(\underline{x}_k) = \min_{m_k^i \in U_k^{i-1}} \left\{ f_{k+1}\left[\underline{g}(\underline{x}_k, m_k^i, k), m_k^i\right] + B_{K-k-1}\left[\underline{g}(\underline{x}_k, m_k^i, k)\right] \right\} \tag{2.21}$$

where U_k^{i-1} is such that $\underline{g}(\underline{x}_k, m_k^i, k) \in R_{k+1}^{i-1}$ for $i \geq 1$ and $\underline{x}_k \in S_k^{i-1}$.[1]

---

[1] In the detailed procedure (Section 2.6), an extrapolation region is allowed such that $\underline{g}(\underline{x}_k, m_k^i, k) \in \bar{R}_{k+1}^{i-1}$ where R_{k+1}^{i-1} is a subset of $\bar{R}_{k+1}^{i-1}$.

An n-dimensional interpolation can be used to obtain the values of B in the right-hand member of (2.21). Next, a new trajectory starting at the given initial point $\underline{x}_1$ is generated using the values of optimal cost calculated after the previous iteration and using (2.21). The points on the new trajectory are restricted to lie within or on the boundary of the region around the previous trajectory. If all the points are within these regions, the new regions that are defined are reduced in size in a systematic way, to be described.

2.5 Quadratic Approximating Function

While more sophisticated approximating functions for B [27, 30] can be used in the region R_k, a quadratic polynomial representation is proposed. In general, the number of points at which a real-valued scalar function of a vector $\underline{x}$ must be known in order to approximate it by a polynomial of s^{th} order is given by

$$N = (n + s)!/(n!s!) \tag{2.22}$$

where n is the dimension of the vector. For a quadratic polynomial, s = 2 in which case N = (n+1)(n+2)/2, which increases as the square of the order of the vector. When this is compared to the exponential relationship (1.6) for points required by straightforward methods of dynamic programming, it is evident that substantial savings in rapid-access storage can be obtained. At the same time, if the dimensions

of the regions are restricted so that cubic and higher-order terms are negligible, good convergence of the approximating function to the optimal cost function is obtained in the regions. A higher-order technique of approximation, because of larger number of base points, requires a larger core memory than the present scheme.

2.5.1 Coefficients of the Quadratic Polynomial

A real-valued quadratic polynomial in $\underline{x}$ can be written as

$$\tilde{P}(\underline{x}) = \underline{x}^T \tilde{A} \underline{x} + \tilde{\underline{b}}^T \underline{x} + \tilde{c} \qquad (2.23)$$

where $\tilde{A}$ is an n x n symmetric matrix, $\tilde{\underline{b}}$ is an n x 1 vector, and $\tilde{c}$ is a scalar. $\tilde{A}$, $\tilde{\underline{b}}$, and $\tilde{c}$ are all real-valued. The total number of unknowns to be determined, elements of $\tilde{A}$, $\tilde{\underline{b}}$ and $\tilde{c}$, is $(n+1)(n+2)/2$. Assume that values of the optimal cost function, hereafter referred to as the minimal cost function, are determined at $(n+1)(n+2)/2$ points, called base points, in a given R_k region. The necessary equations to calculate $\tilde{A}$, $\tilde{\underline{b}}$, and $\tilde{c}$ for this region are obtained by requiring that

$$B(\underline{x}) = \underline{x}^T \tilde{A} \underline{x} + \tilde{\underline{b}}^T \underline{x} + \tilde{c} \qquad (2.24)$$

for all $\underline{x} \in S_k$. If these equations are linearly independent, a unique solution for coefficients of the approximating polynomial can be obtained. Ordinarily the solution involves the inversion of a matrix of the order of $(n+1)(n+2)/2$.

In the actual procedure used here, however, the inversion is not required because of the way in which base points are specified. The base points at each stage are located on a hyper-rectangular region centered at a point on the trajectory. Also, a normalized vector $\underline{z}$ is used instead of $\underline{x}$ in the approximating polynomial. Let $2d_i$ be the width of the hyper-rectangular region along the i^{th} coordinate, and assume this region is centered around $\underline{x}_c$. The components of the normalized vector $\underline{z}$ are given as

$$z_i = (x_i - x_{c,i})/d_i \ , \quad i = 1, 2, \ldots, n \tag{2.25a}$$

Equivalently,

$$\underline{z} = D^{-1}(\underline{x} - \underline{x}_c) \tag{2.25b}$$

where D is the diagonal matrix of d_i's.

Relations (2.23) and (2.24) assume the form

$$\tilde{P}(\underline{x}) = P(\underline{z}) = \underline{z}^T A \underline{z} + \underline{b}^T \underline{z} + c \tag{2.26}$$

and

$$B(\underline{x}) = P(\underline{z}) = \underline{z}^T A \underline{z} + \underline{b}^T \underline{z} + c \tag{2.27}$$

respectively. A, $\underline{b}$, and c of the new approximating polynomial are related to $\tilde{A}$, $\tilde{\underline{b}}$, and $\tilde{c}$ by

$$A = D\tilde{A}D \tag{2.28}$$

$$\underline{b} = 2\underline{x}_c^T\tilde{A}D + \tilde{\underline{b}}^T D \tag{2.29}$$

$$c = \tilde{c} + \tilde{\underline{b}}^T\underline{x}_c + \underline{x}_c^T\tilde{A}\underline{x}_c \tag{2.30}$$

The normalized z_i's at the base points are either 1's, -1's, or 0's. This greatly simplifies calculation of A, $\underline{b}$, and c. The details of the approximation scheme are described in Appendix A.

2.6 The Optimization Procedure

In the initialization of the procedure, the first step is quantization of the control action. The quantization step-size may be specified by taking into account the coarseness of the desired controller and the length of time which can be spent on calculation of an optimal control. The relation between time spent and the number of control steps between M_{min} and M_{max} is approximately linear.

Next, a nominal control sequence $\{\underline{m}_k^0\}$ has to be specified to obtain an initial trajectory $\{\underline{x}_k^0\}$. Both $\{\underline{m}_k^0\}$ and $\{\underline{x}_k^0\}$ must be consistent with the constraint sets (1.3) and (1.4). From that point on, strategy phases and the dynamic programming phases of the method of region-limiting strategies are interwoven for the calculation of the cost function at the base points.

2.6.1 Calculation of Minimal Cost Functions

Starting at stage K-1, (n+1)(n+2)/2 base points are obtained in accordance with the interpolation scheme described in Appendix A. For each of these points, one quantized control action which minimizes the final stage cost

$$B_1(\underline{x}_{K-1}) = \min_{m_{K-1}^i} \min_{U_{K-1}^{i-1} \in} \left\{ f_K \left[\underline{q}(\underline{x}_{K-1}, m_{K-1}^i, K-1), m_{K-1}^i \right] \right\} \tag{2.31}$$

is obtained, and the corresponding costs are stored for use in obtaining the approximating polynomial for the next stage.

After the coefficients of the approximating polynomial at stage K-1 are calculated (as per Appendix A), the base points at stage K-2 are obtained as at the $(K-1)^{th}$ stage, and the minimum cost function is calculated at these points. The recurrence relations used for stages 2 to K-2 are

$$B_{K-k}(\underline{x}_k) = \min_{m_k^i} \min_{U_k^{i-1} \in} \left\{ f_{k+1} \left[\underline{q}(\underline{x}_k, m_k^i, k), m_k^i \right] + P_{k+1}^i(\underline{z}_{k+1}) \right\} \tag{2.32}$$

where U_k^{i-1} is such that $\underline{q}(\underline{x}_k, m_k^i, k) \in \bar{R}_{k+1}^{i-1}$ for $i \geq 1$ and $\underline{x}_k \in S_k^{i-1}$.

