STATE INCREMENT DYNAMIC PROGRAMMING AND THE INDUSTRIAL MANAGEMENT SYSTEMS

by

ANSHUMAN KRISHNAKANT DESAI

B.E. (Mech.), Gujarat University
Ahmedabad, India, 1977

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1979

Approved by:

Major Professor

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 CONCEPT OF DECISION MAKING IN MANAGEMENT

Recently there have been a lot of changes in business organizations and they have become enormously complex undertakings. The increasing business strategies and approaches as well as scarce resources have made them even more complex. During the past couple of decades the tendency has been for blending mathematics with business administration resulting in more efficient organization. Similarly the intermediary fields, such as industrial engineering and operations research have been evolved and developed.

Industrial engineering is concerned with the design, improvement and installation of integrated systems of men, materials and equipment. It draws upon specialized knowledge and skill in the mathematical, physical and social sciences, together with the principles and methods of engineering analysis and design, to specify, to predict and to evaluate the results to be obtained from such systems.

With the scarce resources man has recognized their importance and has been constantly trying to emphasize their best use. The resources can be of any kind, such as material, energy, manpower, etc.

In the span of a day the administrator has to make a number of decisions. The selection of one action or sequence of actions from a number of alternative possible actions is known as a decision. Decision making in business, in the past, was practiced more as an art than a science. Increasing costs are associated with decision making. Businesses are larger; each decision involves a larger outlay of time, money and resources.

Some problems in decision making can be quantified. Objective solutions optimizing particular goals can be obtained through the proper application of techniques for the analysis of quantitative data. Some of these techniques are relatively simple and nonmathematical, others involve high levels of mathematical proficiency. With the development of electronic data processing equipment these techniques are brought to the access of many potential users.

The following list is suggested here because it is short and well suited to the use of quantitative information as the basis for decision making (7).

i. Define the problem.

ii. Determine the assumptions and/or limitations which affect the solution.

iii. Identify the possible courses of action.

iv. Isolate the decision making criteria.

v. Determine and compare the possible outcomes and the probability of success in reaching the objective for the various courses of action.

vi. Make the decision (select a course of action).

vii. Implement the decision.

viii. Monitor the results of the decision.

1.2 STATE INCREMENT DYNAMIC PROGRAMMING

It was realized during the period following World War II that there were a large number of activities which could be classified as multistage decision processes (5). It was also found that there was a shortage in available techniques to solve these problems and those

which existed were not versatile enough. Recognition of these facts led to the evolution of many new techniques and one of them was dynamic programming (3). This was a new approach based on the use of the functional equation and the principle of optimality. The principle of optimality can be stated as,

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

Initially the emphasis was on those processes which were specifically posed as multistage decision processes. Then all the processes to which dynamic programming could be applied were considered. In 1955 R. Bellman and others (7) began a systematic study of the computational feasibility of dynamic programming. They collected a number of optimization problems from many different fields and applied their methods in many different ways.

Despite the attractive features of the standard algorithm (dynamic programming), its applicability thus far has been limited to relatively simple cases. This is due to the large computational requirements of this algorithm. The most severe restriction is generally the high amount of fast storage memory required to implement the basic calculations. Another difficulty is the amount of computing time required to obtain the complete solution. Thus, while dynamic programming is frequently used as an analytical and conceptual tool, the computational difficulties associated with the standard algorithm have severely limited its application to large scale optimization problems.

Along with other methods to overcome the difficulty of fast storage memory requirement in dynamic programming, state increment dynamic

programming (16,17,18) is a good substitute. R. E. Larson was the first to show the application of this new method to the problems which could be solved by dynamic programming but are likely to face the 'dimensionality difficulty.' This method also is based on Bellman's 'principle of optimality.' This procedure retains the desirable properties of the standard algorithm but has a reduced computational requirements. This procedure always reduces high-speed storage memory requirement, often by orders of magnitude. In a number of cases a substantial reduction in computing time can be achieved as well. Thus state increment dynamic programming represents a significant step in increasing the range of optimization problems that can be solved with state-of-the-art computer facilities.

1.3 PURPOSE OF THIS STUDY

Industrial engineers deal with many different varieties of engineering as well as business problems related to engineering. Decision making problems are one of the varieties which an industrial engineer should know 'how to tackle.' There are many ways or techniques to solve these kinds of problems but an engineer always looks for more efficient ways.

The purpose of this study is to evaluate the effectiveness of state increment dynamic programming in solving industrial management problems which involve nonlinear difference equations.

Other computational procedures, such as gradient technique, second variation method, and invariant imbedding, etc., could also be used for solving the problems of this nature, but they will not be discussed here since they are outside the scope of this study.

1.4 HYDRO-DYNAMIC JOURNAL BEARING DESIGN

With the invention of data processing equipment, many fields of science have changed their manner of problem solution. At the same time mechanical engineers concerned with designs have also changed their approach to the design problems.

Mechanical designs used to be based on the standard design equations and were designed on the idea of 'safe' design. Now, optimization techniques have improved their approach and helped them to carry out 'safe' and 'economical' designs.

Chapter 2 of this study is concerned with a basic mechanical design, hydrodynamic journal bearings. Master's research work was started with this work, and because of insufficient information we were not able to extend the problem and the work was discontinued.

CHAPTER 2

OPTIMIZATION OF HYDRODYNAMIC JOURNAL BEARING DESIGN

2.1 GENERAL

In these days of automization, automated mechanical designs have become a fertile field for mechanical engineers. For many years they have been designing very 'safe' mechanical systems and have been using design equations in standard forms to arrive at such 'safe' designs. Frequently these designs, although reliable, are costly to produce. These higher costs of 'safe' designs are mainly due to high material costs, high labor costs and high processing costs. If suitable optimization procedures be used to evaluate such designs, it may result in cost savings. Since an optimization procedure can evaluate many criteria, the saving would not be at the risk of the quality.

As indicated in the previous chapter the master's research work was started with a goal of optimizing some kind of mechanical design using an appropriate optimization technique. The literature survey included a large number of mechanical designs such as cold rolling process, gear trains, hydrodynamic journal bearings, other varieties of bearings, etc.

The design of a hydrodynamic journal bearing is of key importance since it is a basic part in rotating machinery. In the last decade many researchers have tried to optimize different journal bearings. Each design is different because it depends on many factors such as operating conditions and behavior of the parent system of which it is a part, etc.

Mechanical design is a multi-phase process requiring constant decision making on the part of the designer. As engineering design

has matured so have the guidelines and methods that the designer has at his disposal to help him in his choice. Drawing from his experience the engineer is able to define variables, a design objective, and a set of constraints that must be met in order that the design be a workable solution. Thus by developing corresponding equations a design problem can be stated in a suitable form of mathematical programming.

The basis of hydrodynamic journal bearing design is the solution of Reynold's equation. Raimondi and Boyd (25) of the Westinghouse Research Laboratories gave the solution in terms of performance characteristic curves. The series of three papers gave performance curves for different assumptions made in each of them. All of them discuss the journal bearing with the load at the center. The first paper discusses the journal bearing with length-to-diameter ratio of one, constant oil viscosity and no film rupture. The second paper in the series discusses the centrally loaded bearing with length-to-diameter ratio of 1/2 and 1/4. A problem illustrating the L/D ratio on journal misalignment shows that the shortest bearing is not necessarily the one capable of tolerating the maximum misalignment. The third paper discusses the bearing which accounts for the film rupture and performance curves for different L/D ratios.

Seireg and Ezzat (26) presented an automated system for the selection of the length, clearance and lubricant viscosity which optimize the performance of hydrodynamic journal bearings under specified values or range of loads and speeds. The authors derived equations from the curves given by Raimondi and Boyd (25). A curve fitting technique was used to arrive at the functional equations. This paper discussed

optimization of bearings based on competitive objectives of reducing the bearing temperature and at the same time reducing the oil flow. The optimization technique used here was gradient search.

2.2 PROBLEM FORMULATION

The usual design procedure requires mathematical formulation of the problem under consideration as an optimal programming problem. The policy can be as follows;

i. Defining the system variables and the decision parameters.

ii. Stating equality or inequality constraints imposed on the design.

iii. Defining the expressions relating different parameters governing the system behavior.

iv. Developing the search technique best suited for the problem under consideration.

2.2.1 System Parameters

The main independent parameters for the problem under consideration are

(D, L, C), (μ), (W, N), S

where

D = journal diameter, inches

L = bearing length, inches

C = radial clearance, inches

μ = lubricant average viscosity, Reyn

W = bearing load

N = journal rotational speed, rps

S = Sommerfeld number (defined on page 11)

These parameters, as grouped, describe the bearing geometry, oil characteristics, and load specifications, respectively. In journal bearing design the parameters, D, N and W are assumed to be known. Hence, the design parameters left are, L/D, C, μ. Figure 2.1 shows the relationships.

