10
I29 C
#24
cop. 1

UILU-ENG-71-2008
~~UIU-ENG-71-2003~~

CIVIL ENGINEERING STUDIES

HYDRAULIC ENGINEERING SERIES NO. 24

WATER RESOURCES SYSTEMS ANALYSIS BY DISCRETE DIFFERENTIAL DYNAMIC PROGRAMMING

by

MANOUTCHEHR HEIDARI, VEN TE CHOW, and DALE D. MEREDITH

Supported by

OFFICE OF WATER RESOURCES RESEARCH
U.S. DEPARTMENT OF THE INTERIOR

and

UNIVERSITY OF ILLINOIS WATER RESOURCES CENTER

DEPARTMENT OF CIVIL ENGINEERING
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS
JANUARY, 1971

UILU-ENG-71-200
~~UILU-ENG-71-2003~~

WATER RESOURCES SYSTEMS ANALYSIS BY DISCRETE DIFFERENTIAL DYNAMIC PROGRAMMING

by

Manoutchehr Heidari, Ven Te Chow, and Dale D. Meredith

Project No. B-030-ILL

The work upon which this publication is based was supported by funds provided by the U.S. Department of the Interior as authorized under the Water Resources Research Act of 1964, P.L. 88-379
Agreement No. 14-01-0001-1899
and by the
University of Illinois Water Resources Center

DEPARTMENT OF CIVIL ENGINEERING
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801
JANUARY, 1971

FOREWORD

The optimization of operating policies of multiple-unit and multiple-purpose water resources systems by traditional dynamic programming with the use of high-speed digital computers encounters two major difficulties: excessive memory requirements, and tremendous amounts of computer time requirements. This report describes an iterative method named discrete differential dynamic programming (DDDP) which can ease the above difficulties considerably. The method starts with a trial trajectory satisfying a specific set of initial and final conditions and applies Bellman's recursive equation in the neighborhood of this trajectory. At the end of each iteration step a locally improved trajectory is obtained and used as the trial trajectory in the next step. It is shown that the method is particularly effective in the case of so called "invertible" systems. The merits of the proposed approach are demonstrated through its application to two four-unit, two-purpose water resources systems. To save computer time the examples are restricted to deterministic hydrologic inputs.

This study was performed as part of a research project on "Advanced Methodologies for Water Resources Planning" sponsored by the U.S. Department of the Interior as authorized under the Water Resources Act of 1964, P.L. 88-379 Agreement No. 14-01-0001-1899. A major portion of the work performed by the first author was financially supported by the Illinois State Ge-logical Survey. The authors wish to thank those, particularly Dr. Petar V. Kokotović, who have contributed valuable comments on this study.

TABLE OF CONTENTS

Page

Forward ii

1. Introduction 1

1-1. The General Problem 1

1-2. Objective of the Study 3

1-3. Scope of the Study 4

2. A Review of Dynamic Programming 5

2-1. Dynamic Programming 5

2-1-1. Definitions 5

2-1-2. Development of Recursive Equation and Computational Procedures 6

2-1-3. Applications of Dynamic Programming 13

2-2. Advantages of Dynamic Programming 14

2-2-1. Analysis of Multiple-Stage Processes 14

2-2-2. Incorporation of Stochastic Disturbances 14

2-2-3. Incorporation of Constraints 15

2-2-4. Incorporation of Nonlinearity 15

2-3. Disadvantages of Dynamic Programming 15

2-3-1. Dimensionality 16

2-3-2. Computer Time Requirements 17

2-4. Techniques to Reduce the Dimensionality of Dynamic Programming 18

2-4-1. Successive Approximation 18

2-4-2. State Increment Dynamic Programming 20

2-4-3. Differential Dynamic Programming 24

Page

3. A Discrete Differential Dynamic Programming (DDDP) Approach for Water Resources Systems Analysis 25

3-1. Differential Dynamic Programming 25

3-1-1. Differential Dynamic Programming-Theory 25

3-1-2. Discrete Differential Dynamic Programming 30

3-2. Extension of the Proposed Approach to Systems with Stochastic Inflows 36

4. Invertibility of System Equations in Water Resources Systems 38

5. Application of the Discrete Differential Dynamic Programming (DDDP) Approach 42

5-1. A Simplified Water Resources System 42

5-1-1. The Problem and Its Solution by Other Techniques 42

5-1-2. Application of the Proposed Approach to the Problem 47

5-1-3. Effect of Different Trial Trajectories 53

5-1-4. Expansion of Time Horizon 53

5-1-5. Effect of Change of the State Sub-Domains 53

5-2. Advantages and Disadvantages of the Proposed Approach . . 58

5-3. Application of the Proposed Approach to Clearwater River System 61

5-3-1. The Selection of the System 61

5-3-2. The System 62

5-3-3. System's Formulation 65

5-3-4. Return and Cost Functions 67

5-3-4-1. Irrigation 67

5-3-4-2. Hydropower 72

5-3-5. Hydrology of the System 76

Page

5-3-6. Computer Program 78

5-3-7. Constraints and Parameters Used in the Analysis 79

5-3-8. Discussion of Analysis 80

6. Concluding Remarks 89

6-1. Summary and Conclusions 89

6-2. Recommendations for Future Studies 91

6-2-1. Regarding the Approach 91

6-2-1-1. Choice of Trial Trajectory 91

6-2-1-2. Choice of the State Sub-Domains 91

6-2-2. Choice of Return Functions 92

6-2-3. Regarding the Hydrology 93

6-2-4. Regarding the System Optimization 93

7. References 94

Appendix A. Computer Programs and Description of Program Input and Output 97

A-1. Notation Used to Input Data for Program to Analyze Clearwater 97

A-2. FORTRAN IV Program Used to Analyze the Clearwater River System 99

A-3. Notation Used in Output Tables for Clearwater River System .106

A-4. FORTRAN IV Program Used to Analyze the System Described in Figure 7 113

Appendix B. Notations 116

LIST OF FIGURES

Figure Page

1. Schematic Representation of Possible Decisions Which Bring the State of the System to c_1 at n=1. (Forward Algorithm) . . . 9

2. Determination of δt for State Increment Dynamic Programming (Forward Algorithm) 22

3. Examples of State Sub-Domains at Stage n 32

4. Schematic Representation of a Trial Trajectory, the Boundaries Defining Corridor C_k, and Optimal Trajectory in C_k of k-th Iteration for a System with m = 1 and T = 3 33

5. Flow Chart Showing Steps of the DDDP Approach 35

6. Possible Decision Paths Leading to State $s(n) + \delta s_{2,1}(n)$ from Stage n-1 for a System with m = 1 and T = 3 41

7. Reservoir Network of the Simplified System 43

8a. Convergence to Optimal Trajectory from Trial Trajectory I for Reservoir 1 in Figure 7 49

8b. Convergence to Optimal Trajectory from Trial Trajectory I for Reservoir 2 in Figure 7 50

8c. Convergence to Optimal Trajectory from Trial Trajectory I for Reservoir 3 in Figure 7 51

8d. Convergence to Optimal Trajectory from Trial Trajectory I for Reservoir 4 in Figure 7 52

9. Trial Trajectories I, II, and III, and Optimal Trajectory of System in Figure 7 Using $\sigma_{j,1} = 1.0$, $\sigma_{j,2} = 0.0$, and $\sigma_{j,3} = -1.0$ for j = 1, 2, 3, 4 54

10. Total Return, F, as a Function of Number of Iterations for Trial Trajectories I, II, and III 55

11. Trial and Final Trajectories IV for System in Figure 7 During 24 Periods of Operation 56

12. Near-Optimal Trajectory Shown by Solid Circles and Optimal Trajectory of System in Figure 7 Using Trial Trajectory I and $\sigma_{j,1} = +1.3$, $\sigma_{j,2} = 0.0$, and $\sigma_{j,3} = -1.3$ for j = 1, 2, 3, 4 59

13. Schematic Representation of the Clearwater River System 63

Figure Page

14. Assumed Unit Gross Irrigation Return Function 71

15. Assumed Penalty Function for Irrigation Shortage 71

16. Trial and Final Trajectories A for System in Figure 13 During 12 Months of Operation 81

17. Trial and Final Trajectories B for System in Figure 13 During 12 Months Operation 82

18. Return vs. Iteration Number for 12 Operating Periods of System in Figure 13 Using Trial Trajectory A 83

19. Return vs. Iteration Number for 12 Operating Periods of System in Figure 13 Using Trial Trajectory B 84

20. Trial and Final Trajectories C for System in Figure 13 During 24 Months of Operation 85

21. Return vs. Iteration Number for 24 Operating Periods of System in Figure 13 Using Trial Trajectory C 86

LIST OF TABLES

Table Page

1. Return Functions Used to Calculate Optimal Policies of System in Figure 7 45

2. Optimal Trajectory and Policy for System in Figure 7 Total Return = 401.3 46

3. Trial Trajectory I and Trial Policy I for System in Figure 7 Total Return = 362.5 47

4. Computer (IBM 360/75) Time Requirements of the Proposed Approach for the Solution of System in Figure 7 60

5. Assumed Monthly Distribution of the Annual Target Irrigation Diversion Requirements 68

6. Assumed Monthly Distribution of Annual Total Energy Output . . 73

7. Power and Energy Relationships Used in Analysis of Clearwater River System 75

8. Mean Monthly Stream Flows of Streams in Figure 13 77

9. Production and Returns Due to Different Trajectories for System in Figure 13 88

A-1. Trial Trajectory A and Detailed List of Activities 107

A-2. Final Trajectory A and Detailed List of Activities 108

A-3. Trial Trajectory B and Detailed List of Activities 109

A-4. Final Trajectory B and Detailed List of Activities 110

A-5. Trial Trajectory C and Detailed List of Activities 111

A-6. Final Trajectory C and Detailed List of Activities 112

1. INTRODUCTION

1-1. The General Problem

The development and usage of water resources systems consist of three basic phases: (1) the planning phase in which the location and extent of the different system components are determined, (2) the allocation phase in which the proportion of resources to be allocated to each purpose is determined, and (3) the operation phase in which the operation of the system is specified for achieving the allocation of resources in order to satisfy particular demands during particular time periods. The operating policies of a multiple-unit and multiple-purpose water resources system play a major role in the overall productivity of the system. A wrong planning approach in the first phase may jeopardize the efficiency of the allocation and operation phases. An unrealistic allocation of water to some purpose may contradict the planning and operating rules. Attempts to find the best operating rules, subject to severe physical and allocation constraints set in the planning and allocation phases, may lead to results which do not project the true potential of the system as a whole. In brief, consideration of only one of these phases for optimization may lead to sub-optimization. Therefore, an optimization of the system must incorporate all three phases simultaneously.

In such an optimization, the basic questions are: where to start, i.e., which phase in conjunction with the results of the other two phases must be considered first, and, since the nature of these phases differ from each other, which optimization technique must be used for each phase?

It is perhaps reasonable to assume that the answer to the first question is that the third phase can be more easily carried out with regard to the results of the other two phases. Therefore, one may start by assuming results for phases one and two, and using them in phase three. Then, based on the results of phase three, it is possible to improve on the results of the first two phases and repeat the last phase again. Continuing in this manner, a set of effective results may be obtained for the three phases.

The answer to the second question must depend on the characteristics of water resources systems. Some of these characteristics are: (1) the stochastic nature of inflow data; (2) the existence of relatively severe constraints; (3) the non-linearity of objective functions; and (4) the multiple-stage nature of the operation procedure of the systems. Therefore, an optimization technique which can handle the above characteristics properly must be selected for the analysis of the systems. Optimization techniques such as the conjugate gradient [Fletcher and Powell, 1963; Fletcher and Reeves, 1964] and the second variation method [Bryson and Ho, 1969] may be adopted to search among a sequence of feasible decisions for the optimal set of decisions. However, these techniques require a sufficiently differentiable objective function, cannot handle constraints without creating difficulties, and require a major modification of the algorithm in order to incorporate the stochastic disturbances.

In constrast, dynamic programming does not have the above limitations. When expressed in discrete form, its recursive equation can even handle functions defined by tables. Constraints on states and decisions reduce the computation efforts, and stochastic disturbances may be incorporated with little modification in the algorithm except with an increase in computer time. It is a technique specifically designed for analysing multiple-stage processes.

1-2. Objective of the Study

Several attempts have been made to determine the optimum operating rules for water resources systems using Bellman's principle of optimality [Chow and Meredith, 1969a]. The choice of this principle seems to be a logical one. However, in using Bellman's principle of optimality, which is a powerful optimization technique for low-dimensional systems, the investigators have generally concluded that with the present state of computer technology a rigorous analysis of multiple-unit and multiple-purpose systems is not possible. This conclusion results from the need for large high-speed computer storage requirements for the solution of the recursive equation of dynamic programming.

In spite of this prohibitive limitation, dynamic programming is found to be useful for the analysis of many water resources systems. Therefore, techniques based on this principle must be sought to overcome the above difficulty. Several investigators, including Bellman [1969], have attempted to develop modified algorithms of dynamic programming which could curb what Bellman [1957] calls the "curse of dimensionality." Some of the attempts have been rather successful [Mayne, 1966; Larson, 1968; Wong and Luenberger, 1968; Jacobson, 1968a, b, c; and Lee, 1969]. However, the application of these modified algorithms to water resources systems has been made only by Larson [1968] and Larson and Keckler [1967, 1969].

Therefore, the objective of this study is to develop an approach based upon the dynamic programming technique which can be used to determine the optimum operating policies for multiple-unit, multiple-purpose, discrete water resources systems using available computer facilities.

1-3. Scope of the Study

The approach proposed for the analysis of discrete water resources systems may be considered an extension of the differential dynamic programming approach that Mayne [1966] and Jacobson [1968a, b, c] have developed for continuous systems. First, the advantages and limitations of dynamic programming are reviewed along with the available techniques for overcoming some of the limitations. Second, the discrete differential dynamic programming technique and the computation steps of the proposed approach are given. The approach is then employed to obtain the best operating policies of two multiple-unit and multiple-purpose water resources systems whose solutions cannot be determined with regular dynamic programming. The first system is a hypothetical case which was formulated by Larson [1968] and for which the exact solution has been found by linear programming and by successive approximation dynamic programming. The other system is the Clearwater River System presented by Maass, et al. [1962], and is considered much closer to a real system than the first case. In order to save computer time these systems are analysed deterministically. The possiblity of extending the proposed approach to stochastic systems is discussed. It is demonstrated that the proposed approach can substantially reduce the high speed computer storage and computer time requirements.