Some terms in (2.32) need clarification. Components of the vector $\underline{z}_{k+1}$ are obtained from $\underline{x}_{k+1}$ using

$$z_{j,k+1} = (x_{j,k+1} - x_{j,k+1}^{i-1})/d_j \tag{2.33}$$

for $j = 1, 2, \ldots, n$; $2d_j$ is the width along the j^{th} coordinate of the hyperrectangular region on which the base points are located. Here the extrapolation region $\bar{R}$ is introduced and is of width $\eta \cdot 2d_j$ along the j^{th} coordinate, where η is a constant greater than 1.

In the examples in this thesis, an initial value of $\eta=2$ was used. This choice was based on a preliminary study in which quadratic approximation for several nonlinear multidimensional functions (see Appendix C) were obtained with the base points located on a region of dimension $2d_j$, $j = 1, 2, \ldots, n$. The resulting error in the extrapolated values in a region of double the dimensions was found to be less than two percent.

The dimensions of $\bar{R}_k^i$ are computed on the basis of the d_j's. The value of η can be selected to meet requirements of a particular problem. Also, if the number of stages is very large, sufficient error can accumulate to give an inaccurate approximation. To guard against accuracy-reducing factors, a provision is made for reducing the value of η if the minimum cost at any of the base points becomes negative. Negative cost values are discarded while determining the minimum cost.

By iterative application of the above procedure, the discrete values of B for the base points at stages 2 to (K-1) are calculated and stored. It is possible that for some base point no control can be found to satisfy $\underline{q}(\underline{x}_k, m_k, k) \varepsilon \bar{R}_{k+1}^{i-1}$. In such a situation, the dimensions of

R_{k+1}^{i-1} and $\bar{R}_{k+1}^{i-1}$ are increased by a factor of 1.1, and base points at that stage are relocated. Values of B are also calculated anew. The region size can be progressively increased up to twice the original value. However, if an increase at $(k+1)^{th}$ stage further necessitates an expanded region at $(k+2)^{th}$ stage, R_k regions are reduced in size, as in a strategy phase. In the computer program this step is implemented by reading in the next set of d_i's. The dynamic programming phase is then reinitiated.

2.6.2 Updating the Trajectory

After the coefficients of the P_{k+1}^{i} polynomial have been generated, the new trajectory is obtained by an additional minimization at each stage:

$$B_{K-k}(\underline{x}_k^i) = \min_{m_k^i \in U_k^{i-1}} \left\{ f_{k+1}\left[\underline{g}(\underline{x}_k^i, m_k^i, k), m_k^i\right] + P_{k+1}^{i}(\underline{z}_{k+1}) \right\} \qquad (2.34)$$

which gives $\underline{x}_{k+1}^i = \underline{g}(\underline{x}_k^i, m_k^i, k) \in \tilde{R}_{k+1}^{i-1}$ for $i \geq 1$ and $k = 1, 2, \ldots, K-1$, starting with $\underline{x}_1^i$ being equal to the given $\underline{x}_1$.

The extrapolation region $\tilde{R}$ is different from $\bar{R}$ described in Section 2.6.1. At each iteration, the size of the region $\bar{R}$ is dependent on the current dimensions of the region R in which the base points are located. This is done to make the approximation accurate, as the

error in approximation, in this part of the optimization procedure, is cumulative. $\tilde{R}$, however, corresponds to the initial dimensions of R. It is assumed that, as the domain size is reduced, the minimal cost function resembles a quadratic approximating polynomial more closely than it did before. Therefore, with the narrowing down of the domain, the extrapolation regions are not made smaller. At the same time, this allows a comparatively larger domain in which to locate the new trajectory.

At first only those controls which keep the point at the next stage within the region $\tilde{R}$ are used for minimization. However, if no such control action can be found in the range $[M_{min}, M_{max}]$, the extrapolation region is enlarged up to twice its original size. This is accomplished by changing η to 3, and then to 4 if necessary. If a feasible control is still not possible, the present procedure is discontinued. A switch is made to the strategy phase in which smaller d_i's are selected, and another dynamic programming phase is then initiated.

The points $\underline{q}(\underline{x}_k^i, m_k^i, k)$ must satisfy the bounds (1.4) on the bounded components of $\underline{x}_{k+1}$. It can happen that, at stage k, no control from the set of quantized controls can keep the point at $(k+1)^{th}$ stage within the specified constraint boundaries. In such a case, the control at a previous stage is modified by one quantization step at a

time so that the point $\underline{x}_k$ satisfies tighter constraints on the bounded components. An explanation of the way this works is in order.

A term, order of control, is introduced: If the i^{th} row of state equation (1.1) explicitly contains the control m, the i^{th} state variable is said to have zero order of control. Otherwise, the order of the i^{th} variable is the minimum number of sample periods which must elapse before another state variable, itself influenced by a specific control action, affects the i^{th} variable. The order of control is important for the state variables which are bounded. Let r_i equal the order of control for the i^{th} state variable. In order for m_j to affect $x_{i,k+1}$, j must be less than or equal to $(k - r_i)$.

In the computer program written to obtain the optimal control trajectories, the order of control for each bounded state variable is specified during the initialization phase. It is assumed that each bounded state variable varies monotonically with the control action, between its minimum and maximum limits. Appropriate changes can then be made in the control action to obtain a trajectory away from the bounds. The program can be modified to account for the cases where a state variable does not follow the above restriction with respect to the control action.

Once the new trajectory has been generated, the new performance measure is compared with the previous value. Three cases can

occur: (1) the new performance measure is larger than the old one; (2) it is less than the old value, but the rate of change is less than a prespecified value; and (3) it is less than the old value, and the rate of change is larger than the specified value. In case (3) the optimization procedure is repeated. If (2) occurs, the new trajectory is tested for closeness to the previous trajectory. If the new trajectory lies within the R_k domains of the old trajectory, the domain size is reduced by a factor of 1/2 to 1/5, and the optimization procedure is repeated. The same domain-reduction step is taken if case (1) occurs, but here the regions are centered around the old trajectory.

2.7 An Initial Trajectory

A judicious guess of an initial trajectory is possible. It should be recognized, however, that it may be difficult to guess at a trajectory satisfying constraints (1.3) and (1.4). A complete program to obtain an initial trajectory has been written. The set of quantized controls, as defined in Section 2.6, is used in this program, too. The desired values, if any, of the state variables are recognized. At each stage the controls are selected such that the state variables at the next stage meet two conditions: First, they assume values near to the desired values; and second, they conform to the bounds on state variables. It can occur, however, that no quantized control is able to satisfy the latter condition at a given stage. The controls at prev-

ious stages are then changed so as to move the trajectory farther from the bounds.

2.8 Convergence of the Solution

During the i^{th} iteration, a new trajectory is found by minimizing B such that $B^{i}_{K-k} \leq B^{i-1}_{K-k}$ in the region $\tilde{R}^{i-1}_{k}$. The region $\tilde{R}^{i-1}_{k}$ includes the point $\underline{x}^{i-1}_{k}$ at its center. Therefore, every iteration should result in a value of the performance measure which is the same or better than the value obtained after the previous iteration. In a well-posed problem, therefore, a converging set of performance values is obtained.

For a problem with linear state equations, with a quadratic performance measure, and with no bounds on state or control variables, a quadratic polynomial representation of the minimal cost function is its true representation, and the optimal solution can be obtained in one iteration, subject to round-off and discretization errors. In a given nonlinear problem, extrapolation errors may be significant, and it can occur that the value of performance measure for a possible new trajectory is larger than the previous value, in which case no trajectory update is made, but the R_k regions are reduced in size; the size of the extrapolation regions is correspondingly reduced for computing costs at the base points, but not for computing a new trajectory.