Now the constraints on the design parameters are defined as follows;

$$h_0 \geq h_{min}$$

$$t_{max} \leq t_{maxn}$$

$$p_{max} \leq p_{maxn}$$

$$\mu \geq \mu_{min}$$

$$L_{min} \leq L \leq L_{max}.$$

where, h_0 = minimum oil film thickness, inch

t_{max} = maximum oil film temperature, °F

p_{max} = maximum oil film pressure, psi.

These constraints are dictated by the quality of machining, the characteristics of the material-lubricant pair, and available space.

2.2.2 Governing Equations

The governing equations are derived from the curves given by Raimondi and Boyd (25) in their papers. A curve fitting technique is used to derive the equations (26). They are used for the calculations of the temperature rise, maximum oil film pressure, oil flow, frictional loss, etc. For example

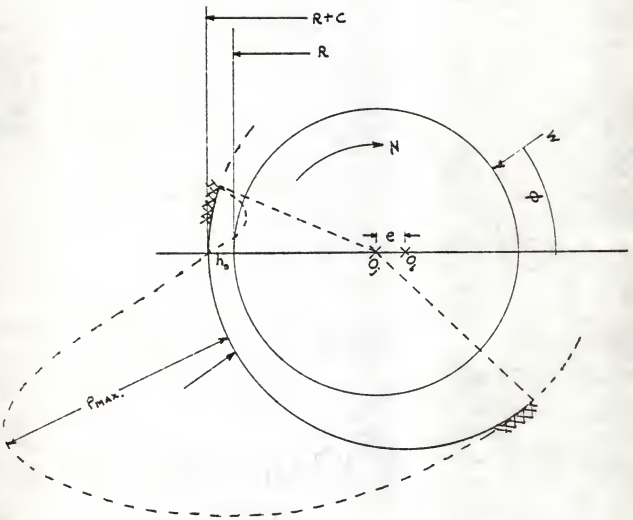

Fig. 2.1 Bearing Geometry.

for $S \leq 0.5$

and $0.25 \leq L/D \leq 0.5$

we have

$$h_0 = 1.585 \cdot C \cdot (L/D)^{0.913} (S)^{0.655(L/D)^{0.922}}$$

$$\Delta t = \frac{0.5}{(L/D)^{0.374}} \cdot P(S)^{0.695/(L/D)^{0.139}}$$

$$\frac{P}{P_{max}} = 0.76 (L/D)^{0.62} (S)^{0.24}$$

$$\frac{RNCL}{Q} = 0.128 (L/D)^{0.048} (S)^{0.1} (L/D)^{0.47}$$

$$f(R/C) = \frac{12.6}{(L/D)^{0.41}} (S)^{0.62/(L/D)^{0.1035}}$$

Similarly, other equations for different ranges of (L/D) and S are the same in structure but have different values of the constants. Here S is known as Sommerfeld number and is defined as

$$S = (R/C)^2 \frac{\mu N}{P}, \text{ where } R = \frac{D}{2}.$$

The rest of the mathematical relationships between the design variables for different values of constants are given in the Appendix A.

2.2.3 Design Criterion

The designer should develop a design criterion which accurately describes the designer's objectives. In the bearing design problem many criteria can be envisioned. Some of them can be stated as follows. Minimizing the maximum temperature rise in the bearing, minimizing the oil flow required for accurate lubrication, minimizing the frictional loss, etc. The objective can be composed of the multitude of the above mentioned factors.

2.3 COMPETING OBJECTIVE FUNCTIONS IN OPTIMIZATION

A competing objective function is the one in which the components are dependent on each other and by decreasing one we have to increase the other and the overall objective may be to optimize the sum of the both at a time. For example, in the bearing design problem we want to decrease the rise in bearing temperature along with the decrease in oil flow in the bearing. Now for minimizing bearing temperature rise we have to increase oil flow, but our objective also consists of minimizing the oil flow in the bearing. Hence we have to compromise at some stage by weighing the relative importance. There are various methods to accomplish this.

One approach to multiple objective problem does not include any formal optimization. It requires that the designer have sound knowledge about the problem. In this the designer selects a candidate design and uses a computer to do the analysis required to determine the behavior of the design. Then interactively the design is altered by the designer until a 'fair' compromise design is obtained. This selected design is not necessarily optimum, and the designer has a little insight in the sacrifice of one objective for the improvement of the other.

Another approach uses an optimization algorithm to find the 'best' design variable changes to determine a new estimate for the optimum design. This interactive procedure continues until a 'good' design is found. In this approach the designer participates directly in optimization process and as a result he obtains further understanding of the problem. The optimization procedure directs the designer continually towards optimum design. Sometimes, the designer's bias may prevent considerations of all possible compromises between the objectives.

Another approach is simply to consider all but one multiple objectives as constraints and one as objective. Thus it reduces the multiple objective problem to the single objective one. This approach requires difficult decisions regarding the selection of constraints and constraint values.

One more approach to multiple objective problem involves the selection of an optimum linear combination F of the objectives. The function F can be defined as

$$F = f_1(u) + w_2 f_2(u) + \ldots + w_n f_n(u).$$

The selection of the best weighing factor w_i depends on many different factors and also on the particular type of problem. This particular type of approach has the advantage of considering only optimum designs but it may be difficult and/or expensive to determine the best weighing factors.

2.4 SEARCH METHOD

The choice of search method adopted for the automated design should be very careful. It should suit the design domain and the criterion under consideration. Due to the complex structure of the design domain in the case of bearing design, the search method should also allow for starting points which may violate the constraints. Seireg and Ezzat (26) and Bartel and Marks (2) have used the gradient search technique for the solution of the problem. But Eason and Fenton (10) suggested that pattern search and simplex search are better than the gradient techniques using secant derivate approximation. Hence, we chose simplex pattern search (11,23,28) for the solution of the optimum bearing design.

After repeating the problem stated before we wanted to extend the static problem to dynamic one. But it was found difficult to do that. The reason was the unavailability of a complete analytical solution to the Reynold's equation. The analytical form was needed for the extension of the problem. Even very early papers from the thirties and forties had only numerical solutions to the Reynold's equation, either in the form of tables or curves. Seireg and Ezzat (26) arrived at algebraic relationships which were found by curve fitting techniques, but they also have constants in the form of absolute values and hence it would not be advisable to use these relationships without completely knowing them.

CHAPTER 3

STATE INCREMENT DYNAMIC PROGRAMMING

3.1 GENERAL

One of the most important optimization techniques developed in last three decades is Bellman's dynamic programming (3,4,5). It is capable of solving, at least in principle, many important and difficult optimization problems irrespective of their nature, linear or non-linear. However, because of the extremely large high speed memory requirement, only relatively simple problems have been solved on existing computers. It has been found by experience that the method works satisfactorily until the problem has three state variables. For more than three state variables it is not a very accurate method and sometimes memory overflow occurs. Bellman calls this difficulty 'the curse of dimensionality.'

After realizing the dimensionality difficulty in dynamic programming many other numerical methods have been developed. Some of them can be listed as (21),

i. polynomial approximation
ii. lagrange multiplier
iii. state increment dynamic programming
iv. differential dynamic programming
v. quasilinearization (19), etc.

The first two of them trade off computer time for the high speed memory requirement. Quasilinearization linearizes the function iteratively using the Newton-Raphson method. The advantage of this method is that one gets rid of the control variables; hence is easy to solve. Also, the method has quadratic convergence whenever the problem converges.

State increment dynamic programming essentially reduces the high speed memory requirement in the problem. The way in which this is achieved will be explained in this chapter. Many times it takes more time than dynamic programming but in some cases reduction in computation time has been also observed. In a particular case reduction from 10^6 storage locations to 100 storage locations has been obtained (15).

R. E. Larson (15,16,17,18) is the inventor of state increment dynamic programming. While working at Stanford Research Institute in mid-sixties he developed this method and published many papers and a book showing method and its applications.

State increment dynamic programming is based on the 'principle of optimality' given by Bellman. As previously stated, "an optimal policy has the property that, whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with respect to the state resulting from the first decision."

The difference between the conventional dynamic programming developed by Bellman (3) and the state increment dynamic programming by Larson (15) is in the choice of the time interval over which a given control is applied. Conventional dynamic programming uses a fixed time interval whereas state increment dynamic programming determines the time interval as the minimum time required for at least one of the state variables to change by one increment. As a result of the choice of time interval, the next state after applying the control will lie on the surface of an n-dimensional hypercube centered at the given point and with length $2(\Delta x_i)$ along the i^{th} state variable axis. This property is used to reduce the fast memory requirement. The entire state-time (X-t) space is partitioned into several blocks. Each block covers some increment along each state

and a longer interval along the time axis. Then the optimal value of the variables is calculated for one block at a time, and not for the entire state space as in the standard dynamic programming algorithm.