2. A REVIEW OF DYNAMIC PROGRAMMING

2-1. Dynamic Programming

Bellman [1953] published the first formal presentation of an optimization technique called dynamic programming. The principle of dynamic programming, its advantages, and its disadvantages for analysing water resources systems and attempts to reduce the disadvantages are reviewed below.

2-1-1. Definitions

Dynamic programming is a tool for optimizing mathematical representations of multiple-stage processes. The formulation of dynamic programming is based on Bellman's principle of optimality [Bellman, 1957] which states: "An optimal policy has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." Before formulating the functional equation on the basis of the principle of optimality, some definitions are necessary. These definitions can be given by describing the features which characterize the problems to which the dynamic programming approach can be applied. These four features are:

(1) The problem must be one which can be divided into stages with a decision required at each stage. The stages may represent different points in space, as for example in selecting a route for a new pipeline, or they may represent different points in time, as for example in determining the optimal releases each month from a reservoir.

(2) Each stage of the problem must have a finite number of states associated with it. The states describe the possible conditions in which the system might find itself at any stage of the problem. In reservoir

operation studies, the states may represent the amount of water stored in the reservoir at that stage.

(3) The effect of a decision at each stage of the problem is to transform the current state of the system into a state associated with the next stage. The decision may represent how much water to release from the reservoir at the current time, and this decision will transform the amount of water stored in the reservoir from the current amount to a new amount for the next stage. Associated with each potential state transformation is a return, a benefit or a cost, which indicates the effectiveness of the transformation.

(4) For a given current state and stage of the problem the optimal sequence of decisions is independent of the decisions made in previous stages. A policy is a set of decisions which contains one decision for each state variable for each stage. A policy may also be called a decision trajectory. The set of states which results from the application of a policy is called a state trajectory or simply trajectory. An optimal policy is the set of decisions that optimizes the objective function which is a measure of effectiveness of the state transformations and hence the policy.

2-1-2. Development of Recursive Equation and Computational Procedures

In most discussions of dynamic programming computational procedures, the computations begin at the final time and work backwards toward the initial time. This process is known as the backward dynamic programming algorithm. It is possible to start instead with the initial time and work towards the final time using a procedure called forward dynamic programming. By interpreting the computations in a suitable manner, it can be shown that this procedure is more useful in water resources system studies than the backward method. We shall confine ourselves to the forward algorithm which is given below.

Let us consider the dynamic equation of a discrete system in the time interval $t_0 \le t \le t_f$ (i.e., $t \in [t_0, t_f]$). If this time interval is divided into N equal segments of small length Δt, and at the beginning of each time interval, called stage, only discrete values of states and decisions are considered, then the difference equation describing the dynamics of the system is:

$$s(n) = \phi[s(n-1), u(n-1), n-1] \text{ for } n = 1, 2, \ldots, N \tag{1}$$

where n is the index of stage variable; s(n) is an m-dimensional state vector at stage n; u(n) is q-dimensional decision vector at stage n which transforms the state of the system from s(n) to s(n+1); and

$$\begin{aligned} s(n) &\in S(n) \\ u(n) &\in U(n) \end{aligned} \tag{2}$$

where S(n) is the admissible domain in the state space at stage n; and U(n) is the admissible domain in the decision space at stage n. From equation (1) we may write:

$$s(n-1) = \theta[s(n), u(n-1), n-1] \tag{3}$$

If the state of the system at stage n = 0 is a(0), the application of a sequence of decision vectors to this system in the time span between n = 0 and n = N will transform the state of the system to some $s(N) \in S(N)$ at stage N and produce some measure of effectiveness of policy; i.e., a return $F[s(N),N]$. If the objective criterion is to maximize the return from the system, the objective function may be written as:

$$\text{Maximize } F[s(N),N] = \sum_{n=1}^{N} R[s(n-1),u(n-1),n-1] \tag{4}$$

where $F[s(N),N]$ is the sum of the returns which results from series of transformations from some initial state at n = 0 to some final state at n = N; and $R[s(n-1),u(n-1),n-1]$ is the return from the system due to the system being in state s(n-1) at stage n-1 and the application of a decision vector u(n-1) in the time interval starting at stage n-1 and lasting Δt.

Assume that the initial values of $F^*[s(0),0]$ (the superscript * signifies the optimum) for all $s(0) \in S(0)$ are known and that computations are being performed for a stage representing $t = t_0 + \Delta t$, i.e., stage n = 1. The portion of equation (4) for this time is designated by $F^*[s(1),1]$ and may be written as:

$$F^*[s(1),1] = \max_{u(0) \in U(0)} F[s(1),1] \tag{5a}$$

$$= \max_{u(0) \in U(0)} \{R[s(0),u(0),0] + F^*[s(0),0]\} \tag{5b}$$

Substituting equation (3) into equation (5b) we obtain

$$F^*[s(1),1] = \max_{u(0) \in U(0)} \{R[\theta,u(0),0] + F^*[\theta,0]\} \tag{6}$$

which for every discrete level of state at n = 1, s(1), may be solved as a function of u(0) only. Therefore, first the admissible state domain at n = 1, S(1), is discretized into L_i levels in the i-th component of the state vector, i = 1, 2, ..., m, and the admissible decision domain at n = 0, U(0), is divided into H_j discrete levels in the j-th component of the decision vector, j = 1, 2, ..., q. Now, a lattice point, s(1), in the discretized state domain may be chosen and all of the admissible discrete levels of the decision vector may be applied to this level of the state vector to determine which decision vector maximizes equation (6). For each u(0) in the decision domain U(0) the first term on the right side of equation (6) is calculated directly and the second term is usually obtained by interpolation in $F^*[s(0),0]$. The values of the sums obtained for the various u(0) are compared to determine the maximum. This procedure is then repeated for each discrete value of s(1).

In Figure 1, which represents the state domain for m = 1 and q = 1, such a state lattice point at stage n = 1 is shown as c_1. Discrete levels of

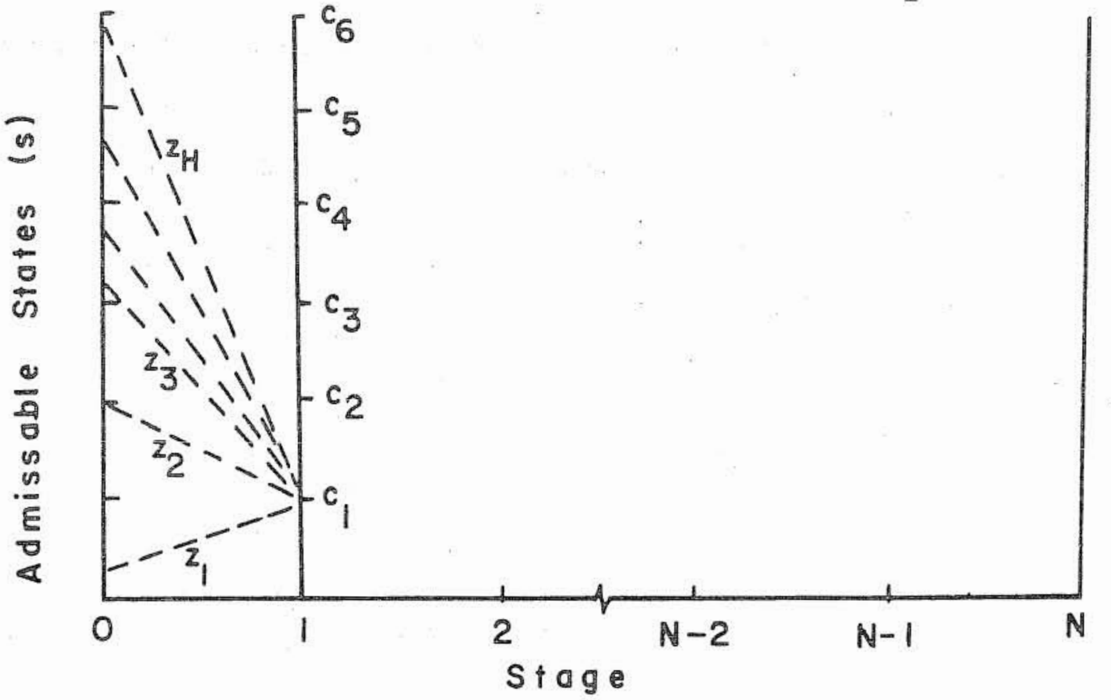


FIGURE 1. Schematic Representation of Possible Decisions which bring the State of the System to c_1 at n=1. (Forward Algorithm)

the decision vector, u(0), shown as z_1, z_2, ..., z_H, are applied to various states (discrete or nondiscrete), s(0), at n = 0 such that the state of the system is transformed to c_1 at n = 1. There may be other admissible discrete decision levels which would bring the state of the system to c_1 but the originating state, s(0), may not be a member of the admissible state domain at n = 0. Therefore, these decisions are excluded from the analysis. $\mathbf{u}^*[c_1,0]$ is then the decision which causes the maximum value of F[s(1),1] when $s(1) = c_1$.

This procedure is then repeated for L_i-1, i = 1, 2, ..., m, other discrete states at stage n = 1, and for each state, $s(1)\in S(1)$, $u^*[s(1),0]$ and $F^*[s(1),1]$ are calculated as a function of s(1). In Figure 1 the L_i-1 other states are designated by $c_2, c_3, \ldots, c_6$. Then, the optimum decision vector and return calculated for each state at stage 1 is stored either as a continuous function of s(1) or in the discrete form for further use.

Now, the calculations may be performed for stage n = 2 representing $t = t_0 + 2\Delta t$. This procedure consists of similar steps, except there are now two decision vectors which must be considered in sequence: the decision vector applied to the system in the time span between stages n = 0 and n = 1 and the decision vector applied between stages n = 1 and n = 2. Writing the maximum of equation (4) for n = 2 we have

$$F^*[s(2),2] = \max_{u(1)\in U(1)} \{R[s(1),u(1),1] + F^*[s(1),1]\} \tag{7a}$$

$$= \max_{u(1)\in U(1)} \{R\Big[\theta[s(2),u(1),1],u(1),1\Big] + F^*\Big[\theta[s(2),u(1),1],1\Big]\} \tag{7b}$$

which for every discrete level of the state vector at $n = 2$ may be solved as a function of $u(1) \in U(1)$ only. This is done exactly as above with $n = 1$ and $n = 2$ replacing $n = 0$ and $n = 1$ respectively. In an analogous manner we may continue the computation to n stages. Therefore, the general form of the equation for state $s(n)$ at stage n may be written as

$$F^{*}[s(n),n] = \max_{u(n-1) \in U(n-1)} \{R[s(n-1),u(n-1] + F^{*}[s(n-1),n-1]\} \quad (8)$$

where $F^{*}[s(n),n]$ is the optimal return over n stages leading to state $s(n)$. Equation (8) is the recursive equation or functional equation of dynamic programming.

The solution of equation (8) over N stages provides the value of the maximum return from the system. It is only at the end of the analysis that we are able to evaluate the entire process. We may now trace back from stage N to stage 0 to retrieve the optimum policy which satisfies specific initial and final states. This optimal policy is retrieved from among the optimum decision vectors previously determined for each state of each stage. The decision vectors of the optimum policy are then introduced into the system equations to obtain the related optimum states which define the optimum trajectory.

The above analysis is general and assumes that the initial and final states can be any admissible states at stages $n = 0$ and $n = N$, respectively. If we have a system with the initial and final states fixed we can still use the above analysis. For example, to obtain a trajectory which starts with state $a(0)$ and ends at state $a(N)$ we exclude all states at $n = 0$ from the admissible state domain except $a(0)$. We, then, determine our optimum policy by beginning the backward tracing at state a(N) for n = N. This allows us to determine the optimum policy for a two point boundary value problem.

The extension of equation (8) to processes with random elements is a straightforward manipulation. Let us assume that the state of the system at any stage, n, may be affected by a decision u(n-1) and a random disturbance y(n-1). Then, the dynamic equation (1) may be written as:

$$s(n) = \phi[s(n-1),u(n-1),y(n-1),n-1] \tag{9}$$

where y(n-1) is the discrete value of the random disturbance affecting the system in the time increment starting at n-1. The introduction of y(n-1) into equation (1) transforms s(n) into a random variable. Therefore, in evaluating the criterion of equation (4) we should search for its mathematical expectation. The new objective criterion may be presented by:

$$\text{Maximize } F[s(N),N] = E\{\sum_{n=1}^{N} R[s(n-1),u(n-1),y(n-1),n-1]\} \tag{10}$$

where E{·} denotes the expected value of the terms in the bracket. Assuming that the disturbances at stage n, n-1, and n+1 are independent from each other, and the probability density function for y(0),y(1), ..., y(N-1) are known for V discrete levels in the range of $-\infty$ to $+\infty$, the recursive equation, equation (8), for a discrete level of state at stage n with random disturbances may be written as:

$$F^*[s(n),n] = \max_{u(n-1)\in U(n-1)} \Big\{E\{R[s(n-1),u(n-1),n-1] + F^*[s(n-1),n-1]\}\Big\} \tag{11a}$$

$$= \max_{u(n-1)\in U(n-1)} \Big\{\sum_{v=1}^{V} P[y(n-1),v] \cdot \{R\big[s(n-1),u(n-1),[y(n-1),v],n-1\big] + F^*[s(n-1),n-1]\}\Big\} \tag{11b}$$

where $P[y(n-1), v]$ is the probability of the v-th discrete value of the random disturbance y affecting the system in the time interval starting at n-1.

For stochastic disturbances, the conditional probability density of disturbances must replace the independent probability density in equation (11b), and the summation must be performed twice or perhaps more.

In the application of dynamic programming to water resources problems, it seems that the forward algorithm is more relevant than the backward algorithm. In these systems, the events of the past, such as recharge or irrigation activities, usually influence the state of the system in the current time or future. For example, a portion of a recharge into an aquifer which took place in the time span between stages n-2 and n-1 may affect the state of the system during the time between stages n and n+1. Therefore, a knowledge of the past activities must be available for the evaluation of the present state of the system. This can easily be achieved by a forward algorithm in which activities and decisions of the past are known or already made.

2-1-3. Applications of Dynamic Programming

Due to its flexibility and simplicity, discrete dynamic programming has been used to investigate a variety of water resources problems. Among these problems the following may be mentioned: aqueduct planning, storage design and operation of multiple-purpose reservoirs, branching multiple-stage water resources systems, conjunctive operation of dams and aquifers, and water quality studies. (See Chow and Meredith [1969a, b] for a more complete list of applications of the dynamic programming approach to water resources systems analysis.)