At every iteration, a search for a new trajectory, with a smaller performance measure, is made in the extrapolation regions around the present trajectory. Therefore, the initial nominal trajectory and the initial domain size affect the number of iterations required to obtain an optimal solution. If the initial trajectory is in the vicinity of an optimal trajectory, convergence can be rapid because the minimal cost function is likely to be more like a quadratic polynomial in this region.

If the i^{th} state variable has zero order of control, one possible choice of the initial domain size along the i^{th} coordinate is

$$2d_{i,k} = \left|\underline{q}(\underline{0}, M_{max}, k)\right| \tag{2.35}$$

But other factors applying to specific problems generally influence the selection of $d_{i,k}$'s.

2.9 Continuously Variable Control

The method described above does have one shortcoming. Very fine quantization of control action can be used only at the cost of increased execution time. It is not easy to circumvent this difficulty. One way would be to calculate a minimizing control sequence using coarse quantization; final steps could then be used only around the best coarse controls.

On the other hand, in the case of a problem with no state variable constraints, differential calculus can be applied usefully to obtain a new control trajectory. The method advocated is as follows: An optimal control sequence $\{\bar{m}_k\}$ using coarse quantization and the corresponding trajectory $\{\bar{x}_k\}$ are found first. Let $\{m_k\}$ be an optimal control sequence not using quantized controls, and $\{x_k\}$ the corresponding optimal trajectory. The stage cost for the new trajectory is obtained by a Taylor series expansion around the old trajectory and is given by[2]

$$\begin{aligned} f(x_{k+1}, m_k) = f &+ (f_x)^T q_x \delta x_k + f_m \delta m_k + (q_m)^T f_x \delta m_k \\ &+ (\tfrac{1}{2})(\delta x_k)^T \left[(q_x)^T f_{xx} q_x + (f_x)^T q_{xx} \right] \delta x_k \\ &+ (\tfrac{1}{2}) \left[f_{mm} + 2(f_{mx})^T q_m + (q_m)^T f_{xx} q_m + (q_{mm})^T f_x \right] (\delta m_k)^2 \\ &+ (\delta x_k)^T \left[(q_x)^T f_{mx} + (q_x)^T f_{xx} q_m + (q_{xm})^T f_x \right] \delta m_k + O(\epsilon^3) \end{aligned} \tag{2.36}$$

The unspecified arguments are $\bar{x}_{k+1}$ and $\bar{m}_k$ for f, and $(\bar{x}_k, \bar{m}_k, k)$ for q.

---

[2] To simplify notation, the subscript k on f is omitted here.

Similarly, the approximating polynomial for the minimal cost function can be expanded as

$$P(\underline{z}_{k+1}) = P + (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_x \delta \underline{x}_k + (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_m \delta m_k$$

$$+ (\tfrac{1}{2})(\delta \underline{x}_k)^T \left[(\underline{q}_x)^T \left(\frac{\partial \underline{z}}{\partial \underline{x}}\right)^T P_{zz} \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_x + (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_{xx} \right] \delta \underline{x}_k$$

$$+ (\tfrac{1}{2}) \left[(\underline{q}_m)^T \left(\frac{\partial \underline{z}}{\partial \underline{x}}\right)^T P_{zz} \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_m + (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_{mm} \right] (\delta m_k)^2$$

$$+ (\delta \underline{x}_k)^T \left[(\underline{q}_x)^T \left(\frac{\partial \underline{z}}{\partial \underline{x}}\right)^T P_{zz} \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_m \right.$$

$$\left. + (\underline{q}_{mx})^T (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \right] \delta m_k + 0(\epsilon^3) \qquad (2.37)$$

The argument of P is $\bar{\underline{z}}_{k+1}$, and that of $\underline{q}$ is $(\bar{\underline{x}}_k, \bar{m}_k, k)$.

Because the trajectory $\{\bar{\underline{x}}_k\}$ is an optimal trajectory, obtained by using coarse quantization of the control action, it is possible that the change in control action, without using quantization, will be less than or of the order of one control step. Under this condition, the cost can be represented by a Taylor series expansion up to second-order terms, and higher-order terms can be neglected. Thus,

$$B_{K-k}(\underline{x}_k) = \min_{\delta m_k} \left\{ f + P + \left[(f_x)^T \underline{q}_x + (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_x \right] \delta \underline{x}_k \right.$$

$$\left. + \left[f_m + (\underline{q}_m)^T f_x + (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_m \right] \delta m_k \right.$$

(cont.)

$$+ \left(\frac{1}{2}\right)(\delta \underline{x}_k)^T \left[(\underline{q}_x)^T \left(\frac{\partial \underline{z}}{\partial \underline{x}}\right)^T P_{zz} \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_x \right.$$

$$+ (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_{xx} + (\underline{q}_x)^T f_{xx} \underline{q}_x + (f_x)^T \underline{q}_{xx} \left.\right] (\delta \underline{x}_k)$$

$$+ \left(\frac{1}{2}\right) \left[(\underline{q}_m)^T \left(\frac{\partial \underline{z}}{\partial \underline{x}}\right)^T P_{zz} \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_m \right.$$

$$+ (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_{mm} + f_{mm} + (\underline{q}_{mm})^T f_x$$

$$\left. + 2(f_{mx})^T \underline{q}_m + (\underline{q}_m)^T f_{xx} \underline{q}_m \right] (\delta m_k)^2$$

$$+ (\delta \underline{x}_k)^T \left[(\underline{q}_x)^T \left(\frac{\partial \underline{z}}{\partial \underline{x}}\right)^T P_{zz} \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_m \right.$$

$$+ (\underline{q}_{mx})^T (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} + (\underline{q}_x)^T f_{mx}$$

$$\left. + (\underline{q}_x)^T f_{xx} \underline{q}_m + (\underline{q}_{xm})^T f_x \right] \delta m_k \Big\} \qquad (2.38)$$

By differentiation, δm_k can be calculated from

$$\delta m_k = \left\{ \left[f_m + (\underline{q}_m)^T f_x + (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_m \right] \right.$$

$$+ (\delta \underline{x}_k)^T \left[(\underline{q}_x)^T \left(\frac{\partial \underline{z}}{\partial \underline{x}}\right)^T P_{zz} \frac{\partial \underline{z}}{\partial \underline{x}} \underline{q}_m \right.$$

$$\left. + (\underline{q}_{mx})^T (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} + (\underline{q}_x)^T f_{mx} + (\underline{q}_x)^T f_{xx} \underline{q}_m \right.$$

(cont.)

$$+ (\underline{g}_{xm})^T \underline{f}_x \Big] \Big\} / \Big[(\underline{g}_m)^T \left(\frac{\partial \underline{z}}{\partial \underline{x}}\right)^T P_{zz} \frac{\partial \underline{z}}{\partial \underline{x}} \underline{g}_m$$

$$+ (P_z)^T \frac{\partial \underline{z}}{\partial \underline{x}} \underline{g}_{mm} + \underline{f}_{mm} + (\underline{g}_{mm})^T \underline{f}_x$$

$$+ 2(\underline{f}_{mx})^T \underline{g}_m + (\underline{g}_m)^T \underline{f}_{xx} \underline{g}_m \Big] \qquad (2.39)$$

m_k is then given by

$$m_k = \bar{m}_k + \epsilon \delta m_k \qquad 0 \leq \epsilon \leq 1$$

where ϵ is selected as follows: if $\bar{m}_k = M_{max}$ and $\delta m_k > 0$, $\epsilon = 0$; if $\bar{m}_k = M_{min}$ and $|\delta m_k| < 0$, $\epsilon = 0$; if $M_{min} \neq \bar{m}_k \neq M_{max}$ and $|\delta m_k|$ is less than the coarse quantization step, $\epsilon = 1$; and if $M_{min} \neq \bar{m}_k \neq M_{max}$ and $|\delta m_k|$ is greater than one step, ϵ is selected so that $\epsilon \delta m_k$ equals sign(δm_k) multiplied by the coarse step size.