3.2 PROBLEM FORMULATION

The optimization problem to which the state increment dynamic programming procedure is applied is most conveniently formulated in the continuous case over the interval $t_i \leq t \leq t_f$. Hence, the system equation becomes a set of nonlinear time varying differential equations.

$$\dot{\underline{x}} = \underline{f}(\underline{x},\underline{u},t) \tag{3.1}$$

where $\underline{x}$ = state vector in n dimensions

$\underline{u}$ = control vector in q dimensions

$\underline{t}$ = stage variable, usually time

$\underline{f}$ = n dimensional vector functional.

The performance criterion to be minimized is a cost function denoted by J. It consists of the sum of an integral with respect to a scalar function of state variables, control variables, and stage variables and a scalar function depending on the final state and the final stage. Thus

$$J = \int_{t_i}^{t_f} \ell[\underline{x}(\sigma),\ \underline{u}(\sigma),\sigma)]d\sigma + \psi[x(t_f),\ t_f] \tag{3.2}$$

where t_i = initial stage (time)

t_f = final stage (time)

σ = dummy variable for stage

J = cost function

ℓ = loss function; cost function per unit time

ψ = final value term in cost function.

Constraints are on both state and control variables of the form

$$\underline{x} \in X(t)$$
$$\underline{u} \in U(t) \tag{3.3}$$

where

$X(t)$ = set of allowable states of the time t

$U(t)$ = set of the admissible controls at state $\underline{x}$ and time t.

Assumption: it is assumed that $\underline{u}(t)$, $t_i \leq t \leq t_f$, is piecewise constant over intervals of length δt. In order to implement the procedure on a digital computer the set of differential equations is approximated by a set of difference equations.

Thus, $$\underline{x}(t+\Delta t) = \underline{x}(t) + \underline{f}[\underline{x}(t), \underline{u}(t), t]\delta t \tag{3.4}$$

and the change in performance criterion over the interval from t to $(t + \delta t)$ is approximated by

$$\int_t^{t+\Delta t} \ell[\underline{x}(\sigma), \underline{u}(\sigma), \sigma]\, d\sigma$$
$$= \ell[\underline{x}(t), \underline{u}(t), t]\delta t \tag{3.5}$$

Again, if δt were fixed to a value Δt then the computation could be done by the conventional dynamic programming method, but it is not the case here. δt is determined by computations and is the basic element of state increment dynamic programming.

The functional equation can be derived from the cost function based on Bellman's principle of optimality and can be stated as

$$I(\underline{x},t) = \min_{\underline{u}\in U} \{\ell[\underline{x},\underline{u},t]\delta t + I[\underline{x} + \underline{f}(\underline{x},\underline{u},t)\ t, \delta t + \delta t]\} \tag{3.6}$$

3.3 CONSTRAINTS AND QUANTIZATION

The constraints are restricted to a set of admissible states X and a set of admissible controls U. For example, inequality constraints of the form $\phi(\underline{x},t) \leq 0$ can be used to bound the state variables

$$\beta_i^- \leq x_i \leq \beta_i^+ , \qquad \text{for } i = 1,2, \ldots, n. \tag{3.7}$$

Inequality constraints of the type, $\phi(\underline{x}, \underline{u}, t) \leq 0$, can be used to restrict the control variables

$$\alpha_j^- \leq u_j \leq \alpha_j^+ , \qquad \text{for } j = 1, 2, \ldots, q. \tag{3.8}$$

The quantities β_i^- and β_i^+ can vary with t, while the quantities α_j and α_j^+ can vary with $\underline{x}$ and t.

Within the allowable range, each state variable x_i is quantized into a finite number of values, N_i. It is convenient to assume the quantization in constant increments, Δx_i. The result is

$$x_i = \beta_i^- + j_i \, \Delta x_i \tag{3.9}$$

where

$$j_i = 1, 2, \ldots, N_i$$

and

$$N_i \, \Delta x_i = \beta_i^+ - \beta_i^- \qquad \text{for } i = 1,2, \ldots, n.$$

The set of state vectors for which each component has the form of Eq. 3.9 is called the set of quantized admissible states, X.

Although the control variables can be quantized in a similar manner, it is necessary only that there be a finite number of admissible controls. The set of admissible controls, U, can be given as

$$U = \{\underline{u}^{(1)}, \underline{u}^{(2)}, \ldots, \underline{u}^{(m)}\} \tag{3.10}$$

The choice of $u^{(m)} \in U$ depends on the problem under consideration.

3.4 DETERMINATION OF THE TIME INTERVAL δt

In state increment dynamic programming the time control for computation of the optimal control (Δt) may or may not be fixed depending on the nature of the problem. But the interval δt varies with the control applied. The interval δt is determined as the minimum time interval required for any one of the n state variables to change by one increment. For example, if Δx_i is the increment in the i^{th} state variable and if control $\underline{u}$ is applied, then

$$\delta t = \min_{i=1,2,\ldots,n} \left\{ \frac{\Delta x_i}{\left| f_i(\underline{x},\underline{u},t) \right|} \right\} \tag{3.11}$$

where $f_i(\underline{x},\underline{u},t)$ is the i^{th} component of $\underline{f}(\underline{x},\underline{u},t)$, the system differential equation vector.

Expression of δt as in Equation (3.11) is the basic equation in state increment dynamic programming. This also shows that the next state lies only Δx_i from the original one. Also, for the iteration of the minimum cost function, only the value at these quantized states at Δx_i from original state need be stored. And this is the property which reduces the fast memory requirement by a considerable amount.

3.5 BLOCK CONCEPT

The significant amount of reduction in high speed memory requirement can be achieved by processing the data so as to obtain the maximum utilization of the reduction for a single calculation. This is done by computations in units called blocks.

Blocks are defined by partitioning the n+1 dimensional space into rectangular subunits. Each block covers an increment w_i along the x_i - axis and ΔT along the time-axis. A particular block can be denoted by the largest value of the coordinates that are contained within the block. Figure 3.1 shows the block for a problem with two state variables. For an n state variable problem the block can be denoted as

$$B(j_0, j_1, j_2, \cdots, j_n).$$

This block contains the values of t and x such that

$$(j_0-1)\ \Delta T \leq t - t_i \leq j_0\ \Delta T$$

and

$$(j_i-1)\ w_i\ \Delta x_i \leq x_i - \beta_i^- \leq j_2\ w_i\ \Delta x_i \qquad (3.12)$$

where

$$j_0 = 1, 2, \ldots, J_0$$

$$J_0 = \Delta T = t_f - t_i$$

$$j_i = 1, 2, \ldots, J_i$$

$$J_i\ w_i\ \Delta x_i = \beta_i^+ - \beta_i^-$$

$$i = 1, 2, \ldots, n.$$

For a two dimensional (n=2) problem each block is a three dimensional rectangular solid, as shown in Figure 3.1, which has two axes for the stage variable.

3.6 COMPUTATIONS WITHIN THE BLOCK

The computational procedure assumes that the next state is within the block under process. In the general case the time interval is

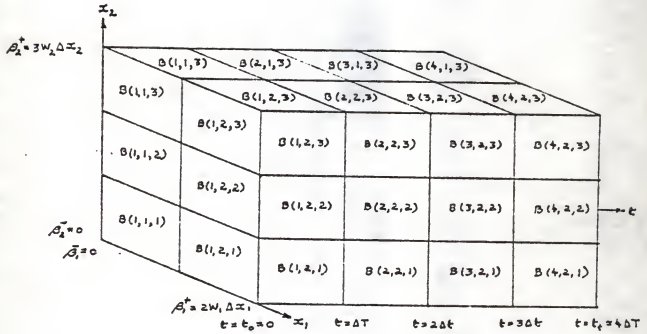

Fig. 3.1 Block Diagram For Two State Variables Case.

divided into smaller increments Δt. The set of quantized times are determined as

$$t = t_0 + (j_0-1)\ \Delta t + S\Delta t \tag{3.13}$$

where

$$s = 0,\ 1,\ 2,\ \ldots,\ S$$

and

$$S\ \Delta t = \Delta T$$

Typical values of S range between 5 and 15. Then the optimal control is computed at each quantized state $x \in X$ for each quantized stage t as given in Equation (3.13). Then each admissible control $U^{(k)} \in U$ is applied. Over each control the time over which it is applied is determined as in Eq. (3.11) as

$$\delta t^{(k)} = \min_{i=1,2,\ldots,n} \{ \left| \frac{\Delta x_i}{f_i(\underline{x}, \underline{u}^{(k)}, t)} \right| \} . \tag{3.14}$$

With this state and time the optimal cost function

$$I(\underline{x}^{(k)},\ t + \delta t^{(k)})$$

is computed by interpolation in (n-1) state variables and time using previously calculated values at quantized state and times $t+\Delta t$, $t+2\Delta t$, ... If the control is such that none of the state variables change, i.e., if $\underline{f}(\underline{x},\ \underline{u}^{(k)},\ t) = 0$, then $\underline{x}^{(k)} = x$ and $\delta t^{(k)}$ is set equal to Δt. The resulting next states for a one-dimensional example are shown in Figure 3.2, where

$$U = \{u^{(1)},\ u^{(2)},\ u^{(3)},\ u^{(4)},\ u^{(5)}\}$$

values of optimal cost function are known at the points indicated by small circles.