2-2. Advantages of Dynamic Programming

The major advantages for using the dynamic programming approach in water resources systems analysis are summarized below.

2-2-1. Analysis of Multiple-Stage Processes

Dynamic programming is especially suitable for the analysis of multiple-stage processes and most water resources systems can be viewed as multiple-stage processes.

2-2-2. Incorporation of Stochastic Disturbances

One of the advantages of dynamic programming is that it offers a way of solving deterministic and stochastic problems without significant changes in the algorithm. This universality is particularly important in setting up large problems where the deterministic algorithm with slight modification can be applied to a problem with stochastic disturbances. The derivation of the probabilistic recursive equation from a deterministic one has been demonstrated in Section 2-1-2. Furthermore, it was mentioned that this derivation may be extended to problems which contain conditional probability distributions. This extension is particularly applicable to water resources systems where the probability distribution of runoffs during two consecutive time periods may best be represented by a joint probability distribution from which the conditional probability distribution of each event may be calculated.

It should be noted, that the incorporation of stochastic disturbances in other optimization models is either impractical or substantial modifications must be made in the deterministic model. A recent study by Gablinger and Loucks [1970] indicates that for a single-state water system a stochastic linear programming problem requires 20 times as much computer time as a stochastic dynamic programming problem (2 hours vs. less than 5 minutes). However, the authors concluded that the cost of developing a dynamic programming computer program as opposed to the already available linear programming routines, such as IBM MPS, should be considered in such comparison.

2-2-3. Incorporation of Constraints

The solution of dynamic programming problems is usually presented in numerical form. Due to the fact that at every stage the range and quantized levels of states and decisions may easily be predetermined or a test may be performed to see if a constraint is violated, handling constraints raises no mathematical or computational difficulties.

The treatment of constraints in other optimization techniques is by no means trivial. In some techniques a penalty function is introduced in the objective criterion of equation (4). The Lagrange multiplier, λ, is frequently used to append the constraints and, thus, create an equivalent optimization problem without constraints which can be solved directly as a function of λ. Eveleigh [1967] presents the theoretical foundation of this approach based on Lagrange's work. He also presents techniques for handling equality and inequality constraints in the steepest ascent technique.

2-2-4. Incorporation of Nonlinearity

In the development of the recursive equation, equation (8), there was no mention of the nature of the objective function. This is because its nature is of no consequence to this development. Therefore, this equation may be used for linear as well as non-linear objective functions. Since in linear problems the optimum values lie at the extreme points of the convex policy set, the knowledge of linearity may be effeciently used in dynamic programming to search only for extremities of linear constraints.

2-3. Disadvantages of Dynamic Programming

The disadvantages of the recursive equation, equation (8), are discussed below.

2-3-1. Dimensionality

The dimensionality requirement of dynamic programming is the amount of high-speed computer storage memory which is required to solve the recursive equation, equation (8), for a state vector of m dimensions and a decision vector of q dimensions. In order to perform the calculations required for this equation at stage n one must have at least ready access to storage locations associated with the following terms:

$F^*[s(n),n]$	for all lattice points $s(n)\in S(n)$;
$F^*[s(n-1),n-1]$	for all lattice points $s(n-1)\in S(n-1)$; and
$u^*[s(n-1),n-1]$	for all lattice points $s(n-1)\in S(n-1)$.

Assuming that s(n) and s(n-1) are quantized with L_i quantization levels in coordinate i, i = 1, 2, ..., m, then the total storage, L_T, required for the above terms is

$$L_T = 3 \prod_{i=1}^{m} L_i \qquad (12)$$

This storage requirement grows geometrically with the dimension of the state domain, m, and the quantized levels of states, L_i. For example, for a water resources system consisting of four reservoirs (four state variables, i.e., m = 4), if each state is quantized into only 15 quantized levels (i.e., $L_i = 15$, i = 1,2,3,4) then at least a total of 151,875, $(3(15)^4)$, highspeed storage memory units is required. It should be mentioned that for most reservoirs dividing the active storage into only 15 levels will not give conclusive results. Even if we could satisfy ourselves with such unreliable results, the required amount of storage is beyond the capacity of available computers.

2-3-2. Computer Time Requirements

This disadvantage may even be more severe than the previous one. Computer time requirements for a realistic high-dimensional system can put a severe restraint on the budget allocated for system analysis.

The savings gained by dynamic programming as compared to direct enumeration increases as the number of stages increases. Bellman and Dreyfus [1962] demonstrate this point by the following example. Consider an N stage process with q decision variables and m state variables per stage. If each state and decision variable is divided into 10 quantized levels, i.e., $L_i = H_j = 10$ for $i = 1, \ldots, m$ and $j = 1, \ldots, q$, then by direct enumeration the objective criterion must be evaluated $10^{n \cdot q+m}$ times. The same problem by dynamic programming requires the evaluation of the objective criterion $N \cdot 10^{q+m}$ times, which is by a factor of $\frac{10^{(N-1)q}}{N}$ more efficient than direct enumeration. It should be noticed that as N increases the value of the above factor increases more rapidly than N.

The comparison of the efficiency of dynamic programming with direct enumeration, although rather impressive, does not tell the full story. The optimization by traditional dynamic programming can indeed be computer time consuming and expensive. Assume that in the above example $N = 100$, and $q = m = 4$. Since for each lattice point four operations must take place in equation (8) for every set of decisions, namely; calculation of the first term on the right-hand side, retrieval or interpolation to obtain the second term on the right-hand side, addition of these two terms, and comparison of the sum with the results of the previous set of decisions, a total of $100 \cdot 4 \cdot 10^8$ operations must be performed by dynamic programming. At the rate of about 10^6 operations per second for the IBM 360/75, this optimization would require about 11.2 hours of computer time. The optimization by direct enumeration would require an astronomical amount of time. However, 11.2 hours of time on the IBM 360/75 is not considered feasible for many projects.

2-4. Techniques to Reduce the Dimensionality of Dynamic Programming

The prohibitive dimensionality of dynamic programming for multiple-dimensional systems with small grid sizes has been discussed previously. Some of the attempts to reduce the dimensionality of dynamic programming are reviewed below. In addition to the techniques reviewed here Bellman and Kalaba [1961], Roberts [1964], Wong and Luenberger [1968], Lee [1969], and Hall et al. [1969] have suggested other techniques.

2-4-1. Successive Approximation

Larson [1968] has presented an algorithm for Bellman's [1961] successive approximation, which has been shown to be very efficient both in high-speed memory and in computer time requirements. The basic approach in this algorithm is the decomposition of a problem with an m-dimensional state vector and a q-dimensional decision vector into a series of problems with only one decision variable. Larson found that this approach works best when $m = q$.

Assume a discrete system such as the one described by equations (1) and (8). The optimization starts with the selection of any trial trajectory which satisfies all constraints imposed upon the system. Since the order of the system is m, the order of the trial trajectory will be m. Now, one of the members of the state vector is selected and, while the other m-1 state variables in the state vector are kept constant, it is allowed to vary within its admissible range. This constraint on m-1 state variables imposes an equality constraint on m-1 decision variables and only one decision variable will be subject to an inequality constraint. Having reduced the problem to a one-dimensional problem, a regular one-dimensional dynamic programming problem is solved for the performance criterion with the selected state variable and its respective decision variable as variables, and the rest of the state variables are kept equal to their trial values. This optimization leads to an improvement of the trajectory of the selected state variable. Then, another state variable is selected and the above steps are repeated. The selection of the state variables continues until no more improvement can be made as a result of optimization.

The structure of the above technique indicates a linear increase of the storage and computer time requirements with an increase in m, the number of components in the state space. For a regular dynamic programming problem the storage and computer time requirements increase exponentially.

Korsak and Larson [1970] have given convergence proofs of this technique for three types of problems: (1) problems in which the states are bounded, the decisions are unbounded, and the elimination of decisions (using the invertibility of the equations) produces an objective function which is convex and a function of the states; (2) problems in which the decisions are bounded, the states are unbounded, and the elimination of states results in a convex objective function of decisions (and initial state); and (3) problems in which both the states and the decisions are bounded, and the objective function is quadratic.

The convergence to the global maximum for other types of problems cannot be guaranteed by this technique. Larson [1968] proposed several techniques for arriving at a set of trial trajectories which are close to the true optimal trajectories. The use of these trajectories will improve the chances of obtaining the global optimum.

Using this technique, a problem with four states and four decision variables is solved. This problem is presented in Chapter 4.

2-4-2. State Increment Dynamic Programming

Larson [1968] has proposed state increment dynamic programming as a means of reducing the amount of computer storage memory required because of the dimensionality of dynamic programming. This technique is based on determining increments of time δt at stage n, which, when used to evaluate the effect of decisions on the states, the transformed state at stage n + 1 will not be more than one lattice point above or below the state at stage n. This may be formulated as follows:

Given a continuous system with a set of non-linear time-varying differential equations:

$$\dot{s} = \phi[s,u,t] \tag{13}$$

where s is an m-dimensional state vector; u is a q-dimensional decision vector; t is a continuous variable denoting time; ϕ is an m-dimensional vector functional; and $\dot{s} = \frac{ds}{dt}$. Let it be required to maximize a performance criterion such as

$$F[s(t_f),t_f] = \{\int_{t_o}^{t_f} R[s(\eta),u(\eta),\eta]d\,\eta\} \tag{14}$$

where $F[s(t_f),t_f]$ is the sum of the returns due to transforming the system from some initial state at time t_0 to some final state at time t_f;

t_0 is the initial time; t_f is the final time; R is the return function per unit time; and η is a dummy variable representing time. Equation (14) is subject to:

$$s(t) \in S(t) \tag{15a}$$

$$u(t) \in U(t) \tag{15b}$$

$$t \in [t_0, t_f] \tag{15c}$$

where S(t) is the admissible state domain at time t; and U(t) is the admissible decision domain at time t. Note that this is the continuous form of the problem defined previously. Thus the solution to it is a continuous decision vector $u^*(t)$, $t \in [t_0, t_f]$, and is called the optimal policy.

In state increment dynamic programming it is assumed that u(t), $t \in [t_0, t_f]$, is piecewise constant over an increment of time δt. Therefore, equation (13) may be approximated by:

$$s(t+\delta t) = s(t) + \phi[s(t), u(t), t]\delta t \tag{16}$$

The performance criterion, equation (14), for time interval t to $t + \delta t$ may be written as:

$$F[s(t), t] = R[s(t), u(t), t]\delta t \tag{17}$$

Now if we were dealing with traditional dynamic programming, δt would remain constant over the entire time horizon, i.e., the time horizon would be divided into N time intervals of Δt, $(\Delta t = \delta t)$. In state increment dynamic programming these two time intervals are calculated separately. Δt is defined as in tradiational dynamic programming. But δt is the interval of time less than or equal to Δt which any of the m state variables requires to change at most by one increment Δs (interval between two adjacent state lattice points), with each decision applied. In effect this technique states that, instead of applying a decision over time increment Δt, it should be applied long enough, δt, to transform the state of the system by at most Δs.

To demonstrate this, let us assume that a system with $m = 1$ and $q = 1$ at time $t-\Delta t$ and t can be at states c_1, c_2, c_3 and e_1, e_2, e_3 respectively. If the admissible decision domain is divided into 4 levels, $z_1, \ldots, z_4$, and if we want to reach state e_2 at time t, then the application of the 4 levels of decision to the system may be as shown in Figure 2. In this figure application of each decision is continued long enough to create a change of one Δs or less.

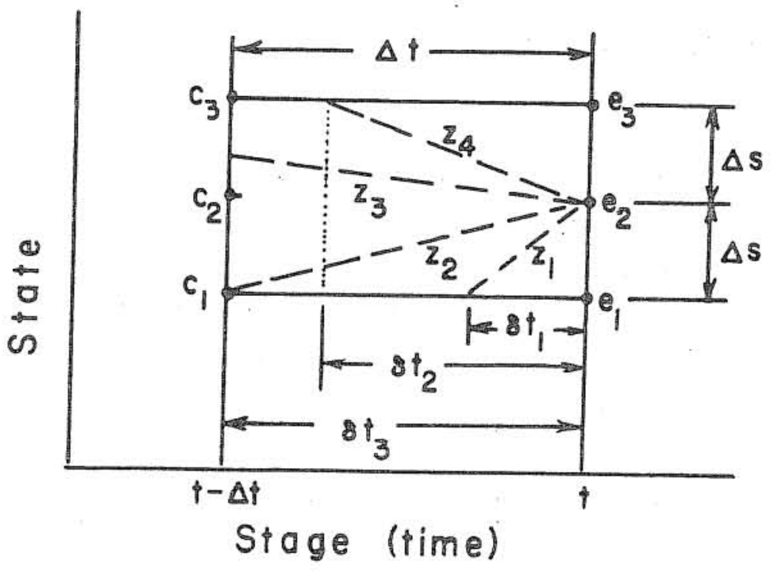


FIGURE 2. Determination of δt for State Increment Dynamic Programming (Forward Algorithm)

Therefore, decisions z_1 and z_4 are continued for time increments δt_1 and δt_2 respectively, and decision z_2 and z_3 are continued for a time increment $\delta t_3 = \Delta t$.

Writing equation (8) for state e_2 we have:

$$F^*[e_2,t] = \max_j \{R[e_2,z_j] + F^*[s_{t-\Delta t},t-\Delta t]\} \qquad (18)$$

where j is the index of z_j, j = 1, 2, 3, 4; and $s_{t-\Delta t}$ is any one of the possible states at t-Δt. Assuming that $F^*[s_{t-\Delta t},t-\Delta t]$ for three possible states at t-Δt have been calculated in the previous optimization stage, application of z_j, j = 1, 2, 3, 4 will provide the first term on the right side of equation (18), i.e., $R[e_2,z_j]$ and perhaps the second term on the right side; namely, $F^*[s_{t-\Delta t}, t-\Delta t]$. For the decisions shown in Figure 2 application of z_2 provides both terms on the right-hand side of equation (18), but the application of z_1, z_3, or z_4 provides $R[e_2,z_j]$ only. Therefore, the second term, $F^*[s'_{t-\Delta t},t-\Delta t]$, (where the prime sign indicates that the state is not one of the quantized states) has to be obtained by interpolation among the neighboring quantized states at t-Δt and t-2Δt.