For evaluation of δm_k, $\delta \underline{x}_k = \underline{0}$ at $k = 1$. For $k = 2, 3, \ldots, K$, $\delta \underline{x}_k$ can be calculated as

$$\delta \underline{x}_k = \underline{x}_k - \bar{\underline{x}}_k \qquad (2.40)$$

where $\underline{x}_k$ is obtained by use of m_{k-1} and equation (1.1). $\frac{\partial \underline{z}}{\partial \underline{x}}$ is the diagonal matrix consisting of reciprocals of corresponding domain sizes at the time of the last iteration. The terms P_z and P_{zz} are obtained from the coefficients of the approximating polynomial at each

stage. This leaves the evaluation of the partial differentials of q_j; a subroutine of the computer program can be used for this purpose.

2.10 Conclusions

In this chapter an exposition of the method of region-limiting strategies is given. The technique is built around three principal features. The first of these is the strategy used to choose the region sizes and the base points in the regions. The second is the quadratic polynomial approximation of the minimal cost, including the technique of its evaluation. And third is the strategy to handle inequality constraints on the state variables, there being no limitation on the number of such variables.

The computation of minimal costs involves a substantial amount of calculations, if the number of discretized controls is large. In comparison, the second-variation techniques require fewer minimizations, in general, but do not effectively deal with state-variable inequality constraints. The method described should find favor for problems with such constraints, and for those in which a finite set of control actions is specified.

No qualifications for an initial trajectory are specified except that it should be a feasible one. The optimization procedure

is the same whatever the initial trajectory. Convergence to an optimal solution depends on the accuracy of the approximating polynomial, and the accuracy of the approximating polynomial depends on the dimensions of the $\bar{R}_k$ regions. If a higher-order approximating polynomial is used, it may be possible to use larger R_k regions, and as a result the convergence to an optimal trajectory may be faster. But keeping in view the increased demands on the core memory the use of a quadratic polynomial is justified.

CHAPTER 3

ALGORITHMS AND COMPUTATIONAL RESULTS

3.1 Introduction

A description of the method of region-limiting strategies is given in Chapter 2. The applicability of this technique for solving trajectory optimization problems is demonstrated in this chapter.

A computer program has been written for solving problems involving from three to ten state variables and a scalar, bounded control action; some or all of the state variables may be bounded. The program consists of a set of subroutines. Four of the subroutines are general purpose. MINMUM is the main subroutine for accomplishing the minimization in recurrence relations (2.32), GRID is for computing the coordinates of the base points, INPOL1 is for computing the coefficients of the approximating polynomial, and INPOL2 is for evaluating the polynomial at a given point. In addition to the general subroutines, three other subroutines are required and are modified for each specific problem. One of these (FOSTEQ) is for specifying the state equation (1.1), another (PRFORM) is for computing the performance measure J, and the third (RETFUN) is for computing one stage costs. Another subroutine (TRJTRY) is available for obtaining an initial feasible trajectory. All of these computer routines are listed in Appendix B, in addition to being described in more detail in the following sections.

The control action is quantized, between its upper and lower bounds, into a set of discrete values. A direct search procedure is employed to obtain the minimizing control action in (2.32). The minimal costs at the base points are stored in the first part of the dynamic programming procedure; in the second part, these minimal costs are utilized to compute an optimal trajectory, constrained to lie in an extrapolation region.

Questions are raised because of the iterative nature of the process: What are the factors that affect the rate of convergence? How many iterations are required before a trajectory sufficiently close to the optimal trajectory is obtained? A partial answer to such questions is obtained by the numerical results in Sections 3.3 and 3.4.

3.2 The Computer Program

MINMUM is the principal subroutine of the program. It is linked to the main calling routine in which the dimensions of all the variables are specified. The set of quantized control actions in the range $[M_{min}, M_{max}]$ is generated in this routine.

A flowchart for the subroutine MINMUM is given in Figure 3.1. It follows from the algorithm described in Chapter 2. Some changes, however, have been incorporated to prevent accumulation of large errors in the approximated values, errors which could result in the costs becoming negative at some of the points. This condition

can arise from the following factors: the monomials of the approximating polynomial can be weighted unevenly because the specified initial grid spacing may be too large; or the grid spacing may be decreased too soon, even though the present trajectory is too far from an optimal trajectory to give an accurate approximation in the extrapolation region.

During the first part of the dynamic programming procedure, a combination of two factors--namely, the point at the next stage falling outside the extrapolation region, and the approximated cost becoming negative--can reduce the set of admissible controls to an empty set. In such a case, if the cost at the base point was negative for any of the nonadmissible control actions, the grid spacing is reduced and the optimization procedure is repeated. Otherwise, as explained in Subsection 2.6.1, the base points at the next stage are relocated with a larger grid spacing, up to twice the original value, with a corresponding increase in the size of the extrapolation region for the present stage.

While determining the next optimal trajectory, an additional condition on an admissible control action is that the point at the next stage be within the specified state variable bounds. The course of action, when proceding normally, is outlined in the flow table.

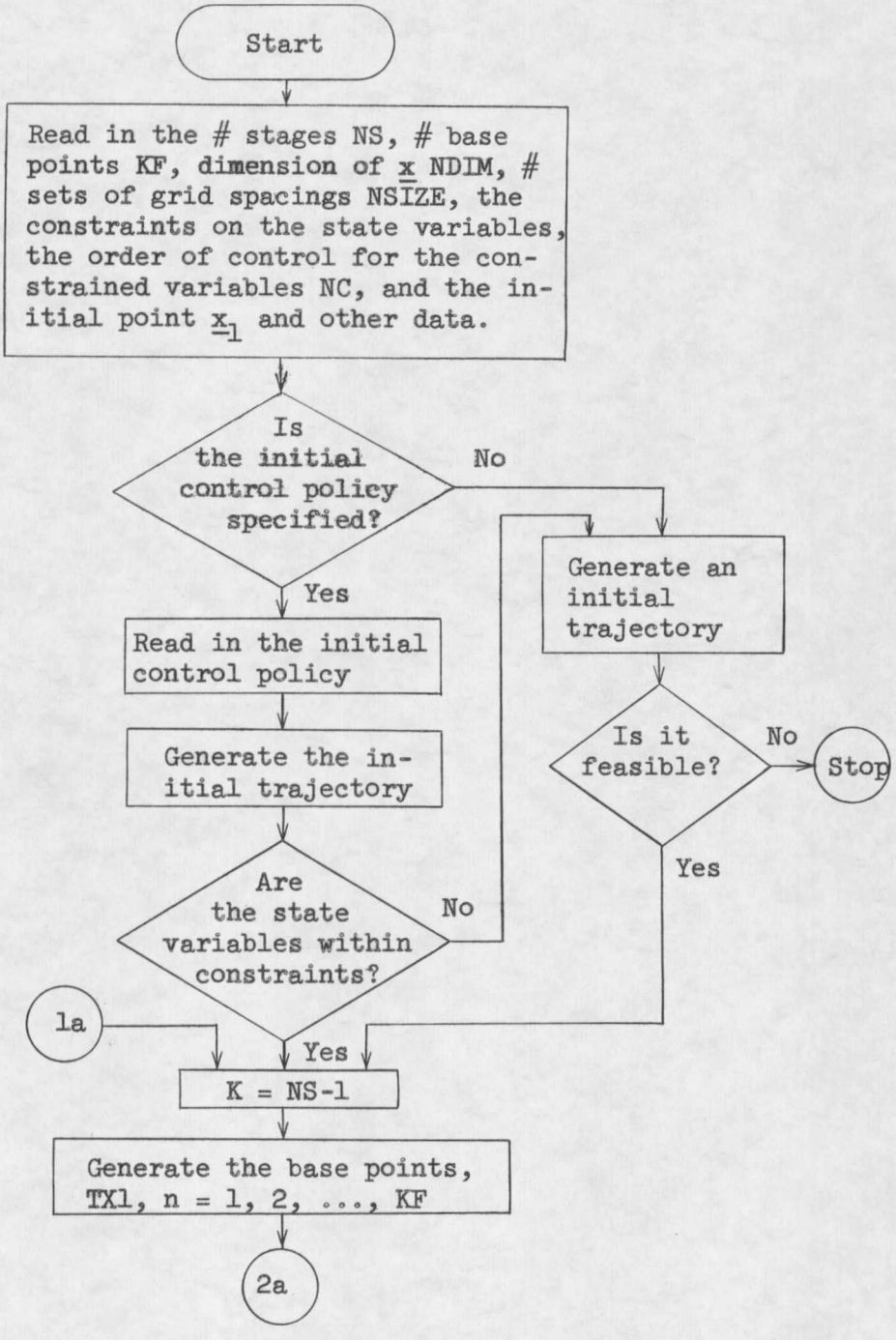

Figure 3.1. Flowchart for subroutine MINMUM.