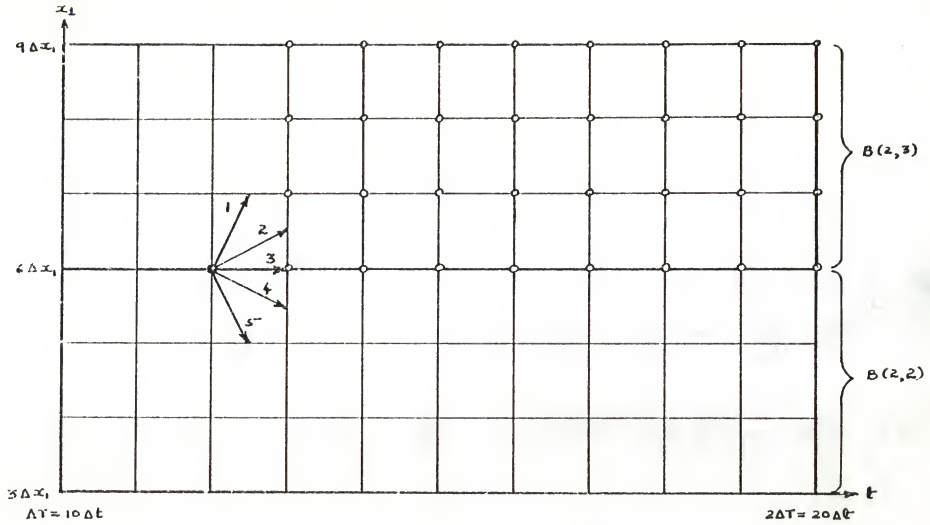

Fig. 3.2 Transition To A Block Not Yet Computed.

Once the optimal cost is found out for each of the next states, the cost of each control over time $\delta t^{(k)}$ is found by evaluating the cost function per unit time

$$\ell(\underline{x}, \underline{u}^{(k)}, t).$$

The optimal cost and optimal control at $\underline{x},t$ are then calculated by using the principle of optimality. The result is

$$I(\underline{x},t) = \min_{k=1,2,\ldots,k} \{\ell(\underline{x},\underline{u}^{(k)},t)\delta t^{(k)} + \underline{T}(\underline{x}^{(k)}, t + \delta t^{(k)}\}, \quad (3.16)$$

$$\hat{k} = \arg \min_{k=1,2,\ldots,k} \{\ell(\underline{x},\underline{u}^{(k)},t)\delta t^{(k)} + \underline{T}(\underline{x}^{(k)},t + \delta t^{(k)}\}, \quad (3.17)$$

$$\hat{u}(\underline{x},t) = \underline{u}^{(\hat{k})}.$$

3.7 INTERBLOCK TRANSITION

This can be divided into two parts. One is the transition into previously processed blocks, and the other transition is into previously not processed blocks.

The simplest case of the previously computed block transition is the one in which the boundary is common between the two blocks. In this case there is no need to find new optimal points on the boundary. However minimum costs at these states are stored in the high speed memory and then they are used in interpolation formulae for minimum costs at next state. The storage of these values allows transition from the block currently being computed to the previously computed block. As long as the values of minimum costs on such a boundary are available, transitions of this type to previously computed blocks can be made without constraint.

In a system to which state increment dynamic programming is applied there is generally some prior knowledge about the behavior of optimal trajectories. The knowledge of direction of the optimal trejectory is called the perferred direction of motion. Then according to the preferred direction the order of processing of the blocks is determined. The processing order is inverse of the preferred direction to achieve the transition to previously computed block. This helps in saving a significant amount of computational time.

The second of the interblock transitions is the transition to blocks not previously computed. The simplest technique is to exclude all controls which result in such a transition during the computation of a block, but to consider such a transition after both the blocks have been computed. Therefore it will be clear that in the case of computation of boundary, the controls which take the next state in the block that is not yet computed are not allowed. However, when the later block has been computed such controls are applied at the boundary at the least value of t within the block. Then the minimum costs are found for all points, the points on the boundary as well as the points within the later block. If one of these costs is less than the existing cost then it replaces that cost and becomes the new minimum cost, and the corresponding control becomes optimal control.

A more accurate procedure for allowing these transitions is to extrapolate the minimum cost function into the not yet computed block. In general, extrapolation procedures are less accurate than interpolation procedures. However, the extension of state by extrapolation is at most Δx_i in the x_i direction and hence the error is strictly bounded.

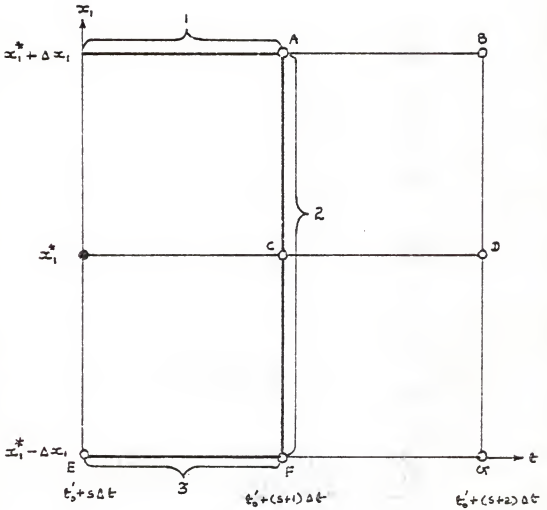

Fig. 3.3 Linear Interpolation In One Dimensional Case.

Moreover at every Δt interval along t, the minimum costs are recomputed along the boundary and that does not allow the error to accumulate due to extrapolation.

If extrapolation does not provide adequate results, then the recomputation of the results along the boundary can be done. In this case the optimal control and the minimum cost along the boundary are recomputed when the block in which the extension was made is processed. In general, the increase in computing time and high speed memory requirement are not worth the slight increase in accuracy.

A better alternative is to precompute some results in order to know the preferred direction of motion. If this is done then fewer of the optimal trejectories will go in the non-preferred directions and hence the results of extrapolation procedure will be used less often in computing minimum cost.

CHAPTER 4

AN ADVERTISEMENT PROBLEM

The computational aspects of the state increment dynamic programming will be discussed in this chapter. The problem used to illustrate the method is the one of inventory and advertisement system with two state variables and one control variable.

4.1 DEVELOPMENT OF THE MODEL

A diffusion model for advertisement was originally developed by Teichroew (29). The model discussed here is an extension of his model. Consider a particular sales system where in a group of people, only certain members possess a particular piece of information. The group size, i.e. the number of people in the group, is assumed to be constant. Also, the diffusion of information occurs only through personal contact. The number of 'contacts' made by an 'average' person in an arbitrary unit of time and is given as a contact coefficient. It is assumed the same for all the people in the group. A contactee receives the information only if he/she does not already have it; otherwise the contact is wasted as far as increasing the number of informed persons is concerned.

Let $Q(0) = Q_0$ = number of informed persons at time t_i.

N = total number of people in group

C_c = contact coefficient, the number of contacts made by one informed person per unit of time

$Q(t)$ = number of informed persons at time t.

Therefore $\frac{Q(t)}{N}$ = the fraction of informed persons at time t, and

$1 - \frac{Q(t)}{N}$ = the fraction of uninformed persons at time t.