This is the basic idea behind the state increment dynamic programming. The advantage of it is that at every state, such as e_2, only the maximum returns of the three discrete states at t-Δt, and perhaps t-2Δt have to be stored. This reduces the computer storage requirements for dynamic programming to a point that high-dimensional systems can be handled. Larson [1968] has presented an algorithm of this technique which can handle up to four state variables. He concludes that it reduces the storage requirements of a 4-dimensional state vector with 100 quantized levels in each state from 10^6 words to 799 words and reduces the computer time required substantially.

Perhaps it should be mentioned that the accuracy of the interpolation for a multiple-dimension state variable cannot be guaranteed. Also the time required for such interpolations can be substantial.

Hall *et al*. [1969] have applied the concept of incremental dynamic programming to water resources problems. This approach is based on the suggestion by Larson [1968] for state increment dynamic programming. However, the theoretical anlaysis for state incremental dynamic programming was not presented and the effectiveness of the conergency of this approach was not satisfactorily demonstrated.

2-4-3. Differential Dynamic Programming

Differential dynamic programming is a successive approximation technique for determining the optimal policy for nonlinear systems. This technique reduces the computer storage requirement such that large systems can be analyzed on available equipment (IBM 360/75). Therefore, the technique was chosen for application to water resources systems analysis and is presented in Chapter 3.

3. A DISCRETE DIFFERENTIAL DYNAMIC PROGRAMMING (DDDP) APPROACH FOR WATER RESOURCES SYSTEMS ANALYSIS

3-1. Differential Dynamic Programming

Differential dynamic programming is a successive approximation technique for determining the optimal policy for nonlinear systems. It was first introduced into optimal control theory by Mayne [1966] and was, then further developed by Jacobson [1968a, b, c] and Jacobson and Mayne [1970]. The approach presented in this chapter is a discrete differential dynamic programming (DDDP) approach for water resources systems analysis. It is the extension of the work of Mayne and Jacobson.

3-1-1. Differential Dynamic Programming-Theory

Fel'dbaum [1965] presents a formal derivation of Bellman's partial differential equation from the principal of optimality for continuous systems. The equation is a first order non-linear partial differential equation of the form

$$-\frac{\partial F^*}{\partial t}[s(t),t] = \max_{u(t)\in U(t)} \{R[s(t),t] + \langle F_s^*[s(t),t],\phi[s(t),u(t),t]\rangle\} \qquad (19)$$

where $F_s^*[s(t),t]$ is the gradient of the function $F^*[s(t),t]$ and is equal to

$(\frac{\partial F^*}{\partial s_1}, \frac{\partial F^*}{\partial s_2}, \ldots, \frac{\partial F^*}{\partial s_m})$ and the notation $\langle , \rangle$ signifies the scalar product of two vectors which have m components. For example:

$$\langle Y,Z\rangle = \sum_{i=1}^{m} y_i z_i \qquad (20)$$

Equation (19) may be solved with a boundary condition to determine $F^*[s(t),t]$ for all $t\in[t_0,t_f]$ [Bryson and Ho, 1969].

Differential dynamic programming is a technique based on equation (19) in which the return function is not considered globally.

Consider the continuous system presented in equation (13). Let it be required to maximize the performance criterion of equation (14) subject to equations (15a,b,c). In the development of equation (19) the major assumption is that the optimal return function F^* has continuous partial derivatives with respect to all state variables. Now, let us assume a trial policy $\overline{u}(t)$, $t\in[t_0,t_f]$ which satisfies equation (15b). Introduction of this policy into equation (13) will provide a trial trajectory $\overline{s}(t)$, $t\in[t_0,t_f]$, which either must satisfy equation (15a) or $\overline{u}(t)$ must be changed. The return from the system, due to $\overline{u}(t)$ and $\overline{s}(t)$, calculated by equation (14) and denoted by $\overline{F}(\overline{s},t_0)$ may not be the optimal return. When a given policy such as $\overline{u}(t)$, is applied, the system can only occupy the states defined by the trajectory $\overline{s}(t)$. If we now permit the policy to vary by $\delta u(t)$, $t\in[t_0,t_f]$, then a new policy given by

$$u(t) = \overline{u}(t) + \delta u(t) \qquad (21)$$

will influence the trajectory through equation (13). The new trajectory will be given by:

$$s(t) = \overline{s}(t) + \delta s(t) \qquad (22)$$

where $\delta u(t)$ and $\delta s(t)$ are changes in the trial policy and the trial trajectory respectively for $t\in[t_0,t_f]$. Introduction of equation (21) and (22) into equations (13), (14), and (19) will produce a set of equations such as:

$$\frac{d}{dt}\ (\overline{s}+ds) = \phi[\overline{s}+\delta s, \overline{u}+\delta u, t] \tag{23}$$

$$\text{Maximize } \overline{F}[s_f, t_f] = \int_{t_0}^{t_f} R[\overline{s}+\delta s, \overline{u}+\delta u, t]dt \tag{24}$$

and

$$-\frac{\partial F^*}{\partial t}\ [\overline{s}+\delta s, t] = \max_{\delta u}\ \{R[\overline{s}+\delta s, \overline{u}+\delta u, t]$$

$$+ \left\langle F^*_s\ [\overline{s}+\delta s, t],\ \phi[\overline{s}+\delta s, \overline{u}+\delta u, t] \right\rangle \tag{25}$$

It should be noted that as yet no restrictions have been put on the magnitudes of δs and δu.

$F^*[\overline{s}+\delta s, t]$ may be expressed by a power series expansion with respect to $\overline{s}$ as follows:

$$F^*[\overline{s}+\delta s, t] = F^*[\overline{s}, t] + \left\langle F^*_s\ [\overline{s}, t], \delta s \right\rangle$$

$$+ \frac{1}{2}\left\langle \delta s, F^*_{ss}\ [\overline{s}, t]\delta s \right\rangle + \text{h.o.t.} \tag{26}$$

$$F^*[\overline{s}, t] = \overline{F}[\overline{s}, t] + c^*[\overline{s}, t] \tag{27}$$

where $F^*[\overline{s}, t]$ is the maximum return due to the optimal trajectory measured with respect to the trial trajectory from time t_0 to t_f; $\overline{F}[\overline{s}, t]$ is the return due to the trail trajectory $\overline{s}$ from time t_0 to t_f; $c^*[\overline{s}, t]$ is the difference between the maximum return due to the optimal trajectory $s^*(t)$ and the trial trajectory; $F^*_s\ [\overline{s}, t]$ is an m-dimensional column vector equal to $(\frac{\partial F^*}{\partial s_i}$, $i = 1, \ldots, m)$ evaluated at $\overline{s}(t)$; $F^*_{ss}[\overline{s}, t]$ is an m x m dimensional matrix equal to $(\frac{\partial^2 F^*}{\partial s_i \partial s_j}$, i, j = 1, 2, ..., m) evaluated at $\bar{s}(t)$; $F^*_{ss}[\bar{s},t]\delta s$ is an m-dimensional vector; and h.o.t. stands for higher order terms. Similarly $F^*_s[\bar{s}+\delta s,t]$ may be expressed by

$$F^*_s[\bar{s}+\delta s,t] = F^*_s[\bar{s},t] + F^*_{ss}[\bar{s},t]\delta s + \text{h.o.t.} \qquad (28)$$

Substituting equations (26), (27), and (28) into equation (25) and, for the sake of compactness, dropping $[\bar{s},t]$ wherever possible

$$-\frac{\partial \bar{F}}{\partial t} - \frac{\partial c^*}{\partial t} - \left\langle \frac{\partial F^*_s}{\partial t}, \delta s \right\rangle - \frac{1}{2}\left\langle \delta s, \left(\frac{\partial F^*_{ss}}{\partial t}\right)\delta s \right\rangle - \text{h.o.t.}$$

$$= \max_{\delta u}\{R[\bar{s}+\delta s,\bar{u}+\delta u,t] + \langle (F^*_s + F^*_{ss}\delta s + \text{h.o.t.}), \phi[\bar{s}+\delta s,\bar{u}+\delta u,t]\rangle\} \qquad (29)$$

The solution of equation (29), if possible, will provide $\delta u^*(t)$, $t\in[t_0,t_f]$, which is the amount that the trial policy must be incremented at time t to obtain the optimum policy. Thus,

$$u^*(t) = \bar{u}(t) + \delta u^*(t), \text{ for } t\in[t_0,t_f] \qquad (30)$$

where $u^*(t)$ is the optimum policy. But, the solution of equation (29) requires "possibly infinite computing time and storage requirements for the parameters of the power-series expansion" [Jacobson, 1968b].

In order to make the solution of equation (28) possible, Jacobson [1968b] proposes truncation of the higher order terms. This proposal can only be justified if δs is small enough to make these terms negligible. Assuming that δs is kept small enough so that the highest order terms in equation (29) are quadratic:

$$-\frac{\partial\overline{F}}{\partial t}-\frac{\partial c}{\partial t}-\left\langle\frac{\partial F_s}{\partial t},\delta s\right\rangle-\frac{1}{2}\left\langle\delta s,\left(\frac{\partial F_{ss}}{\partial t}\right)\delta s\right\rangle$$

$$=\max_{\delta u}\{R[\overline{s}+\delta s,\overline{u}+\delta u,t]+\langle(F_s+F_{ss}\delta s),\phi[\overline{s}+\delta s,\overline{u}+\delta u,t]\rangle\} \tag{31}$$

where

$$F_s=\frac{\partial\overline{F}}{\partial s}[\overline{s}+\delta s,t]=F_s[\overline{s},t]+\overline{F}_{ss}\delta s \tag{32}$$

and

$$F[\overline{s}+\delta s,t]=\overline{F}[\overline{s},t]+c+\langle F_s,\delta s\rangle+\frac{1}{2}\langle\delta s,F_{ss}\delta s\rangle \tag{33}$$

This process reduces the global optimization of equation (29) to a local optimization, i.e., optimization takes place in the neighborhood of the trial trajectory. Therefore, solution of equation (31) is only an improvement over that of the trial trajectory and not an optimum one. It is because of this that F has replaced F^* in equation (31). However, if the improved trajectory is optimized again in its neighborhood, it may provide a still better trajectory. Continuing in this manner, the trajectory gradually converges to the optimal trajectory.

Based on equation (31) Jacobson [1968a] has set up algorithms for second order and first order unconstrained and inequality constrained problems. The algorithm optimizes the Hamiltonian in the neighborhood of the tiral trajectory. It contains a step to calculate a reasonable δs which may improve the rate of convergence.

This technique reduces the "almost infinite" storage requirement of equation (31) to a level which can be handled with available equipment. This is done by limiting the optimization process to the neighborhood of the trial trajectory. For systems in which only certain trajectories are to be investigated, this technique can be quite time saving from the computational point of view.

3-1-2. Discrete Differential Dynamic Programming

Let us assume that the objective function, equation (4), for the system of equation (1) is to be optimized subject to equation (2) and that the m-dimensional state vectors at the initial and final stages are specified such that

$$\begin{aligned} s(0) &= a(0) \\ s(N) &= a(N) \end{aligned} \tag{34}$$

In the proposed DDDP approach a trial sequence of admissible decision vectors, $\overline{u}(n)$, $n = 0, 1, \ldots, N-1$, called the trial policy, satisfying equation (2) is assumed and the state vectors at different stages are determined. The sequence of values of the state vector satisfying equations (2) and (34) is called the trial trajectory and is designated by $\overline{s}(n)$, $n = 0, 1, \ldots, N$. For invertible systems which will be defined later, it is possible to first assume an admissible trial trajectory, $\overline{s}(n)$, $n = 0, 1, \ldots, N$, and then use it to calculate the trial policy $\overline{u}(n)$, $n = 0, 1, \ldots, N-1$.

Introducing $\overline{u}(n)$ and $\overline{s}(n)$ into equation (4) we obtain

$$\overline{F} = \sum_{n=1}^{N} R[\overline{s}(n-1), \overline{u}(n-1), n-1] \tag{35}$$

where $\overline{F}$ is the total return due to the trial trajectory and trial policy over the entire time horizon. $\overline{F}$ may not be the optimum reutrn.

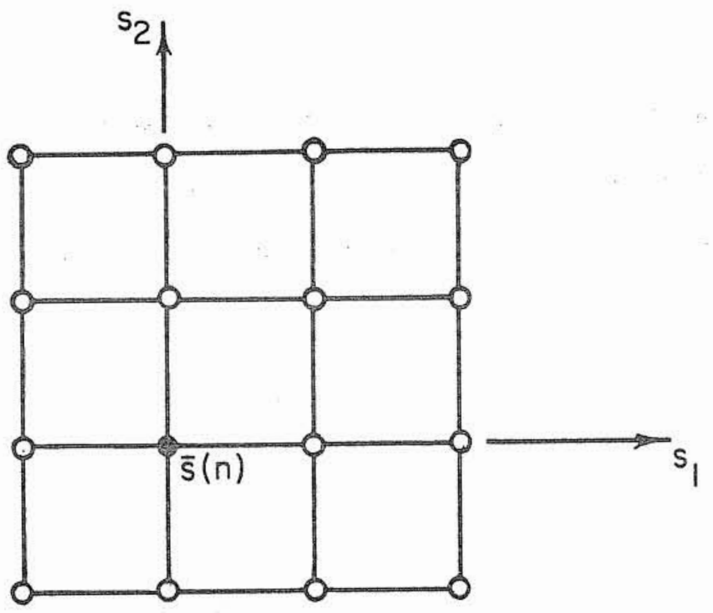
Now, consider a set of incremental m-dimensional vectors

$$\Delta s_i(n) = \begin{bmatrix} \delta s_{i,1}(n) \\ \delta s_{i,2}(n) \\ \vdots \\ \delta s_{i,j}(n) \\ \vdots \\ \delta s_{i,m}(n) \end{bmatrix} \qquad \begin{array}{l} n = 0, 1, \ldots, N \\ i = 1, 2, \ldots, T^m \end{array} \tag{36}$$