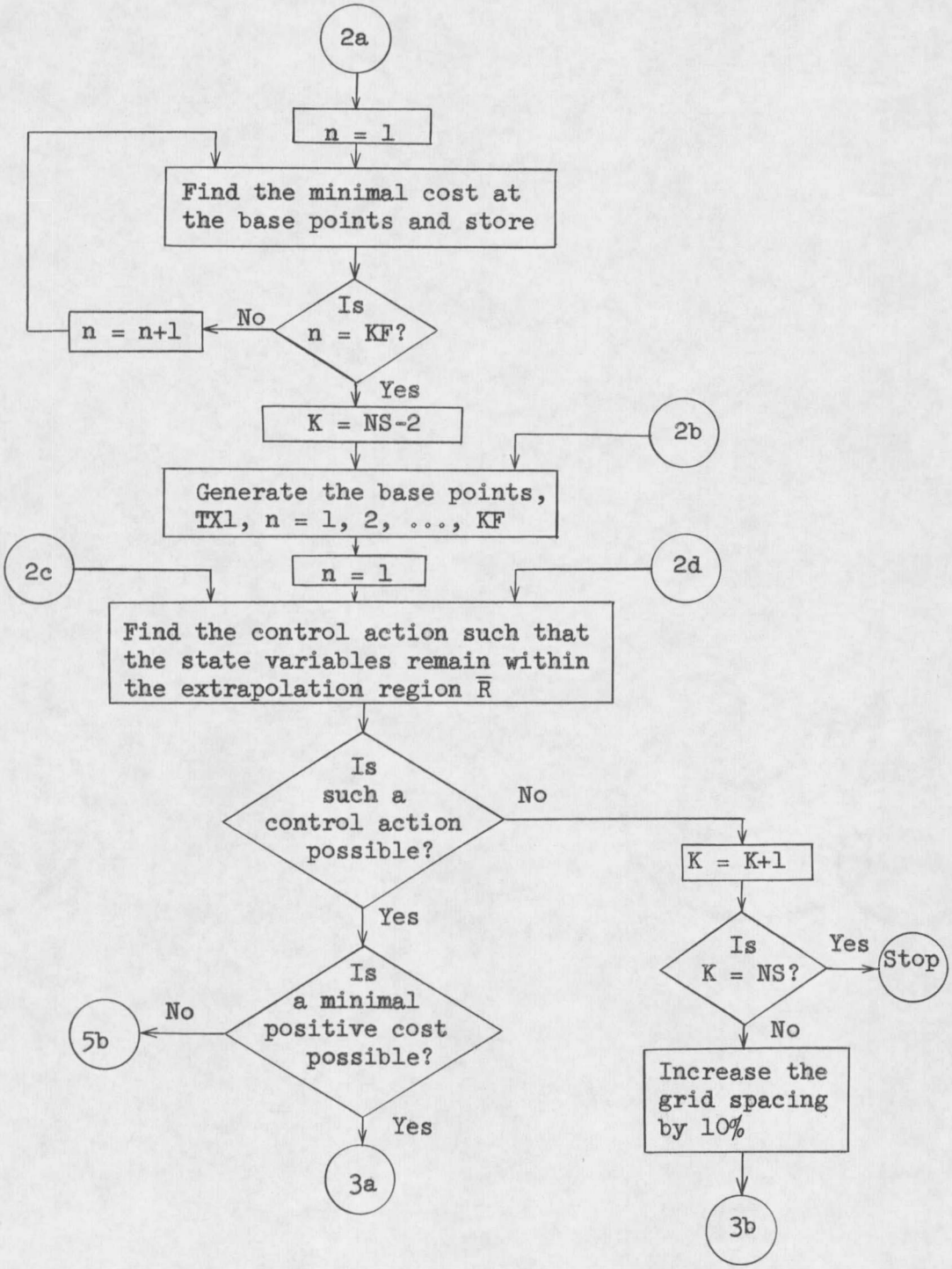

Figure 3.1. (continued)

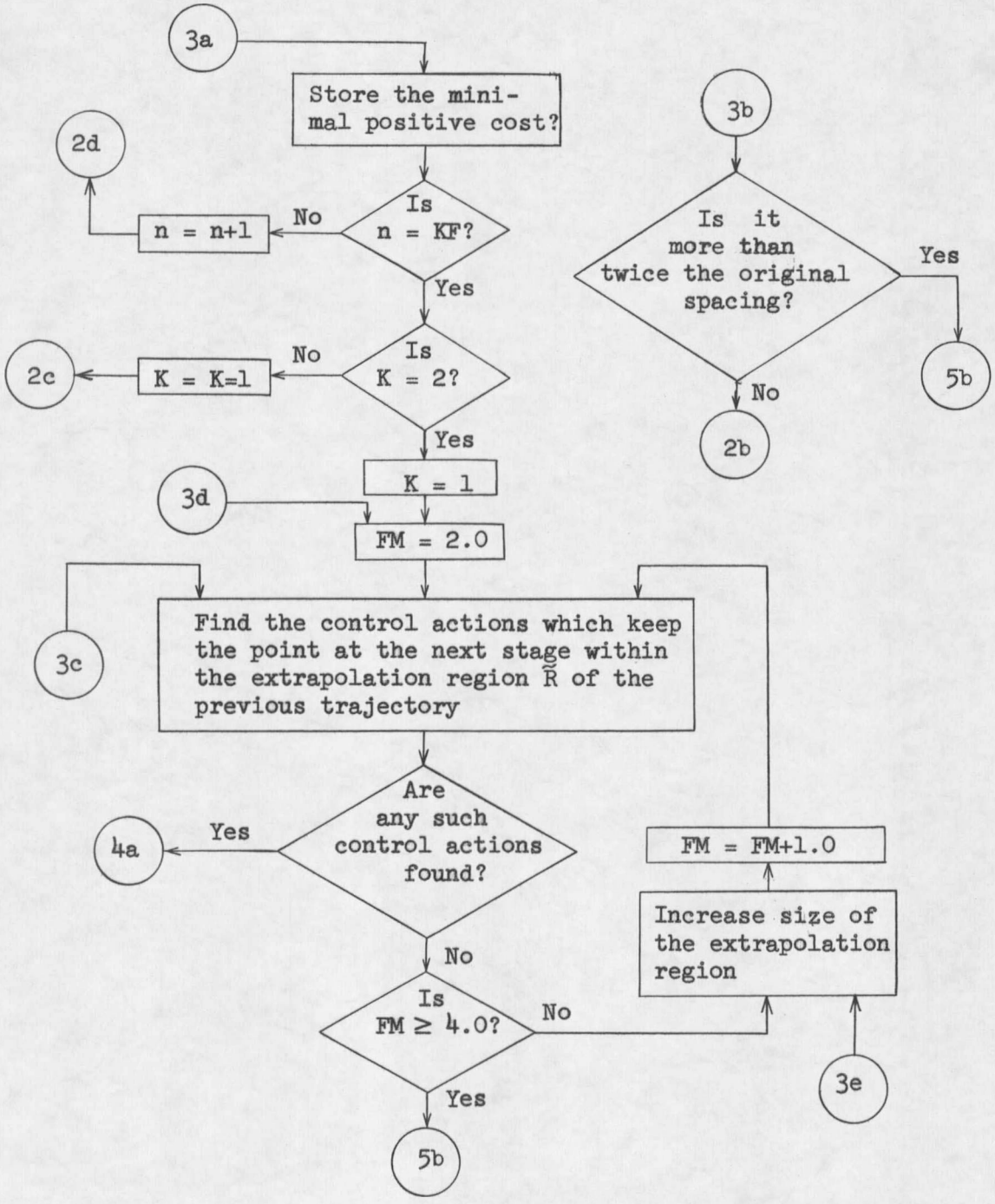

Figure 3.1. (continued)