The contacts made during a time interval dt can be given by

$$C_c\ Q(t)\ dt.$$

The increase in the total number of informed people during short time interval Δt is found by multiplying the number of contacts by the proportion of the uninformed persons because an increase in the informed persons can be caused only by the proportion of the uninformed people of the group. Therefore

$$dQ(t) = C_c\ Q(t)\ dt\ (1 - \frac{Q(t)}{N})$$

and

$$\frac{dQ(t)}{dt} = C_c\ Q(t)\ (1 - \frac{Q(t)}{N})\ . \tag{4.1}$$

Suppose the company thinks that it can influence the number of contacts by spending money on advertising and that the rate of contacts by each informed person can be increased by an amount A per unit time, then

$$\frac{dQ(t)}{dt} = [C_c + A(t)]\ Q(t)(1 - \frac{Q(t)}{N})\ . \tag{4.2}$$

If each informed person purchases C_q units of company's product and the sales at the time t can be denoted by S(t), then

$$S(t) = C_q\ Q(t) \tag{4.3}$$

For simplifying, if C_q can be taken to be unity, then

$$S(t) = Q(t) \tag{4.4}$$

Substituting S(t) for Q(t) in Equation 4.2, we have

$$\frac{dS(t)}{dt} = [C_c + A(t)]\ S(t)(1 - \frac{S(t)}{N})\ . \tag{4.5}$$

The rate of change of the company's inventory is given by

$$\frac{dI(t)}{dt} = P(t) - S(t) \tag{4.6}$$

where P(t) = production by time t. The production rate is assumed to be a linear function of time and can be given by

$$P(t) = a + b\,t \tag{4.7}$$

where a and b are constants.

The objective can be defined at this stage. The management desires to maximize profit. In this problem the profit is

Profit = Sales revenue - Inventory carrying cost - Advertisement cost.

In mathematical form it can be written as

$$J = \int_{t_i}^{t_f} [C\,S(t) - C_I(I_m - I(t))^2 - C_A S(t)\,A^2(t)]dt \tag{4.8}$$

where J = net total profit

C = sales revenue

I_m = capacity of storage of inventory

C_I = inventory carrying cost

C_A = advertisement cost

We have explained and derived the mathematical model in differential form. But here we are solving the problem by using state increment dynamic programming, which requires discrete form of the problem. Therefore we shall transform the above differential equations into difference equations.

The system variables are inventory at time t and sales by time t. In difference equations they can be given as

$$I(t + \Delta t) = I(t) + [P(t) - S(t)]\,\Delta t \tag{4.9}$$

and

$$S(t + \Delta t) = S(t) + S(t)[C_c + A(t)]\left[1 - \frac{S(t)}{N}\right] \tag{4.10}$$

By inspection of the equation 4.10 one can see that if it be kept in the same form then $S(t + \Delta t)$ will be greater than N at one stage. This cannot be allowed. Therefore a little modification is done in it. The term $(1 - \frac{S(t)}{N})$ has been replaced by the term $(1 - \frac{S(t + \Delta t)}{N})$. The new equation is

$$S(t + \Delta t) = S(t) + S(t)\,[C_c + A(t)]\left[1 - \frac{S(t + \Delta t)}{N}\right]$$

$$= S(t) + [C_c + A(t)]\,S(t)$$

$$= [C_c + A(t)]\,S(t) \cdot S(t + \Delta t)$$

$$S(t + \Delta t)\,\left[1 + \{\frac{C_c + A(t)}{N}\} \cdot S(t) \cdot \Delta t\right]$$

$$= S(t)\,[1 + \{C_c + A(t)\}]$$

$$S'(t + \Delta t) = \frac{S(t)\,[1 + \{C_c + A(t)\}\,\Delta t]}{1 + S(t)\{\frac{C_c + A(t)}{N}\} \cdot \Delta t} \tag{4.11}$$

Now we have to change the profit function into difference equation form. The result is

$$\int_{n\Delta t}^{(n+1)\,\Delta t} [c\,S(t) - C_I(I_m - I(t))^2 - C_A\,S(t)\,A^2(t)] \cdot dt$$

$$= [c\,S(t) - C_I(I_m - I(t))^2 - C_A\,S(t)\,A^2(t)]\Delta t$$

$$= \ell(\underline{x}, \underline{u}(t), t)\,\delta t \tag{4.12}$$

which is a current stage profit equation in terms of state and control variables. The complete functional equation for state increment dynamic programming can be written as

$$I(\underline{x}, t) = \min_{u \in U} \{\ell(\underline{x}, \underline{u}(t), t)\ \delta t + I[x + f(\underline{x}, \underline{u}(t), t)\ \delta t,\ t + \delta t]\}. \tag{4.13}$$

4.2 DEFINITION OF PROBLEM

The goal is to maximize

$$I(\underline{x}, t) = \min_{u \in U} \{\ell(\underline{x}, \underline{u}(t), t) \cdot \delta t + I[x + f(\underline{x}, \underline{u}(t), t) + I[x + f(\underline{x}, \underline{u}(t), t)\ \delta t, t + \delta t]\}$$

and

$$\ell(\underline{x}, \underline{u}(t), t) \cdot \delta t = [c \cdot S(t) + C_I\ (I_m - I(t))^2 - C_A\ S(t)\ A^2(t)]\ \delta t \tag{4.14}$$

subject to

$$P(t) = a + bt \tag{4.15}$$

$$I(t + \Delta t) = I(t) + [P(t) - S(t)]\ \delta t \tag{4.16}$$

and

$$S'(t + \Delta t) = \frac{S(t)\ [1 + \{C_c + A(t)\}\ \Delta t]}{1 + S(t)\ \{\frac{C_c + A(t)}{N}\} \cdot \Delta t}. \tag{4.17}$$

4.2.1 Numerical Aspects

In order to solve this problem the constants were assumed to have following values

$a = 70$ $N = 150$

$b = 100$ $I_m = 50$

$C_i = 2$ $t_i = 0$

$C = 20$ $t_f = 1$

$C_A = 0.5$

$C_c = 0.15$

Initial conditions are:

$I(0) = 20$

$S(0) = 20$

The maximum amount of advertising at any time has been restricted to a value of 6. This means that $A(t) \leq 6$. This is the constraint on the control variable.

4.3 SOLUTION BY STATE INCREMENT DYMAMIC PROGRAMMING

From the problem solution by quasilinearization (27), it is known that the value of both the state variables, x(t) and S(t) respectively, increase with the time. Hence this will be the preferred direction of motion for them.

Using the block concept, we have three dimensional block of Δx X S X ΔT. The time interval

$$\Delta T = t_f - t_0 = 1,$$

is divided into ten equal parts giving

$$\Delta t = 0.1.$$

Since the preferred direction of motion of x(t) and s(t) is such as to increase with time, we shall first process the blocks with the largest values of x, s and t. First of all, in this block, x and s

are kept constant, and optimal function values for all ten time increments are found out. Then x is kept constant and s is lowered by one increment and then optimal function values are found for each stage (time interval). Then x is lowered by one increment and above procedure is repeated. This goes on till the optimal function is evaluated for all values of x, s, and t.

As shown in Figure 4.1, the adjacent block will have a common boundary with the original one. The just-counted optimum values at the original block boundary will become the initial values for the new block. These values will be used to find the optimum values of the next block.

4.4 PROCEDURE FOR OBTAINING STARTING VALUES FOR THE FIRST BLOCK

To get good accuracy by state increment dynamic programming, the average value of the time interval δt required to change any of the states by one increment should be close to Δt. If it is large or small compared to Δt the accuracy is decreased because of inaccurate approximations due to the interpolations or extrapolations performed. In other words $\frac{\Delta x}{\Delta t} = \frac{\Delta x}{\delta t} = 1$. Such a condition when Δx is small is illustrated in Figure 4.2.

In this figure the optimum values at points p_1 and p_2 are known. The optimum value at point p_0 is to be found from these two values. Since Δx is very small, $n\Delta x$ will be increased by one increment to $(n+1)\Delta x$ on application of control z for a very short time interval δt. As a result the point p_3 is at a far distance from points p_1 and p_2. This will result in an inaccurate optimal value of p_3 based on extrapolation.

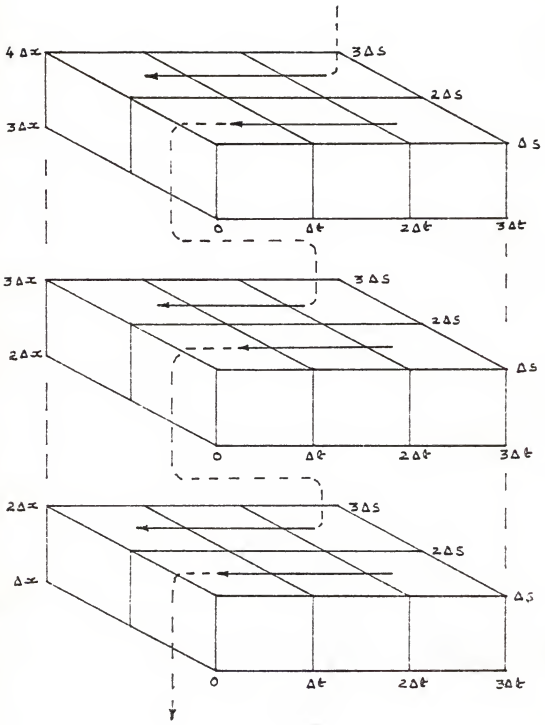

Fig. 4.1 Processing With Too Small ΔX.