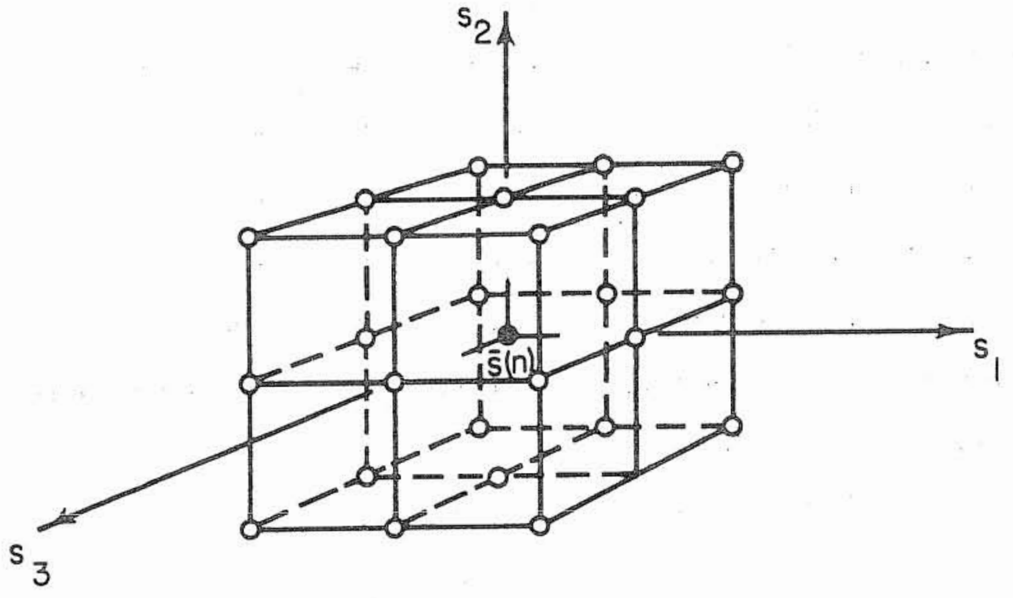
whose j-th component $\delta s_{i,j}(n)$, $j = 1, 2, \ldots, m$, can take any one value $\sigma_{j,t}$, $t = 1, 2, \ldots, T$, from a set of assumed incremental values of the j-th state domain. Thus, assuming that from each state domain a fixed number, T, of incremental values are considered at each stage, the total number of Δs_i vectors at that particular stage is T^m. When added to the trial trajectory at a stage, these vectors form an m-dimensional sub-domain designated by D(n),

$$\overline{s}(n) + \Delta s_i(n) \qquad i = 1, 2, \ldots, T^m \tag{37}$$

It should be noted that one value of $\sigma_{j,t}$ must be zero since the trial trajectory is always in the sub-domain. In Figure 3 two such sub-domains for $m = 2$, $T = 4$ and $m = 3$, $T = 3$ are presented. All D(n), $n = 0, 1, \ldots, N$ together are called a "corridor" and designated by C as shown in Figure 4 by the space between two solid lines for a system with $m = 1$, $T = 3$, and $N = 10$.



A state sub-domain D(n) defined by 16 lattice points in the neighborhood of $\bar{s}(n)$ for a 2-dimensional state vector and T = 4($\sigma_{j,1}$ = +2.0, $\sigma_{j,2}$ = +1.0, $\sigma_{j,3}$ = 0, $\sigma_{j,4}$ = -1.0 for j = 1,2)



A state sub-domain D(n) defined by 27 lattice points in the neighborhood of $\bar{s}(n)$ for a 3-dimensional state vector and T = 3($\sigma_{j,1}$ = +1.0, $\sigma_{j,2}$ = 0, $\sigma_{j,3}$ = -1.0 for j = 1,2,3)

FIGURE 3. Examples of State Sub-domains at Stage n

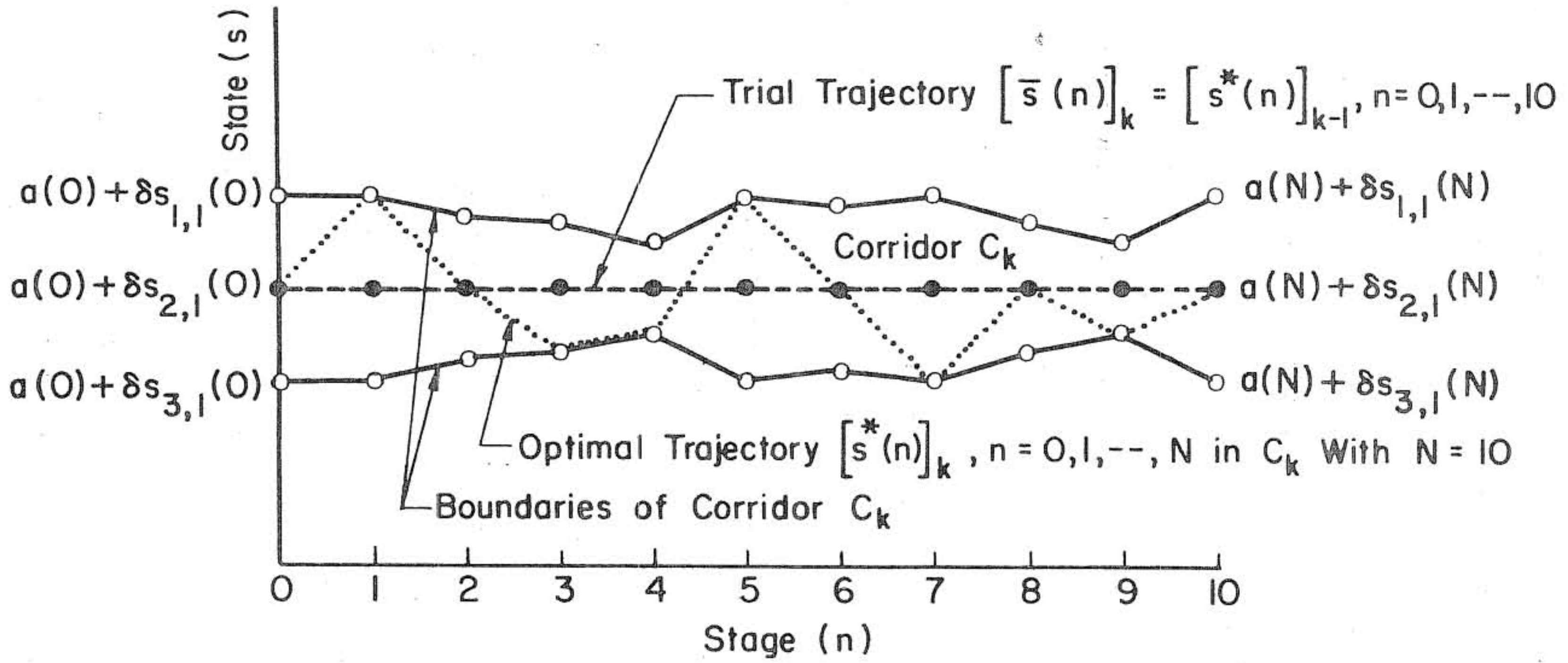

State (s)
Trial Trajectory $\left[\bar{s}(n)\right]_k = \left[s^*(n)\right]_{k-1}$, n = 0,1,--,10
$a(0)+\delta s_{1,1}(0)$
$a(0)+\delta s_{2,1}(0)$
$a(0)+\delta s_{3,1}(0)$
Corridor C_k
$a(N)+\delta s_{1,1}(N)$
$a(N)+\delta s_{2,1}(N)$
$a(N)+\delta s_{3,1}(N)$
Optimal Trajectory $\left[s^*(n)\right]_k$, n = 0,1,--, N in C_k With N = 10
Boundaries of Corridor C_k
0
1
2
3
4
5
6
7
8
9
10
Stage (n)

FIGURE 4. Schematic Representation of a Trial Trajectory, the Boundaries Defining Corridor C_k, and Optimal Trajectory in C_k of k-th Iteration for a System with m = 1 and T = 3

In DDDP a corridor C is used as a set of admissible states and the optimization constrained to these states is performed employing the recursive relation, equation (8). The value of return F obtained is at least equal to or greater than $\overline{F}$ in equation (35). If F is greater than $\overline{F}$, the corresponding trajectory and policy obtained from corridor C are used in the next iteration step as the trial trajectory and trial policy. Thus the k-th iteration step is as follows:

1. Use the results $[s^*(n)]_{k-1}$ and $[u^*(n)]_{k-1}$ of the (k-1)-st iteration step as the trial trajectory and policy for the k-th iteration step, i.e.,

$$\begin{aligned} [\overline{s}(n)]_k &= [s^*(n)]_{k-1} \\ [\overline{u}(n)]_k &= [u^*(n)]_{k-1} \end{aligned} \tag{38}$$

2. Select $[\sigma_{j,1}]_k, [\sigma_{j,2}]_k, \ldots, [\sigma_{j,T}]_k$, $j = 1, 2, \ldots, m$, to define the k-th corridor C_k, and use equation (8) to maximize F subject to

$$s(n) \in C_k \qquad n = 0, 1, \ldots, N$$

3. In corridor C_k, trace the optimum trajectory satisfying the boundary conditions of equation (34) $[s^*(n)]_k$ and the corresponding optimum policy $[u^*(n)]_k$.
4. Determine F_k^*; if $F_k^* - F_{k-1}^* < \varepsilon$ where ε is some prespecified constant, stop the iteration, otherwise go to step 1.

Figure 5 shows the flow chart of this procedure.

Since the boundary conditions of equation (34) must be satisfied, one may exclude from the analysis all the states in the sub-domain at stage

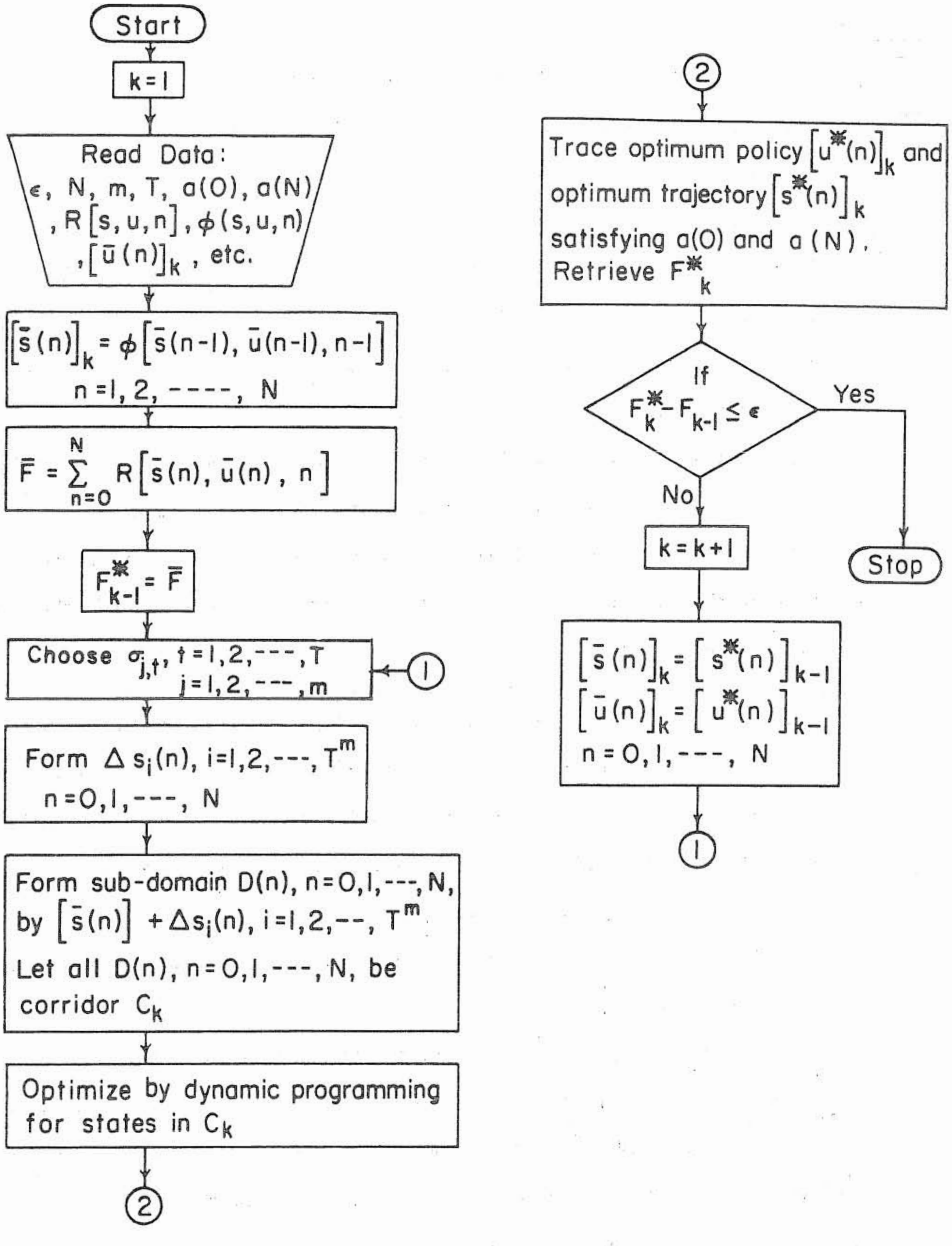

Start
$k=1$
Read Data:
ϵ, N, m, T, a(0), a(N), $R[s,u,n]$, $\phi(s,u,n)$, $[\bar{u}(n)]_k$, etc.
$[\bar{s}(n)]_k = \phi[\bar{s}(n-1), \bar{u}(n-1), n-1]$ $n=1,2,----,N$
$\bar{F} = \sum_{n=0}^{N} R[\bar{s}(n), \bar{u}(n), n]$
$F^*_{k-1} = \bar{F}$
Choose $\sigma_{j,t}$, $t=1,2,---,T$ $j=1,2,---,m$
Form $\Delta s_i(n)$, $i=1,2,---,T^m$ $n=0,1,---,N$
Form sub-domain D(n), $n=0,1,---,N$, by $[\bar{s}(n)] + \Delta s_i(n)$, $i=1,2,--,T^m$. Let all D(n), $n=0,1,---,N$, be corridor C_k
Optimize by dynamic programming for states in C_k
2
2
Trace optimum policy $[u^*(n)]_k$ and optimum trajectory $[s^*(n)]_k$ satisfying a(0) and a(N). Retrieve F^*_k
If $F^*_k - F_{k-1} \leq \epsilon$
Yes
No
Stop
$k=k+1$
$[\bar{s}(n)]_k = [s^*(n)]_{k-1}$
$[\bar{u}(n)]_k = [u^*(n)]_{k-1}$
$n=0,1,---,N$
1
1

FIGURE 5. Flow Chart Showing Steps of the DDDP Approach

n = 0 except s(0) = a(0). If in step 3 the trajectory having a final state a(N) is traced, the preservation of the boundary conditions of equation (34) is guaranteed.