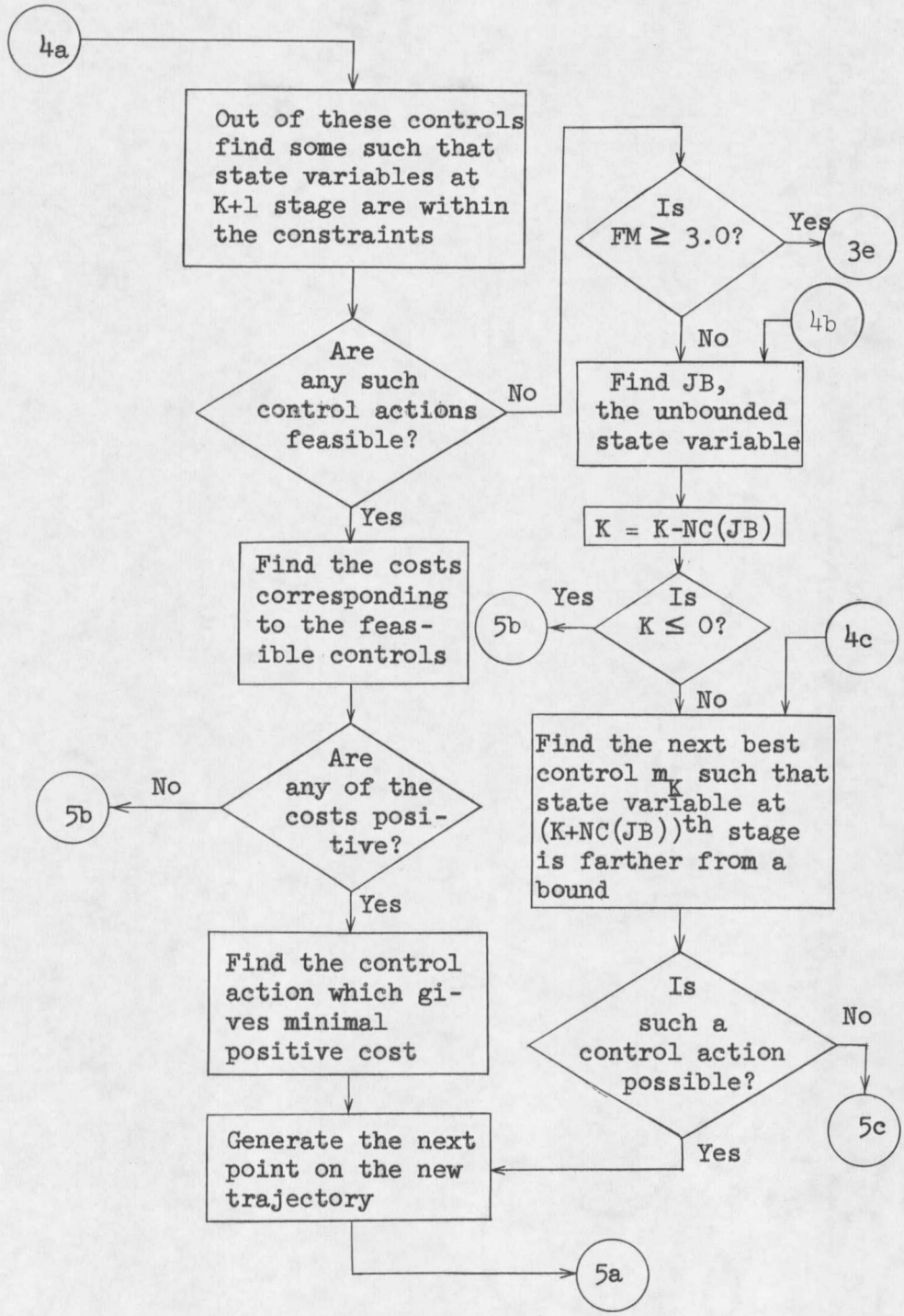

Figure 3.1 (continued)

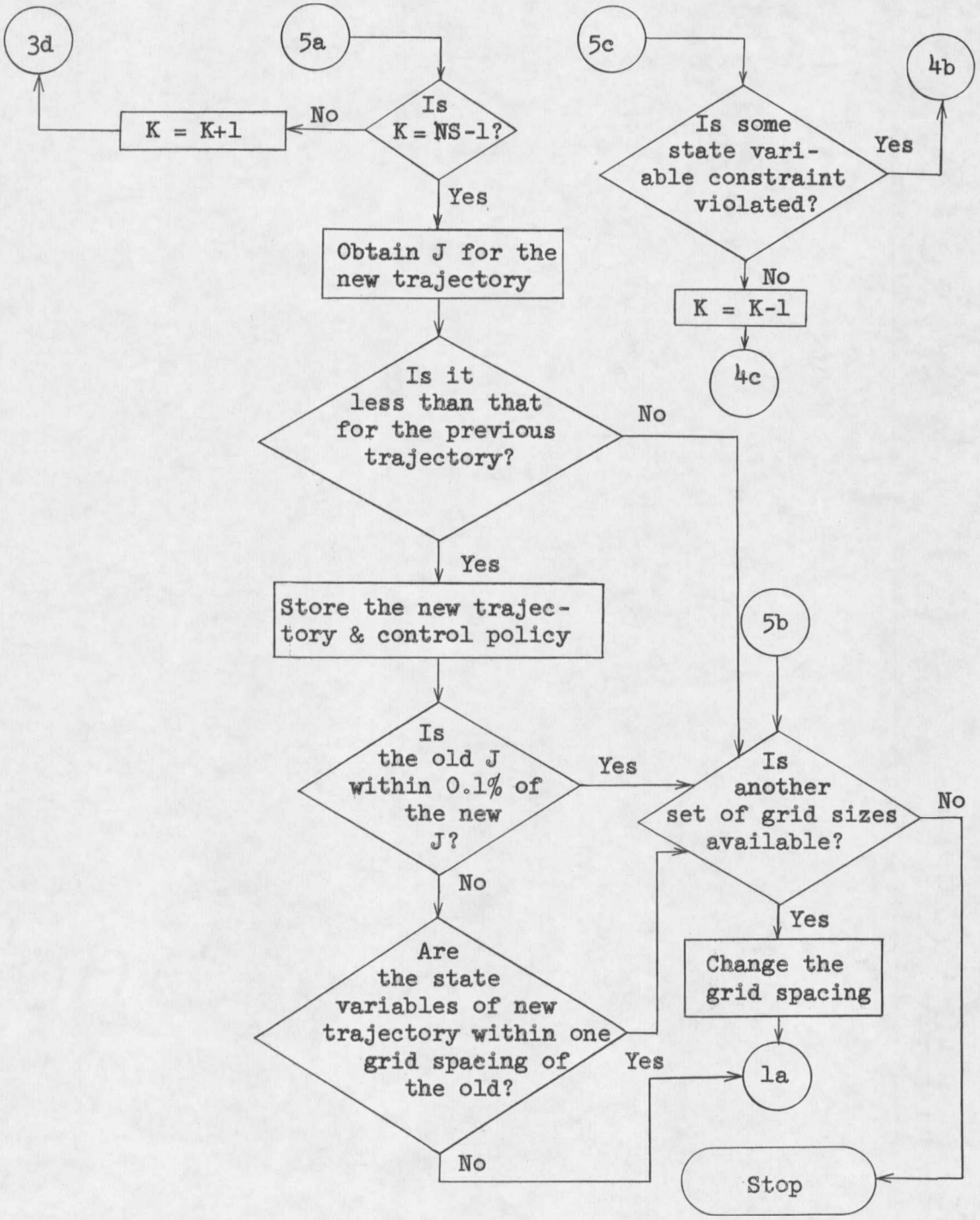

Figure 3.1 (continued)