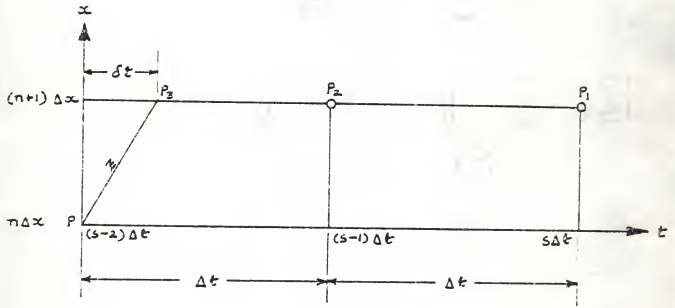

Fig. 4.2 Order Of Processing The Blocks.

Since the state variable increases with time the width of time block $w\Delta x$ is taken equal to Δx and hence $w = 1$.

Figure 4.1 shows in simplified manner the order of processing the blocks.

The computer program in FORTRAN language with a sample printout is shown in Appendix A. It was solved on the ITEL AS/5 computer. The flow chart for the state increment dynamic programming is also shown in Appendix B.

The results for the initial condition were interpolated manually and are given in Table 1. The plots of the results for each visualization are also given in Figures 4.3-4.7.

4.5 RESULTS

The optimal profit in this problem was $J = 891.00$ and the optimal initial and final values are

$I(0) = 20$

$S(0) = 20$

$A(0) = 6.0$

$I(1) = 50.0$

$S(1) = 123.22$

$A(1) = 0.0$

From Table 1 we can see the advantage of using state increment dynamic programming. Since we are using the principle of optimality we get the optimal values of each stage for all parameters. This is done by dividing the solution space into a number of grid points and evaluating optimal values at each grid point where at least one of the state variables changes its state.

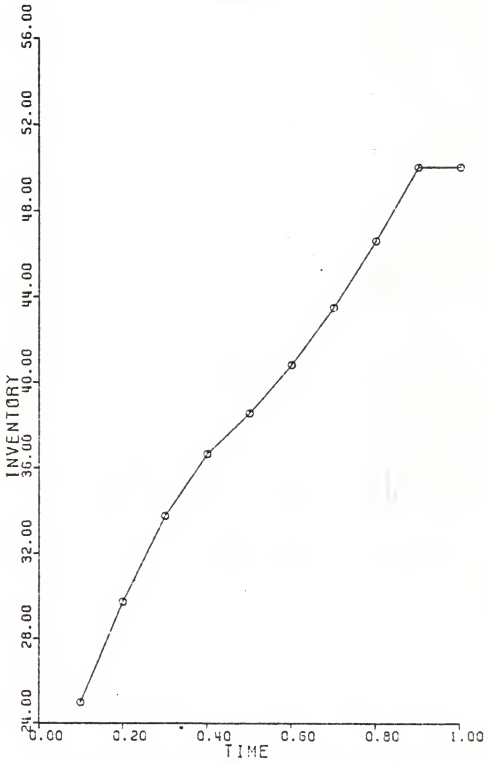

Fig. 4.3 Inventory Level For A Ten Stage Process.

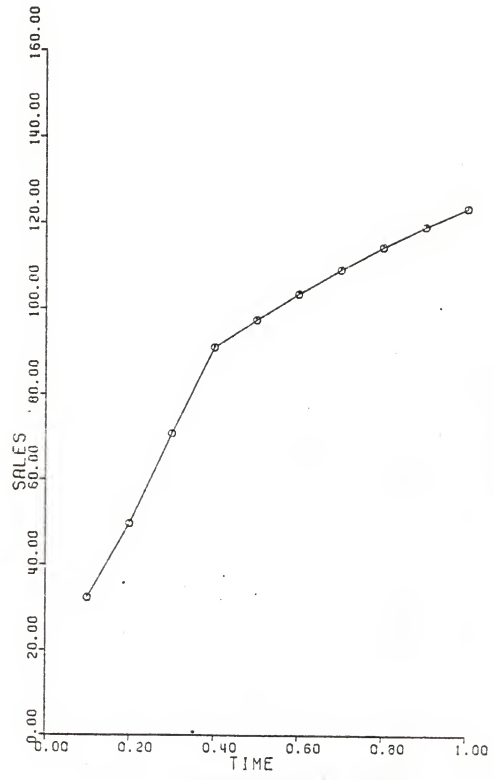

Fig. 4.4 Sales For A Ten Stage Process.

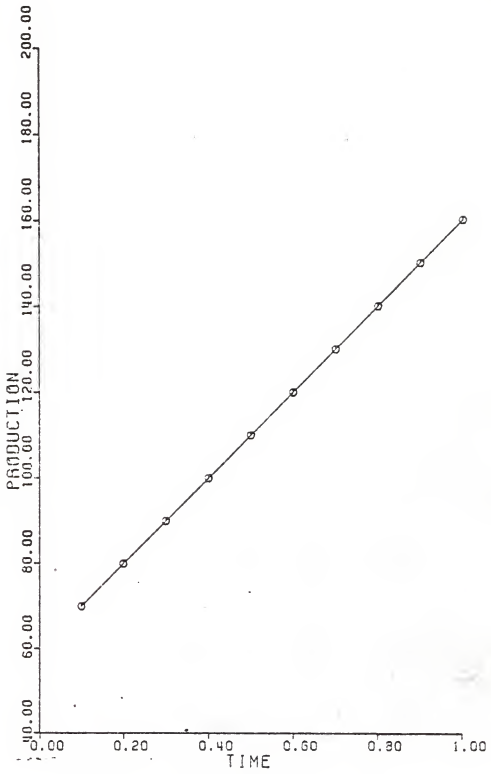

Fig. 4.5 Production For A Ten Stage Process.

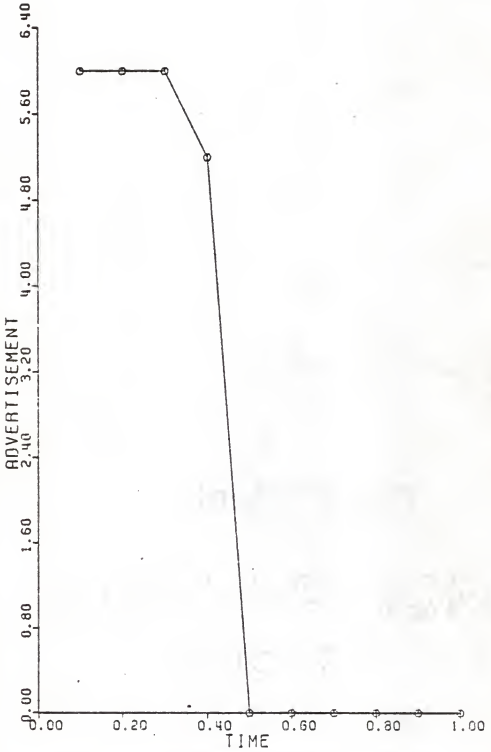

Fig. 4.6 Advertisement For A Ten Stage Process.

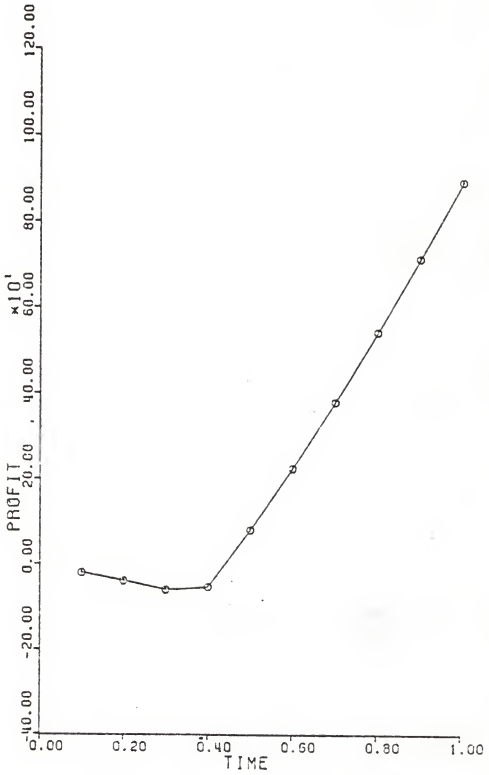

Fig. 4.7 Profit For A Ten Stage Process.

The problem was solved on ITEL AS/5 computing facility at Kansas State University on WATFIV compiler. It took about 3.79 min. to get the overall results.

4.6 DISCUSSION

From Table 1 and Figures 4.3 to 4.7 we can see that the different parameters increase with the time and so does the profit.

The production rate, being a linear function of time, increases with the time, while other parameters like inventory and sales also increase with the time. The amount of advertisement decreases with time and eventually becomes zero. Since the sales increase with time and advertisement decrease with time, the profit, as would be expected, should increase with the time. This easily can be visualized from the plots in above mentioned figures. Also the production is a linear function of time and hence should be maximum at the final stage which would give the maximum profit.