Note that in the course of the iteration process, the corridor size may be varied gradually by choosing different $[\sigma_{j,t}]_k$, t = 1, 2, ..., T, j = 1, 2, ..., m, in step 2. If the corridor size is kept constant for every iteration and little or no improvement can be achieved after the k-th iteration, it is then suggested that $[\sigma_{j,t}]_k$, t = 1, 2, ..., T, j = 1, 2, ..., m, to be reduced starting at the (k+1)-st iteration and the process be continued with the new corridor size until another iteration, which behaves like the k-th iteration, is reached. Then, the corridor size is further reduced starting at the next iteration and the procedure is repeated until the condition in step 4 is satisfied.

3-2. Extension of the Proposed Approach to Systems with Stochastic Inflows

The above algorithm may be adapted to reservoir systems with random or stochastic inflows. This may be accomplished in the manner described in Section 2-1-2 which requires an interpolation technique. However, if the invertibility of the system equations, which will be described in the next chapter, is used the formulation of the probabilistic recursive equation, equation (11b), should be slightly modified.

When using the invertibility of system equations the states of the system at n-1 and n are known. Therefore, the second term on the right-hand side of equation (11b) becomes deterministic. The random disturbance influences just the first term on the right-hand side of this equation. Thus, the recursive equation with a random component becomes

$$F^*[s(n),y(n),n] =$$

$$\max_{s(n-1)\in S(n-1)} \{ \sum_{v=1}^{V} P[y(n-1),v]\cdot R\left[s(n-1),\psi[s(n-1),s(n),y(n-1),v],n-1\right] + F^*[s(n-1),n-1]\} \quad (39)$$

where P[y(n-1),v] is the v-th level of the random variable y which occurs in the span of time between stages n-1 and n; and ψ is described in Chapter 4.

For systems with stochastic disturbances the independent probability density function in equation (39) must be replaced by the conditional probability density function as stated in Section 2-1-2.

4. INVERTIBILITY OF SYSTEM EQUATIONS IN WATER RESOURCES SYSTEMS

As described previously, dynamic programming requires interpolation to retrieve the second term on the right-hand side of equation (8). For high-dimensional systems this interpolation usually produces inaccurate results and requires a large amount of computer time. It can be shown, however, that in the case of invertible systems interpolation may be avoided in the above procedure.

A system is said to be invertible if the order of the state vector is equal to the order of the decision vector, i.e., $m = q$, and the matrix $\partial\phi_i/\partial u_j$, $i, j = 1, \ldots, m$ of the system:

$$\begin{aligned} s_1(n) &= \phi_1[s(n-1), u(n-1), n-1] \\ s_2(n) &= \phi_2[s(n-1), u(n-1), n-1] \\ &\vdots \\ s_i(n) &= \phi_i[s(n-1), u(n-1), n-1] \\ &\vdots \\ s_m(n) &= \phi_m[s(n-1), u(n-1), n-1] \end{aligned} \tag{40}$$

is non-singular for every n, $u(n) \in U(n)$, and $s(n) \in S(n)$. Assuming that equation (40) is an invertible system, the decision variables can be solved in terms of the state variables:

$$\begin{aligned}
u_1(n-1) &= \psi_1[s(n),\ s(n-1),\ n-1] \\
u_2(n-1) &= \psi_2[s(n),\ s(n-1),\ n-1] \\
\vdots \quad & \quad \vdots \\
u_i(n-1) &= \psi_i[s(n),\ s(n-1),\ n-1] \\
\vdots \quad & \quad \vdots \\
u_m(n-1) &= \psi_m[s(n),\ s(n-1),\ n-1]
\end{aligned} \tag{41}$$

In the case of water resources systems it will be demonstrated that the above assumption is not restrictive. The i-th component of the vector equation (40) for a water resources system may be written as

$$s_i(n) = s_i(n-1) + y_i(n-1) - u_i(n-1) - L_i(n-1) \tag{42}$$

where $s_i(n)$ is the storage at stage n; $y_i(n-1)$ is the inflow during the time period starting at stage n-1 and lasting until stage n; $u_i(n-1)$ is release and $L_i(n-1)$ is losses due to seepage, etc. in the same time period. Since, for a system consisting of m components such as equation (42)

$$\frac{\partial \phi_i}{\partial u_j} = -1 \text{ or non-zero} \qquad i = j = 1,\ 2,\ \ldots,\ m \tag{43}$$

or $\partial\phi_i/\partial u_j$, i, j, = 1, 2, ..., m, is non-singular, the water resources system is invertible. Note that invertibility is due to the fact that in most controlled reservoir systems a release is associated with each storage unit. Ignoring the losses, $u_i(n-1)$ in equation (42) may be written in terms of inflow and states as

$$\begin{aligned}
u_i(n-1) &= s_i(n-1) - s_i(n) + y_i(n-1) \\
&= \psi_i[s_i(n-1),\ s_i(n),\ y_i(n-1)]
\end{aligned} \tag{44}$$

Assume that equation (4) is to be optimized with the forward dynamic programming algorithm for state s(n). Instead of using the state s(n) and a decision $u(n) \in U(n-1)$ in equation (3) to calculate s(n-1), one may use equation (44) to calculate the decisions that would be required for the states at stage n-1 for which $F^*[s(n-1),n-1]$ has already been calculated to go forward to state s(n). These decisions can then be tested to determine if they violate the constraints of equation (2). If the optimization is being carried out for the states in the corridor as defined for the DDDP, then the use of invertibility provides T^m possible decisions which when applied to the states in D(n-1) will bring the system to s(n). Figure 6 shows the possible decisions for a system with m = 1 and T = 3. The T^m decisions then may be used in equation (8) to determine $u^*[s(n),n-1]$ and $F^*[s(n),n]$ without interpolation to retrieve $F^*[s(n-1),n-1]$. The same procedure may be repeated for other states in the sub-domain D(n) as defined in Figure 3.

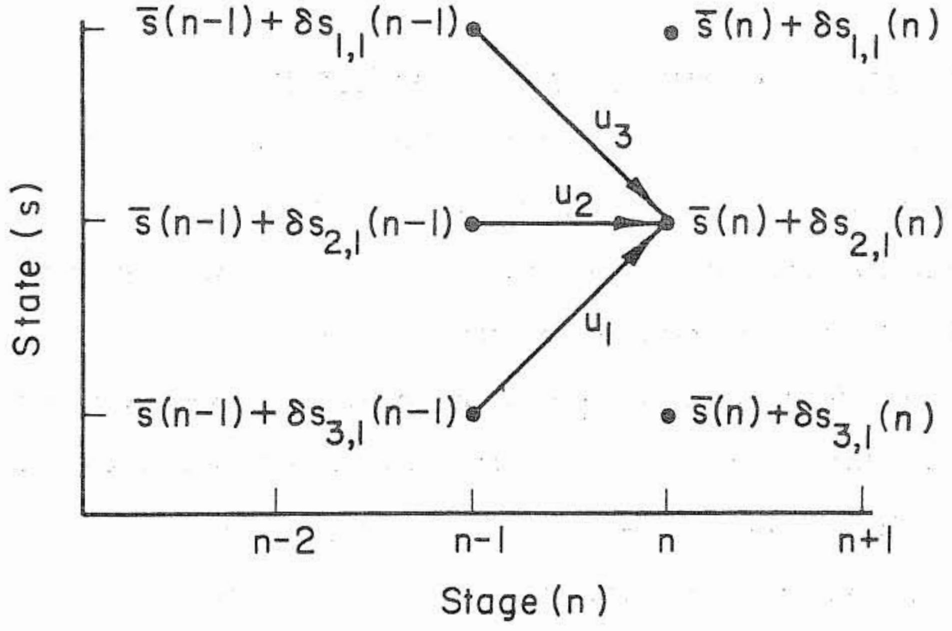
Using the invertibility of the system equations, equation (8) may be written as

$$F^*[s(n),n] = \max_{s(n-1) \in D(n-1)} \{R[s(n-1),\psi[s(n-1),s(n),y(n-1)],n-1] + F^*[s(n-1),n-1]\} \qquad (45)$$

where D(n-1) is the state sub-domain located in the neighborhood of the trial trajectory at stage n-1.

It must be emphasized that the justification of this process lies in the assumption that $\sigma_{j,t}$, t = 1, 2, ..., T, j = 1, 2, ..., m, are chosen properly. If this were not the case, most of the decisions calculated by equation (44) for state s(n) may be inadmissible. By keeping values of $\sigma_{j,t}$ within an admissible range the policy slowly converges to the optimal one in the DDDP approach.

The use of invertibility of the systems equations eliminates the inaccurate and time consuming interpolation required to retrieve the term $F^*[s(n-1),n-1]$ in equation (8). This is done by forcing the trajectories to go through the states at stage n-1 for which $F^*[s(n-1),n-1]$ has already been calculated and stored. In case of invertible multiple-dimensional systems the accuracy and speed of this procedure are much greater than the interpolation procedure.



$\bar{s}(n-1)+\delta s_{1,1}(n-1)$
$\bar{s}(n)+\delta s_{1,1}(n)$
u_3
u_2
$\bar{s}(n-1)+\delta s_{2,1}(n-1)$
$\bar{s}(n)+\delta s_{2,1}(n)$
u_1
$\bar{s}(n-1)+\delta s_{3,1}(n-1)$
$\bar{s}(n)+\delta s_{3,1}(n)$
State (s)
n-2
n-1
n
n+1
Stage (n)

FIGURE 6. Possible Decision Paths Leading to State $s(n) + \delta s_{2,1}(n)$ from Stage n-1 for a System with m = 1 and T = 3

5. APPLICATION OF THE DISCRETE DIFFERENTIAL DYNAMIC PROGRAMMING (DDDP) APPROACH

In this chapter the DDDP approach is applied to two water resources systems (a simplied water resources system and Clearwater River System) and its advantages and disadvantages are evaluated. These systems are analyzed deterministically to avoid the high computer time demand which is beyond the means of this study.

5-1. A Simplified Water Resources System

The following simplified system which was formulated and solved by Larson [1968] using linear programming and successive approximation dynamic programming was solved using the proposed approach.

5-1-1. The Problem and Its Solution by Other Techniques

The operating policy of the four-dimensional (m=4) reservoir network presented in Figure 7 is to be optimized over 12 operating periods (N=12). The inflows into reservoirs 1 and 2 during any operating period are y_1 and y_2 respectively. The outflows or releases (decisions), $u_i(n)$ $i = 1, 2, 3, 4$, $n = 0, 1, \ldots, 11$ from the reservoirs are used to generate hydropower, and $u_4(n)$ after passing through the turbines is diverted towards an irrigation project. The storages of the four reservoirs represent a four-dimensional state vector whose constraints during any operating period were set as:

$$\left.\begin{array}{l} 0 \leq s_1(n) \leq 10 \\ 0 \leq s_2(n) \leq 10 \\ 0 \leq s_3(n) \leq 10 \\ 0 \leq s_4(n) \leq 15 \end{array}\right. \quad \text{for } n = 0, 1, \ldots, 12 \qquad (46)$$

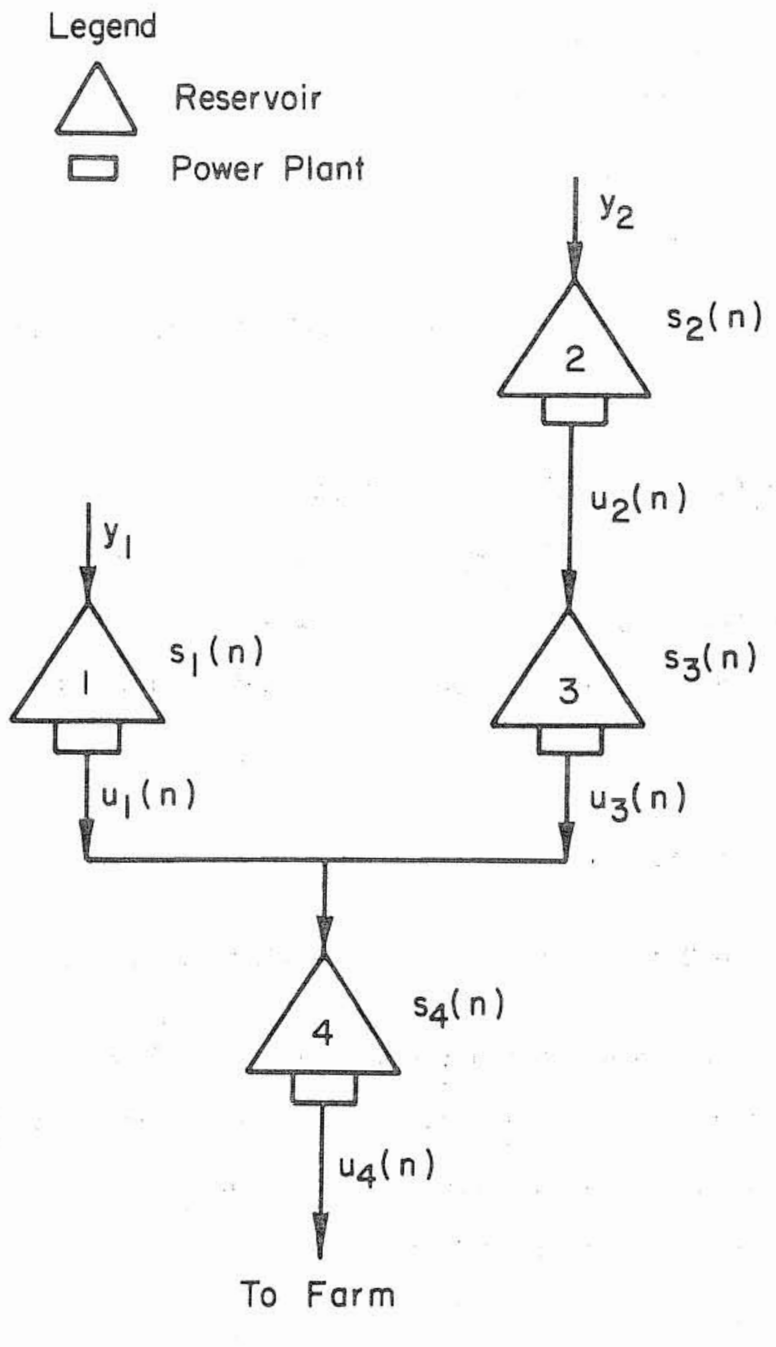


FIGURE 7. Reservoir Network of the Simplified System

The constraints on decisions during any operating period are:

$$\begin{aligned} 0 &\le u_1(n) \le 3 \\ 0 &\le u_2(n) \le 4 \\ 0 &\le u_3(n) \le 4 \\ 0 &\le u_4(n) \le 7 \end{aligned} \qquad \text{for } n = 0, 1, \ldots, 11 \qquad (47)$$

The system equations expressing the dynamic behavior of each component at any stage n are:

$$\begin{aligned} s_1(n) &= s_1(n-1) + y_1 - u_1(n-1) \\ s_2(n) &= s_2(n-1) + y_2 - u_2(n-1) \\ s_3(n) &= s_3(n-1) + u_2(n-1) - u_3(n-1) \\ s_4(n) &= s_4(n-1) + u_3(n-1) + u_1(n-1) - u_4(n-1) \end{aligned} \qquad (48)$$

$$\text{for } n = 1, 2, \ldots, 12$$

The inflows were set at:

$$y_1 = 2 \text{ and } y_2 = 3 \qquad \text{for all time increments} \qquad (49)$$

All the above variables and constants have units of volume.