Another hazard, not mentioned before, may be encountered when the control trajectory is modified so as to drive a state variable away from one of its bounds. The required change in the control action is calculated by assuming that the state variables vary monotonically with the variation in the control action. However, if two of the bounded state variables are affected by the control action in a complementary fashion, and at a given stage both the variables cross the bounds simultaneously, the computed control may chatter back and forth. A provision is made to detect this condition, and to stop further computations or to reduce the grid spacing and start with the optimization procedure again.

In the strategy phase of this particular program, the dimensions of the regions R_k are not changed by a uniform factor. Instead, a provision is made for reading in new dimensions; thus, non-uniform reduction is easily possible along different coordinate axes. During the initialization procedure only the number of sets of grid spacing is specified.

3.2.1 Subroutine GRID

This subroutine generates the coordinates of the base points. A flowchart for the algorithm is given in Figure 3.2. The normalized coordinates of the base points are patterned after the description in Appendix A. In the present routine the coordinates of all of the

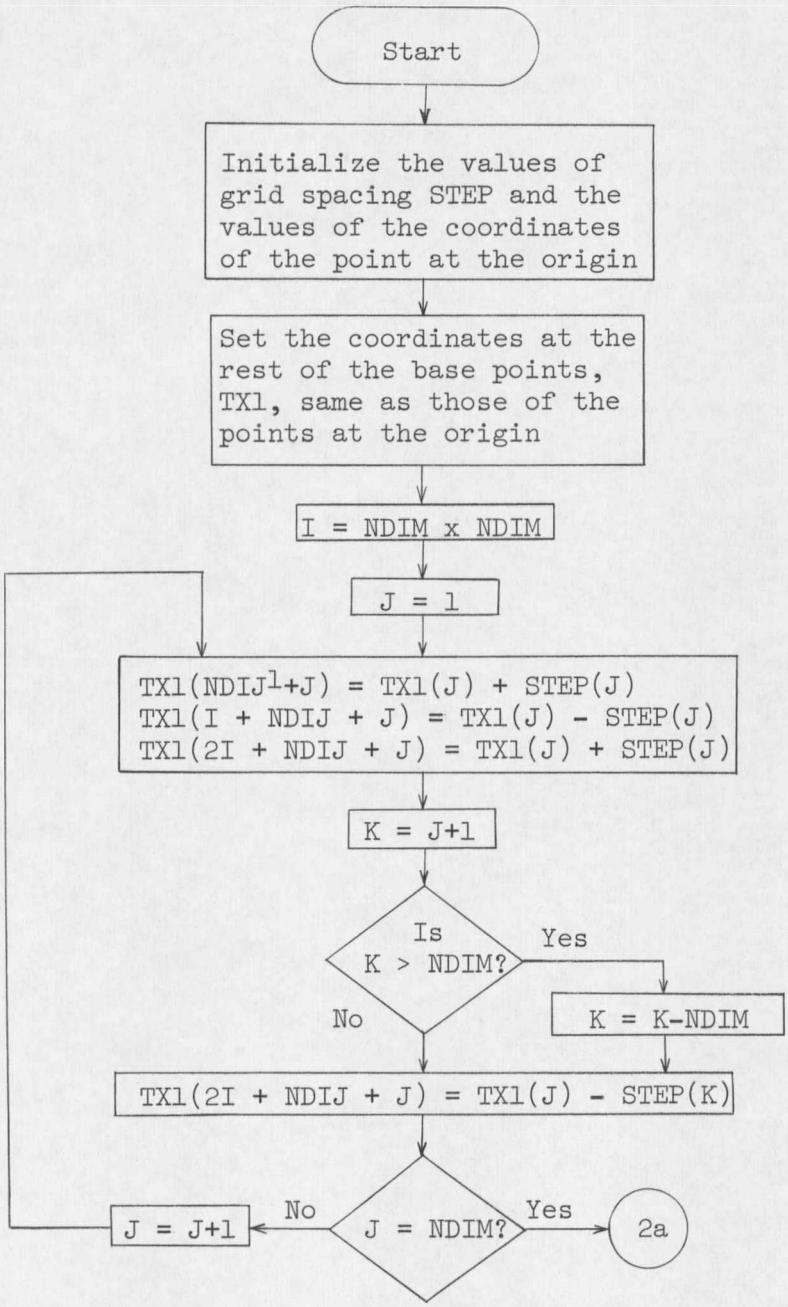

Figure 3.2. Flowchart for subroutine GRID.

[1] NDIJ = NDIM x J.