As has been mentioned before the problem with the same model has been solved by P. Shah (26). The optimization technique he used was quasilinearization. Therefore the best way to evaluate the state increment dynamic programming would be to compare our results with his.

Since the constants used here are different, we can not expect the same results as his. But the trend of the optimal values for different stages can be compared. It is found that his results and ours match very well and follow the same pattern. The quasilinearization took about 3.72 minutes to get the completely converged solution. State increment dynamic programming took 3.79 minutes to get the complete solution.

Table 1. Optimal Values of Parameters

Time $t-t+\Delta t$	Inventory $x(t)$	Sales $S(t)$	Inventory $x(t+\Delta t)$	Sales $S(t+\Delta t)$	Production $P(t)$	Advertisement $A(t)$	Profit
0.0 - 0.1	20.00	20.00	25.00	32.50	70	6.0	- 19.50
0.1 - 0.2	25.00	32.50	29.75	49.86	80	6.0	- 38.63
0.2 - 0.3	29.75	49.86	33.78	70.90	90	6.0	- 59.73
0.3 - 0.4	33.78	70.90	36.67	90.98	100	5.2	- 53.18
0.4 - 0.5	36.67	90.98	38.57	97.36	110	0.0	80.62
0.5 - 0.6	38.57	97.36	40.83	103.40	120	0.0	224.70
0.6 - 0.7	40.83	103.40	43.49	109.05	130	0.0	378.36
0.7 - 0.8	43.49	109.05	46.59	114.25	140	0.0	541.30
0.8 - 0.9	46.59	114.25	50.00	118.97	150	0.0	712.50
0.9 - 1.0	50.00	118.97	50.00	123.22	160	0.0	891.00

The great advantage with SIDP is that we got optimal values of objective function at all grid points in the solution space. Thus if one is interested in seeing the optimal values at different stages or states, can get directly from tables. This is helpful in case of constraint changes on stage and state variables. One does not need to go through the whole tedious procedure again and again. Thus little increase in computing cost can be justified.

The same kind of results with probably a little more accuracy could be obtained by the standard dynamic programming algorithm. But the 'dimensionality' difficulty restricts the application. For example 66 fast memories are required for this problem when solved by SIDP while 1200 fast memories would be required for solution by standard algorithm.

As for accuracy, we do not claim that the results are the most accurate ones by this method. The reason is the average time interval over which control is applied should be about the same order of magnitude as the fixed time interval to get very accurate results. But the time interval over which the control is applied is divided by the state variables and the control variables. If one of the state variable changes much more rapidly than the other, then the time interval for the application of the control for that particular variable will be the least and hence it will go on changing all the time. This keeps the other state variables unchanged, and the problem moves in a plane rather than space. This brings in accuracy in the results due to interpolations.

Moreover SIDP is a fairly new method. There have not been very much researches on it nor computational experiences. Some of the assumptions make its use restrictive. One of them is mentioned above

while the other one is movement of the objective in the preferred direction. We do not have any answers for the consequences of moving in non-preferred direction.

In spite of these restrictions, SIDP still is a versatile method and can be used very successfully where the standard algorithm does not work very well. It has extremely good potential and more development can make it very powerful technique.

APPENDIX A

Mathematical expressions for journal bearing operative characteristics:

1. $S \leq 0.15$

   i) $0.25 \leq L/D \leq 0.5$

$$h_0 = 1.585\, C(L/D)^{0.913}\, (S)^{0.655(L/D)^{0.0922}}$$

$$\Delta t = \frac{0.5}{(L/D)^{0.374}}\, P(S)^{0.695/(L/D)^{0.139}}$$

$$\frac{P}{P_{max}} = 0.76\, (L/D)^{0.62}\, (S)^{0.24}$$

$$\frac{RNCL}{Q} = 0.128(L/D)^{0.048}\, (S)^{0.1(L/D)^{0.47}}$$

$$+ (R/C) = \frac{12.6}{(L/D)^{0.41}}\, (S)^{0.62/(L/D)^{0.1035}}$$

   ii) $0.5 \leq L/D \leq 1$

$$h_0 = 1.84C\, (L/D)^{1.13}\, (S)^{0.731(L/D)^{0.252}}$$

$$\Delta t = \frac{0.43}{(L/D)^{0.62}}\, P(S)^{0.56/(L/D)^{0.302}}$$

$$\frac{P}{P_{max}} = 0.76\, (L/D)^{0.62}\, (S)^{0.294(L/D)^{0.292}}$$

$$\frac{RNCL}{Q} = 0.128\, (L/D)^{0.048}\, (S)^{0.06/(L/D)^{0.212}}$$

$$f(R/C) = \frac{11.8}{(L/D)^{0.503}}\, (S)^{0.62/(L/D)^{0.1035}}$$

2. $S \geq 0.15$

i) $0.25 \leq \frac{L}{D} \leq 0.5$

$$h_0 = 1.035\ C\ (L/D)^{0.673}\ (S)^{0.33/(L/D)^{0.2}}$$

$$\Delta t = \frac{0.695}{(L/D)^{0.214}}\ (S)^{0.875(L/D)^{0.042}}$$

$$\frac{P}{P_{max}} = 0.76\ (L/D)^{0.62}\ (S)^{0.24}$$

$$\frac{RNCL}{Q} = 0.128\ (L/D)^{0.048}\ (S)^{0.1(L/D)^{0.47}}$$

$$f(R/L) = \frac{16.85}{(L/D)^{0.318}}\ (S)^{0.922(L/D)^{0.087}}$$

ii) $0.5 \leq L/D \leq 1$

$$h_0 = 0.95\ C\ (L/D)^{0.556}\ (S)^{0.375}$$

$$\Delta t = \frac{0.695}{(L/D)^{0.214}}\ (S)^{0.875(L/D)^{0.042}}$$

$$\frac{P}{P_{max}} = 0.55\ (L/D)^{0.1535}\ (S)^{0.083/(L/D)^{1.535}}$$

$$\frac{RNCL}{Q} = 0.128\ (L/D)^{0.048}\ (S)^{0.06/(L/D)^{0.212}}$$

$$f(P/C)\ \frac{19}{(L/D)^{0.127}}\ (S)^{0.922}\ (L/D)^{0.087}$$

APPENDIX B

```
// EXEC WATFIV
//SYSIN     DD *
$JOB                ,TIME=(5,),PAGES=250
      DIMENSION SB(51,51),SP(51,51)
      DX=5
      DR=5
      DA=.2
      DT=.1
      NT=10
      NX=21
      NA=31
      NR=30
      PI=50
      AN=150.2
      AB=70
      B=100
      C=2
      F=20
      CI=.15
      CA=0.5
      XMAX=100
      PRES=-988888
      NRR=NR+1
      NXX=NX+1
      LINENO=0
      NBLOCK=0
      NS=N
  100 FORMAT (1H1,'STAGE NO       TIME      X(N-1)      R(N-1)      X(N)
     1    R(N)          P(N)          A(N)          PROFIT')
  101 FORMAT (1H ,5X,I2,8(4X,F7.2))
  102 FORMAT (1H ,'BLOCK NO = ',I3)
      PRINT 100
      DO 71 K=1,51
      DO 71 L=1,51
      SB(K,L)=0
   71 SP(K,L)=0
      DO 10 KX=1,NX
      X=NX-KX
      X=X*DX
      DO 11 KR=1,NR
      R=NRR-KR
      IR=R
      R=R*DR
      I=1
      P=AB+B*(I-DT)
      NBLOCK=NBLOCK+1
      PRINT 102,NBLOCK
      DO 25 KA=1,NA
      A=KA-1
      A=A*DA
      XO=X+(P-R)*DT
      RO=(R*(1.+(C+A)*DT))/(1.+R*(C+A)*DT/AN)
      IF (XO.LT..001)GO TO 25
      SNEW=(R**2-(PI-X)**2*(1-CA*A**2*R)*DT
      IF (SNEW.LT.PRES) GO TO 25
      PRES=SNEW
      AA=A
      XX=XO
      RR=RO
   25 CONTINUE
```

```
      IF (LINENO.LT.45) GO TO 80
      LINENO=0
      PRINT 100
80    PRINT 101,NS,T,X,R,XX,RR,P,AA,PRES
      LINENO=LINENO+1
      SP(NS,IR)=PRES
42    PRES=-988888
      NS=NS-1
      T=NS*DT
      P=AA+B*(T-DT)
      DO 22 KA=1,NA
      A=KA-1
      A=A*DA
      U=P-R
      IF (U.EQ.0) GO TO 90
      DELTX=ABS(DX/(P-R))
      GO TO 91
90    DELTX=999
91    DELTR=ABS(DR/(R*(C+A)*(1.-(R+DR)/AN)))
      IF (DELTX.GT.DELTR) GO TO 51
      DELT=DELTX
      DELR=DELT*R*(C+A)*(1.-R/AN)
      XO=X+(P-R)*DT
      RO=(R*(1.+(C+A)*DT))/(1.+R*(C+A)*DT/AN)
      S1=(R**2-(DT-X)**2*(1.-CA*A**2*R)*DELT
      R1=SR(NS+1,IR)+(SR(NS+1,IR+1)-SR(NS+1,IR))*DELR/DR
      R2=SR(NS+2,IR)+(SR(NS+2,IR+1)-SR(NS+2,IR))*DELR/DR
      S2=R1-(R1-R2)*(DELT-DT)/DT
      SNEW=S1+S2
      GO TO 52
51    DELT=DELTR
      DELX=DELT*(P-R)
      XO=X+(P-R)*DT
      RO=(R*(1.+(C+A)*DT))/(1.+R*(C+A)*DT/AN)
      S1=(R**2-(DT-X)**2*(1.-CA*A**2*R)*DELT
      X1=SP(NS+1,IR+1)+(SR(NS+1,IR+1)-SP(NS+1,IR+1))*DELX/DX
      X2=SP(NS+2,IR+1)+(SR(NS+2,IR+1)-SP(NS+2,IR+1))*DELX/DX
      S2=X1-(X1-X2)*(DELT-DT)/DT
      SNEW=S1+S2
52    IF(SNEW.LT.PRES) GO TO 22
      PRES=SNEW
      AA=A
      XX=XO
      RR=RO
22    CONTINUE
      IF (LINENO.LT.45) GO TO 81
      LINENO=0
      PRINT 100
81    PRINT 101,NS,T,X,R,XX,RR,P,AA,PRES
      LINENO=LINENO+1
      SP(NS,IR)=PRES
      IF(NS.LE.1) GO TO 43
      GO TO 42
43    T=1
      NS=NT
      PRES=-988888
11    CONTINUE
      DO 31 LNS=1,NT
      DO 31 LR=1,NR
31    SR(LNS,LR)=SP(LNS,LR)
```

```
      I=1
      NS=NT
      PRES=-988388
   10 CONTINUE
      STOP
      END
$ENTRY
```