The performance criterion to be maximized is the sum of the returns due to power generated by the four power plants and the return from the diversion of $u_4(n)$ to the irrigation project;

$$F = \sum_{n=0}^{11} \sum_{i=1}^{4} b_i(n)u_i(n) + \sum_{n=0}^{11} b_5(n)u_4(n) + \sum_{i=1}^{4} g_i[s_i(N), a_i(N)] \qquad (50)$$

where F is the total return from the system for the 12 time periods; $b_i(n)$ is the unit return due to activity i, i = 1, ..., 5, during a period starting at stage n and lasting until stage n + 1; and $g_i[s_i(N),a_i(N)]$ is a function which assesses a penalty to the system when the final state of the i-th component of the system at stage N is $s_i(N)$ instead of the desired state $a_i(N)$, i = 1, 2, 3, 4. Such a penalty function for traditional dynamic programming where boundary conditions may not be satisfied is necessary.

The penalty function in equation (50) was assumed to be

$$g_i[s_i(N),a_i(N)] = \begin{cases} -40[s_i(N) - a_i(N)]^2 & \text{if } s_i(N) \leq a_i(N) \\ 0 & \text{Otherwise} \end{cases} \tag{51}$$

The desired state vectors of the initial and final stages for i = 1, 2, 3, 4 were assumed to be

$$a(0) = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \end{bmatrix} \qquad \text{and } a(N) = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 7 \end{bmatrix} \tag{52}$$

There are a total of five activities in the above criterion: four hydro-power generation activities and one irrigation activity. The unit return functions of these activities $b_1(n)$, i = 1, 2, ..., 5, are given in Table 1.

TABLE 1. Return Functions Used to Calculate Optimal Policies of System in Figure 7

n	$b_1(n)$	$b_2(n)$	$b_3(n)$	$b_4(n)$	$b_5(n)$
0	1.1	1.4	1.0	1.0	1.6
1	1.0	1.1	1.0	1.2	1.7
2	1.0	1.0	1.2	1.8	1.8
3	1.2	1.0	1.8	2.5	1.9
4	1.8	1.2	2.5	2.2	2.0
5	2.5	1.8	2.2	2.0	2.0
6	2.2	2.5	2.0	1.8	2.0
7	2.0	2.2	1.8	2.2	1.9
8	1.8	2.0	2.2	1.8	1.8
9	2.2	1.8	1.8	1.4	1.7
10	1.8	2.2	1.4	1.1	1.6
11	1.4	1.8	1.1	1.0	1.5

Larson [1968] solved this problem using linear programming and successive approximation dynamic programming. The linear nature of the objective functions of Table 1 makes the solution of a multiple-stage linear programming problem possible. The solution by this algorithm, according to Larson, is that of Table 2.

In the application of successive approximation dynamic programming, which was described in Section 2-4-1, to this problem, Larson used the trial trajectory of Table 3 (this trajectory is denoted I because later on other trajectories will be tried using the proposed technique) with a total return of 362.5. Using a B-5500 computer it took 30 seconds of computer time, and 9 iterations (iteration in successive approximation dynamic programming is defined as keeping all state variables, except one, constant and optimizing

TABLE 2. Optimal Trajectory and Policy for System in Figure 7*
Total Return = 401.3

n	$s_1^*(n)$	$s_2^*(n)$	$s_3^*(n)$	$s_4^*(n)$	$u_1^*(n)$	$u_2^*(n)$	$u_3^*(n)$	$u_4^*(n)$
0	5	5	5	5	1	4	0	0
1	6	4	9	6	0	1	0	2
2	8	6	10	4	0	2	4	7
3	10	7	8	1	2	0	4	7
4	10	10	4	0	3	3	4	7
5	9	10	3	0	3	4	4	7
6	8	9	3	0	3	4	4	7
7	7	8	3	0	3	4	4	7
8	6	7	3	0	3	4	4	7
9	5	6	3	0	3	4	4	7
10	4	5	3	0	3	4	4	0
11	3	4	3	7	0	2	0	0
12	5	5	5	7				

*Note: Optimal Trajectory and policy presented in Table 12.11 of Larson [1968] are slightly in error as noted through private communication with R. E. Larson, 1969.

TABLE 3. Trial Trajectory I and Trial Policy I for System in Figure 7
Total Return = 362.5

n	$s_1(n)$	$s_2(n)$	$s_3(n)$	$s_4(n)$	$u_1(n)$	$u_2(n)$	$u_3(n)$	$u_4(n)$
0	5	5	5	5	2	3	3	5
1	5	5	5	5	2	3	3	5
2	5	5	5	5	2	3	3	5
3	5	5	5	5	2	3	3	5
4	5	5	5	5	2	3	3	5
5	5	5	5	5	2	3	3	5
6	5	5	5	5	2	3	3	5
7	5	5	5	5	2	3	3	5
8	5	5	5	5	2	3	3	5
9	5	5	5	5	2	3	3	5
10	5	5	5	5	2	3	3	5
11	5	5	5	5	2	3	3	3
12	5	5	5	7				

in the admissible domain of the variables state) to arrive at the optimal trajectory listed in Table 2 with a total return of 401.3.

5-1-2. Application of the Proposed Approach to the Problem

Application of the proposed approach to this system, which is invertible, starts with the assumption of a trial trajectory $\overline{s}(n)$, $n = 0, 1, \ldots, 12$, satisfying equations (46) and (52). When substituted in equation (48) together with constants in equation (49) the trial trajectory will produce a trial policy $\overline{u}(n)$, $n = 0, 1, \ldots, 11$, which should be checked for constraints of equation (47). It is considerably easier to treat this problem as a free-end point problem, i.e., not to satisfy the initial or final boundary condition. However, the simplicity of the system equations in this example makes it possible to satisfy both boundary conditions in equation (52). The penalty function of equation (51) is therefore not needed in the DDDP as boundary conditions of equation (52) are always satisfied. Trial trajectory I of Table 3 was chosen and shown in Figures 8. Next, three values of $\sigma_{j,t}$ are assumed,

$$\sigma_{j,1} = 1.0, \quad \sigma_{j,2} = 0, \quad \text{and} \quad \sigma_{j,3} = 1.0 \qquad \text{for } j = 1, 2, 3, 4 \tag{53}$$

and a set of $T^m = 3^4$ incremental vectors is formed which when added to the trial trajectory I produces a sub-domain consisting of 81 lattice points at each stage.

It took 7 iterations for trial trajectory I in Table 3 to converge exactly to the optimal trajectory in Table 2. These iterations are illustrated in Figures 8. In iteration 1 of these figures, the trial trajectory I of Table 3 is shown with dashed lines, the corridor defined by equation (53) is shown in blank strips, and the improved trajectory within the corridor is shown in solid lines. In iteration 2 the improved trajectory of iteration 1 is considered as the trial trajectory (shown in dashed lines), the corridor defined by equation (53) is shown in blank strips, and the improved trajectory is presented in solid lines.

At iteration 8 it was noticed that no more improvement could be made on the improved trajectory of iteration 7, i.e., the improved trajectory in iteration 8 coincided with the trial trajecotry in that iteration. Therefore, it was concluded that the improved trajectory of iteration 7 is the optimal trajectory and convergency had been achieved.

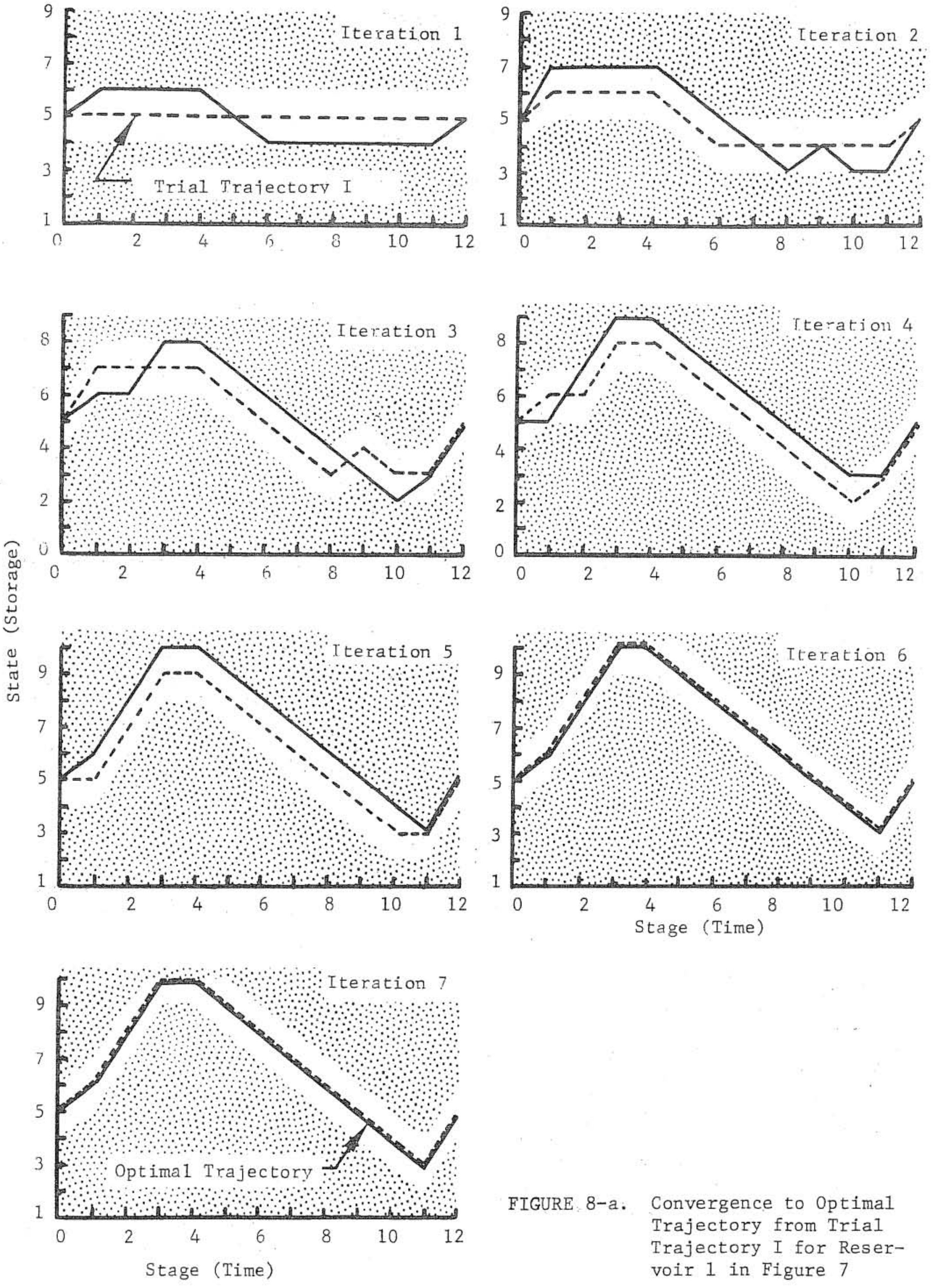


FIGURE 8-a. Convergence to Optimal Trajectory from Trial Trajectory I for Reservoir 1 in Figure 7

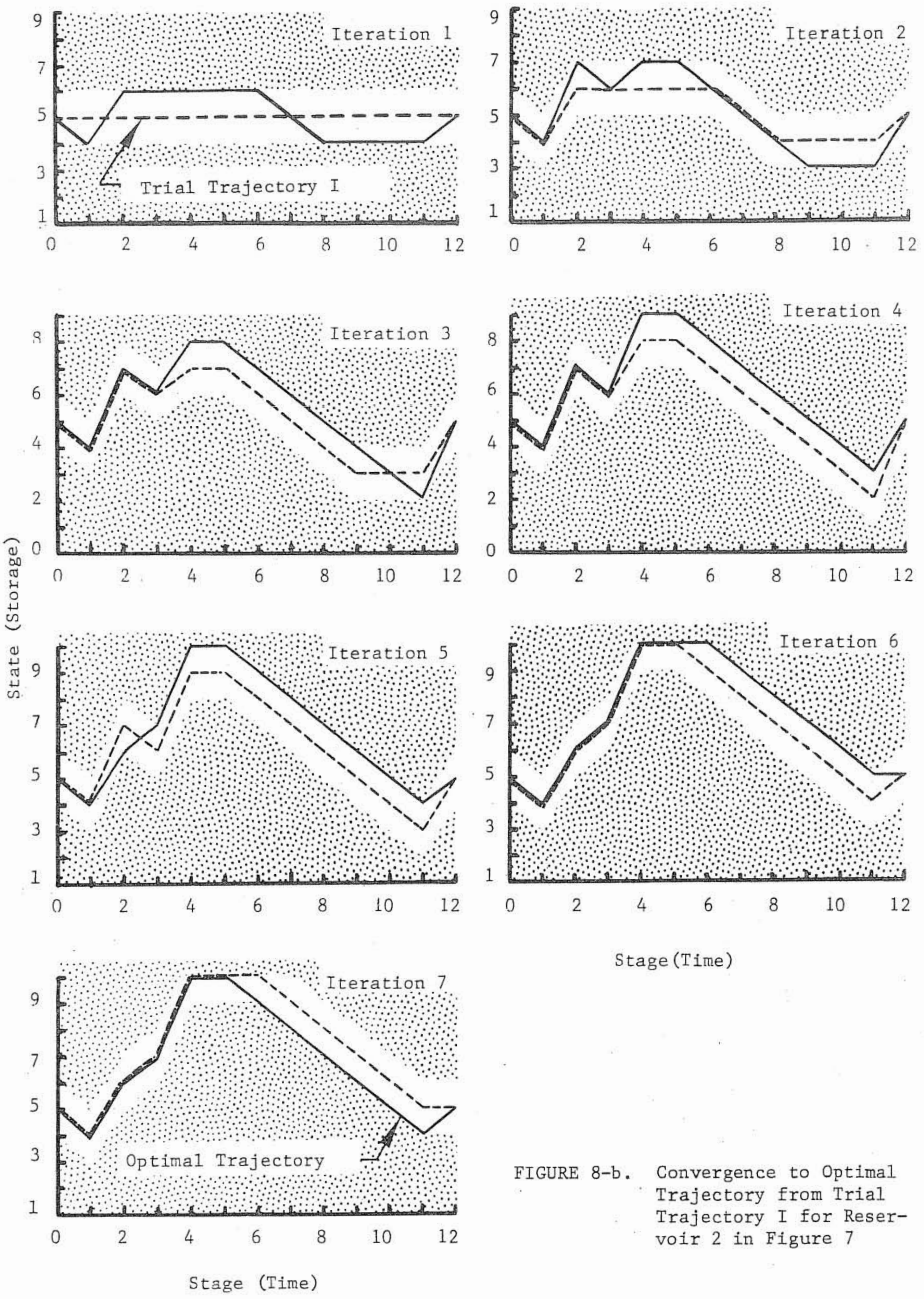


FIGURE 8-b. Convergence to Optimal Trajectory from Trial Trajectory I for Reservoir 2 in Figure 7