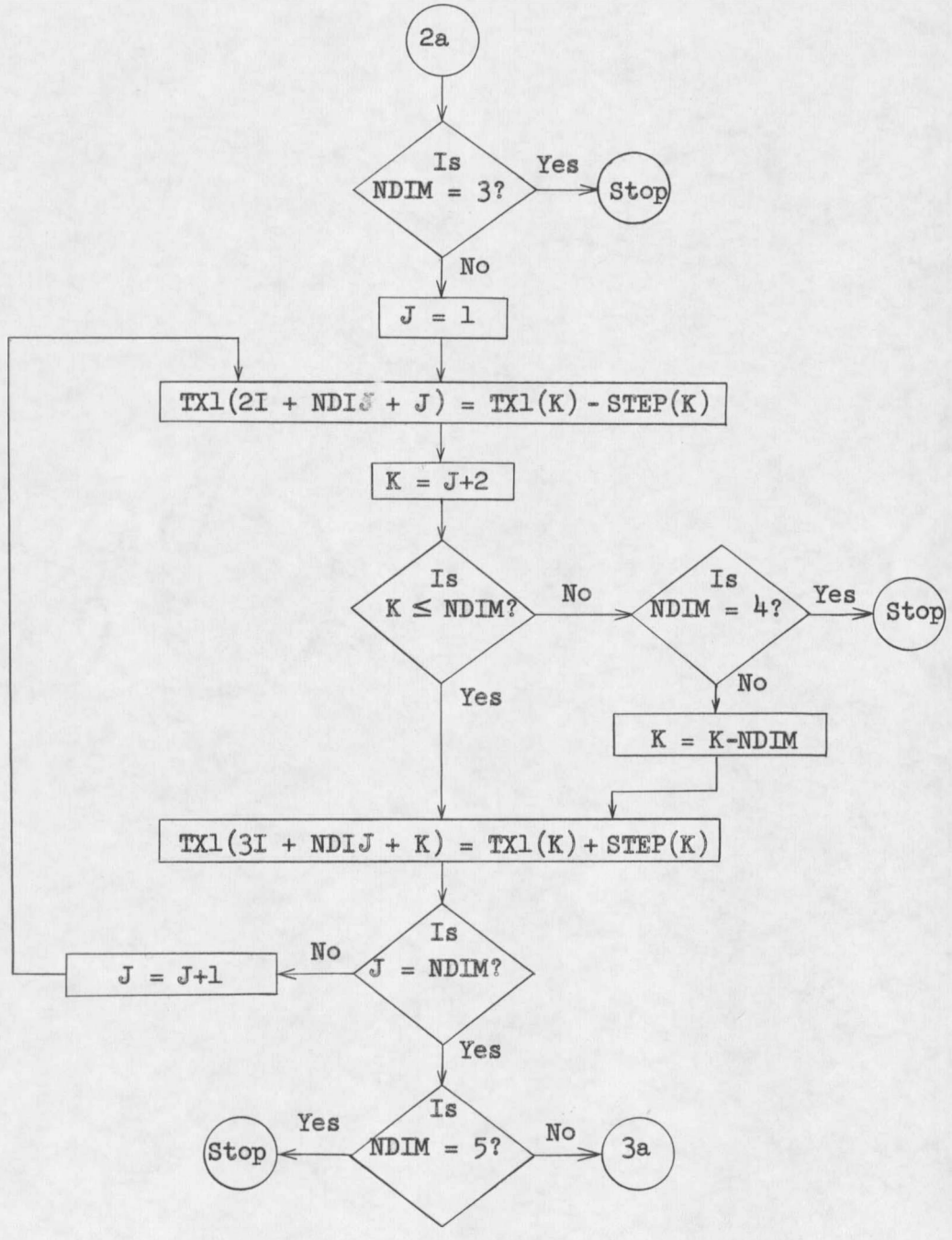

Figure 3.2 (continued)

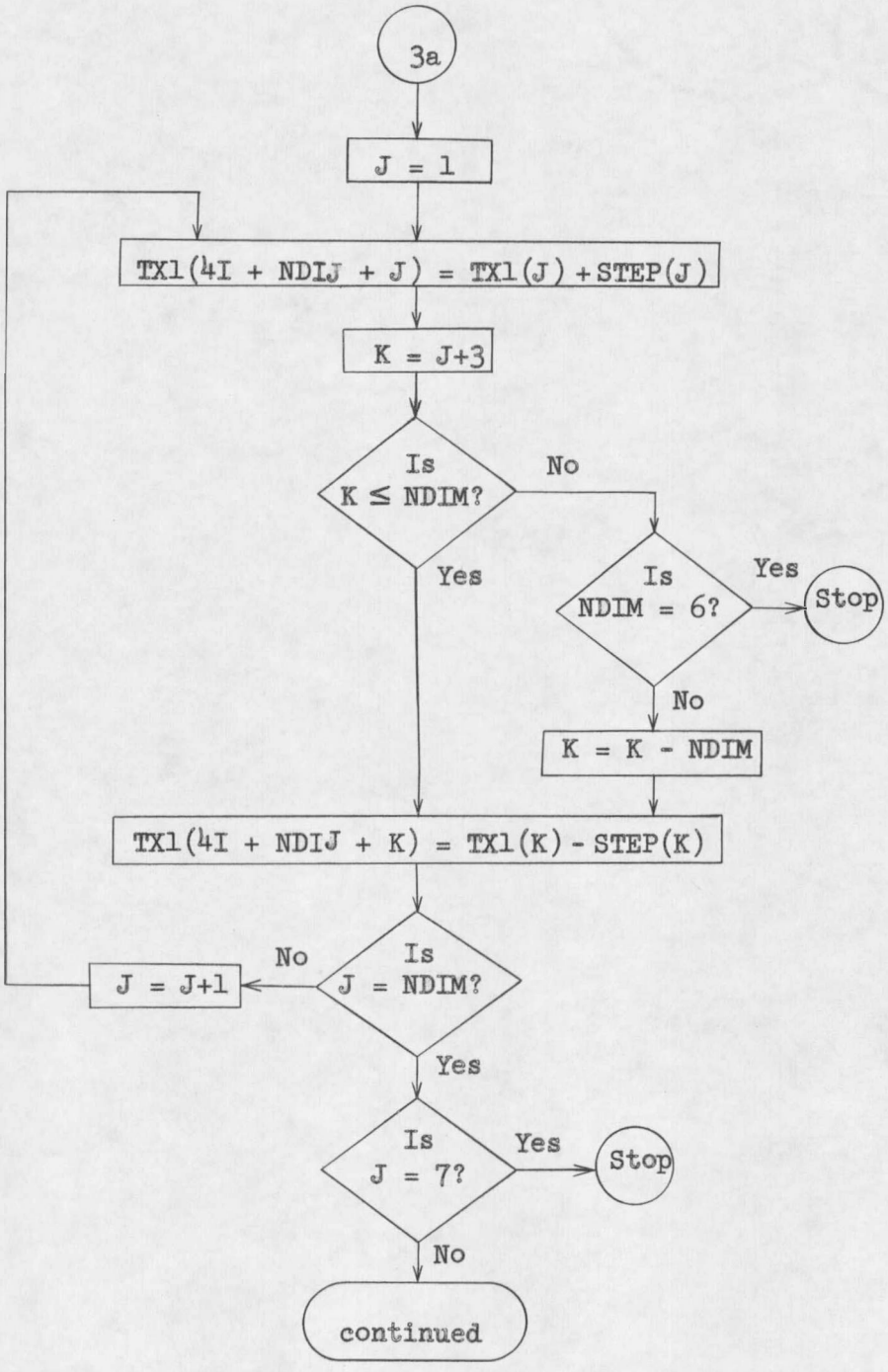

Figure 3.2 (continued)

base points at a stage are initially set equal to the values at the point located on the present trajectory. Thus, only those coordinate values which differ from zero when normalized need to be calculated.

Only actual values of the base point coordinates are obtained in the present routine. In three-variable problems, the normalized base points are as follows: one with all zero coordinates, six with one nonzero normalized coordinate 1 or -1, and three with two consecutive nonzero normalized coordinates 1 and -1, the rest being zero. In n-variable problems, the points described for the three-variable case account for $(3n + 1)$ base points, and that part of the subroutine which is adequate for the three-variable case establishes these points. The rest of the base points number $(n^2 - 3n)/2$; when divided by n, this number yields integer values for odd values of n only. The nonzero normalized coordinates of these points are 1 and -1, and are obtained as described in Appendix A. The number of points which result from each combination of 1, -1, and zeros is equal to n. Furthermore, in successive sets of n such points, the value of x_1 for the first point alternates between 1 and -1 with the number of zeros between 1 and -1 increasing by one each time as in (1, 0, -1, 0, 0) and (-1, 0, 0, 1, 0, 0) for a fifth-order state vector. This is illustrated by including the algorithm for state vectors of fifth- and seventh-order in the flow chart. An extension for higher-order vectors is straightforward.

3.2.2 Cost From the Approximating Polynomial

Two different subroutines, INPOL1 and INPOL2, are used to compute the coefficients of the approximating polynomial and to evaluate the polynomial at a given point. The key to ascertaining the coefficients of the quadratic approximating polynomial, without having to invert a large matrix, is the order in which the monomials of the basis vector and the base points are arranged. This order is explained in Appendix A, and is followed in subroutines GRID, INPOL1, and INPOL2. The flowcharts for the last two of these are given in Figures 3.3 and 3.4, respectively. As in subroutine GRID, a third-order state vector is the minimum order for which these routines are applicable.

The coefficients of the polynomial, that approximates the cost function at a stage, are computed by use of the values of minimal costs at the base points at that stage. Besides the point at the origin, three types of base points correspond to three different expressions for the coefficients in terms of the costs at the base points. The coefficients that correspond to the points with only one nonzero normalized coefficient, 1 or -1, are calculated first. The rest of the coefficients, that correspond to two nonzero normalized coefficients, a 1 and a -1, are computed by using costs at four of the base points, one of which is always the cost at the origin. Each set of n coefficients is calculated in two loops. If there are n_1