REFERENCES

1. Aris, R., "The Discrete Dynamic Programming," Blaisdell, New York, 1964.

2. Bartel, D. L. and R. W. Marks, "The Optimum Design of Mechanical Systems with Competing Design Objectives," Journal of Engineering for Industry, Transactions of ASME, 1974 (February), pp. 171-178.

3. Bellman, R. E., "Dynamic Programming," Princeton University Press, Princeton, New Jersey, 1957.

4. Bellman, R. E., "Adaptive Control Processes," Princeton University Press, Princeton, New Jersey, 1962.

5. Bellman, R. E. and S. Dreyfus, "Applied Dynamic Programming," Princeton University Press, Princeton, New Jersey, 1962.

6. Booker, J. F., "Dynamically - Loaded Journal Bearings: Mobility Method of Solution," Journal of Basic Engineering, Transactions of ASME, 1965 (September), pp. 537-546.

7. Brabb, G. J., "Introduction to Quantitative Management," Holt, Rinehart and Winston, Inc., New York, 1968.

8. Buffa, E. S., "Operations Management: The Management of Productive Systems," John Wiley & Sons, Inc., New York, 1976.

9. Dreyfus, S. E., "Computational Aspects of Dynamic Programming," Operations Research, Vol. V, No. 3, 1957 (June), pp. 409-415.

10. Eason, E. D. and R. G. Fenton, "A Comparison of Numerical Optimization Methods, for Engineering Design," Journal of Engineering for Industry, Transactions of ASME, 1974 (February), pp. 196-200.

11. Fan, L. T., C. L. Hwang, and F. A. Tillman, "A Sequential Simplex Pattern Search Solution to Production Planning Problems," AIIE Transactions, Vol. 1, No. 3, 1969 (September), pp. 267-273.

12. Fedor, J. V., "Half Somerfeld Approximation for Finite Journal Bearings," Journal of Basic Engineering, 1963 (September), pp. 435-438.

13. Hadley, G., "Nonlinear and Dynamic Programming," Addison Wesley Publishing Co., Reading, Massachusetts, 1964.

14. Hilderbrand, F. B., "Introduction to Numerical Analysis," McGraw Hill, New York, 1956.

15. Larson, R. E., "State Increment Dynamic Programming," American Eisevier Publishing Company, Inc., New York, 1968.

16. Larson, R. E., "State Increment Dynamic Programming; Theory and Applications," Proc. of 2nd Allerton Conf. on Ckt. and System Theory, University of Illinois, 1964 (September), pp. 643-665.

17. Larson, R. E., "Dynamic Programming with Reduced Computational Requirements," IEEE Transactions on Automatic Control, Vol. AC-10, No. 2, 1965 (April), pp. 135-143.

18. Larson, R. E., "An Approach to Reducing the High Speed Memory Requirement of Dynamic Programming, J. Math. Anal. and Appl., Vol. 11, Nos. 1-3, 1965 (July), pp. 519-537.

19. Lee, E. S., "Quasilinearization and Invariant Imbedding," Academic Press, New York, 1968.

20. Lee, E. S. and P. Shah, "Optimization of Production Planning by Generalized Newton-Raphson Method," AIIE Transactions, Vol. II, No. 1, 1970 (March), pp. 1-10.

21. Lee, E. S., "Dynamic Programming," Class Notes, Kansas State University, Manhattan, Kansas 1978.

22. Martin, F. A., and J. F. Booker, "Influence of Engine Inertia Forces on Minimum Film Thickness in Con-Rod Big-End Bearing," The Institution of Mechanical Engineers, Proceedings, Vol. 181, Part I, No. 30, 1966-67, pp. 749-764.

23. Nelder, J. A. and R. Mead, "A Simplex Method for Function Minimization," The Computer Journal, Vol. 7, No. 4, 1965 (January), pp. 309-313.

24. Nemhauser, G. L., "Introduction to Dynamic Programming," John Wiley, New York, 1966.

25. Raimondi, A. A. and J. Boyd, "A Solution for the Finite Journal Bearing and Its Application to Analysis and Design, Part I, Part II, Part III, "Transactions of ASLE," Vol. I, No. 1, 1958 (April), pp. 159-203.

26. Seireg, A. and H. Ezzat, "Optimum Design of Hydrodynamic Journal Bearing," Journal of Lubrication Technology, Transactions of ASME, Series F, Vol. 91, No. 3, 1969 (July), pp. 516-523.

27. Shah, P., "Application of Quasilinearization to Industrial Management Systems," Master's Thesis, Kansas State University, Manhattan, Kansas, 1969.

28. Spendley, W., G. R. Hext, and F. R. Himsworth, "Sequential Application of Simplex Designs in Optimization and Evolutionary Operation," Technometrics, Vol. 4, No. 4, 1962 (November), pp. 441-459.

29. Teichroew, D., "An Introduction to Management Science, Deterministic Models," John Wiley and Sons, Inc., New York, 1966 (November).

ACKNOWLEDGMENT

The author wishes to express his deep sense of appreciation to his major professor, Dr. E. S. Lee for his constant guidance, constructive criticism, helpful suggestions and the personal interest taken in the preparation of this master's thesis.

STATE INCREMENT DYNAMIC PROGRAMMING AND THE INDUSTRIAL MANAGEMENT SYSTEMS

by

ANSHUMAN KRISHNAKANT DESAI

B.E. (Mech.), Gujarat University
Ahmedabad, India, 1977

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1979

ABSTRACT

The importance of quantitative techniques in decision making emphasizes the need of efficient techniques as a tool for solving management problems. The most difficult are the boundary value problems with nonlinear differential and/or difference equations. The non-linearity in the system equations does not allow the application of superposition principle.

One of the most important techniques developed in recent years for the solution of optimization problems is R. Bellman's dynamic programming. This technique solves at least in principle a large number of important optimization problems. However, because of extremely large amount of fast storage memory requirement, called by Bellman "curse of dimensionality", only relatively simple problems can be solved on existing computers.

State increment dynamic programming, developed in 1965 by R. E. Larson, requires considerably less fast-access memory but it still retains the general applicability and other desirable features of the standard algorithm.

In this thesis first a brief introduction and computational procedure of state increment dynamic programming are given. Then its application to advertisement production problem with two state variables and a control variable is duscussed in detail.

The production planning with consideration of advertisement provides a good base of comparison of this method with others used to solve the problems with same model in the past. The advantages and disadvantages of state increment dynamic programming have been highlighted.