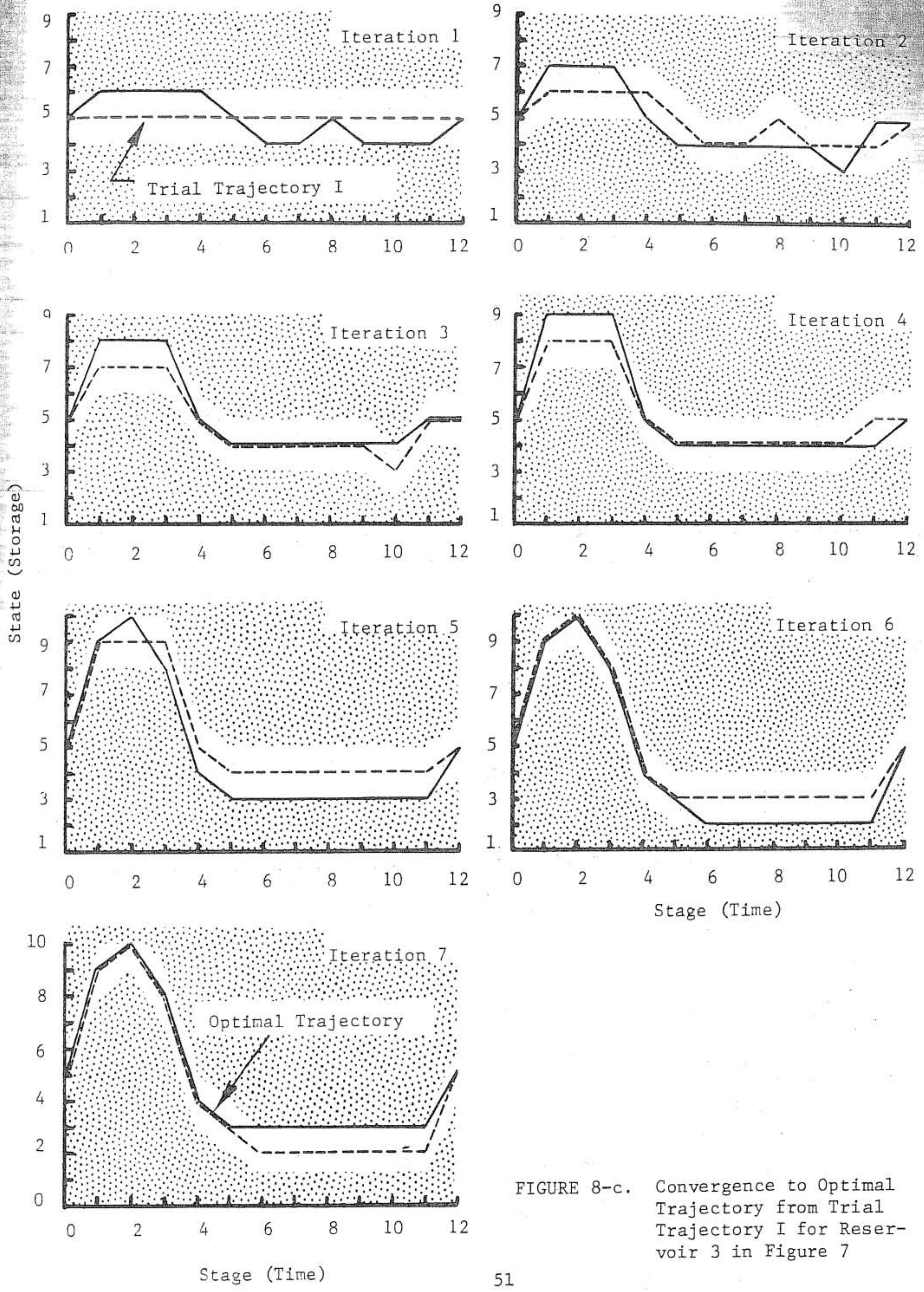


FIGURE 8-c. Convergence to Optimal Trajectory from Trial Trajectory I for Reservoir 3 in Figure 7

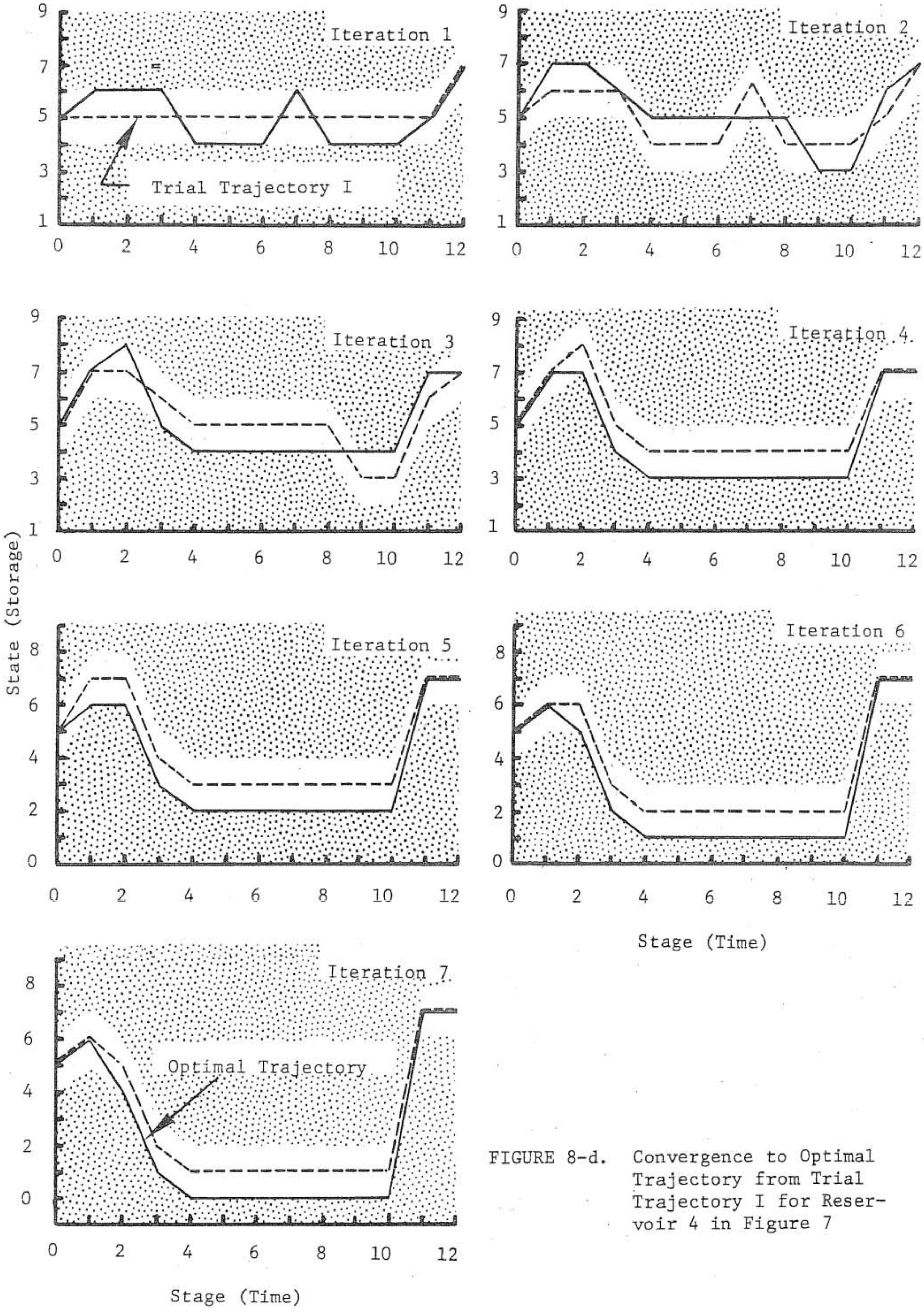


FIGURE 8-d. Convergence to Optimal Trajectory from Trial Trajectory I for Reservoir 4 in Figure 7

5-1-3. Effect of Different Trail Trajectories

In an effort to study the effect of various trial trajectories on the rate of convergence to the optimal trajectory, two other trial trajectories (II and III) were assumed and the DDDP approach repeated with the corridor defined by equation (53). All trial trajectories and the optimal trajectory are presented in Figure 9. Trial trajectory II took 12 iterations, and trial trajectory III took 7 iterations to converge to the optimal trajectory.

Figure 10 shows the rate of improvement of the trail trajectory with the number of iterations for each of the three trial trajectories.

5-1-4. Expansion of Time Horizon

The time horizon of the operation of the system was exanded from 12 to 24 stages to investigate the response of the system to the same inflows and objective function over a longer period of time. The results together with the trial trajectory IV are presented in Figure 11. Comparison of Figure 9 with Figure 11 indicates noticable similarity between them. If Figure 9 is considered as one cycle, Figure 11 may be considered almost two cycles of the patterns in Figure 9.

This tentative conclusion is reached with the understanding that all the functions to which the system responds are kept constant over the time of analysis. This result may allow one to expand the operating periods of the system without repeating the computations. The return due to the optimal trajectories in Figures 11 is 810.60.

5-1-5. Effect of Change of the State Sub-Domains

In the solution of this problem the invertibility of system equations, equation (48), were used, and the width of all corridors were defined by equation (53). Therefore, at every stage there were 3 lattice points in each corridor. Using these data and constant inflows of equation (49) in equation (44) to calculate the decision u, one should expect that the set of

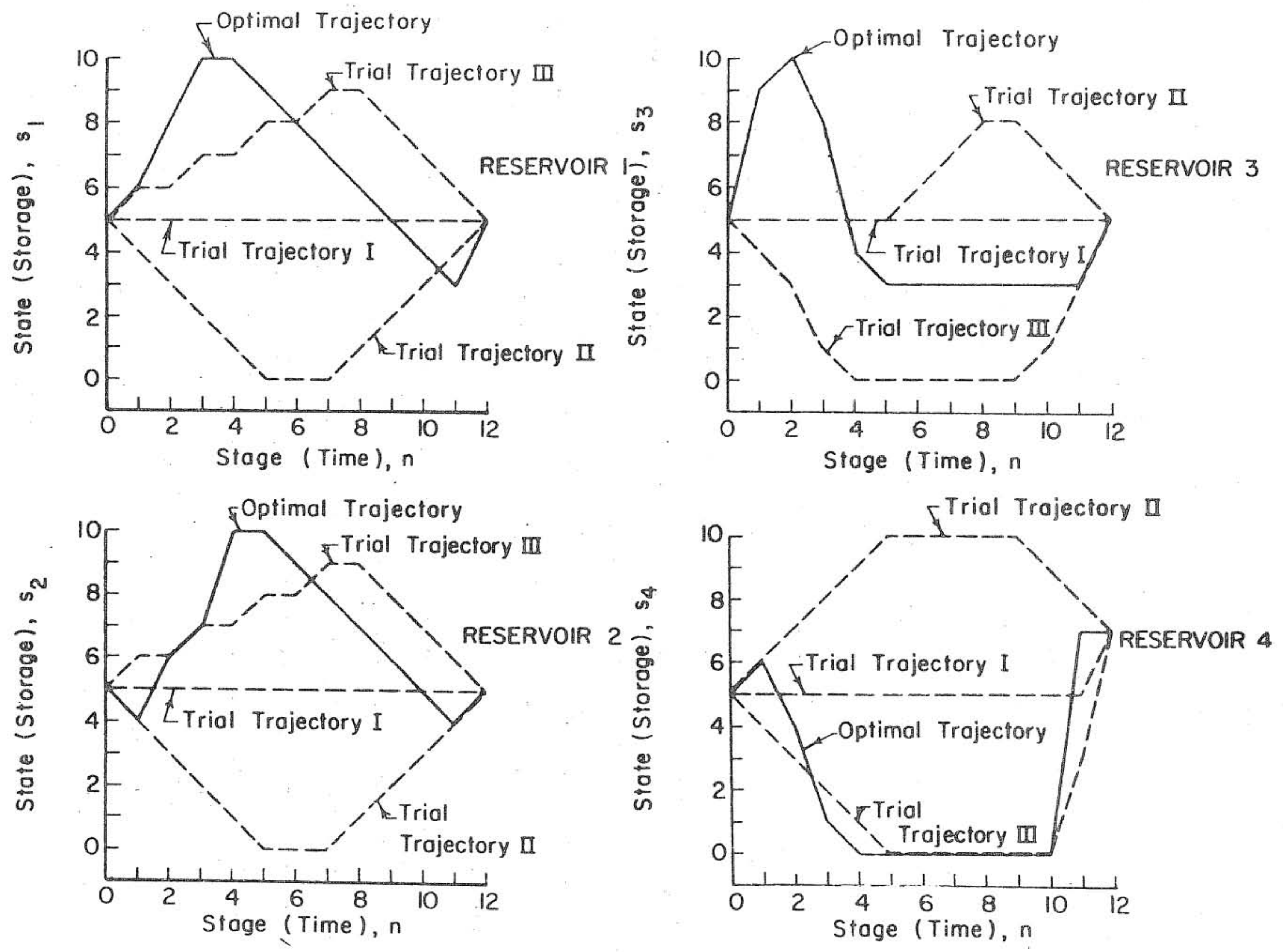


FIGURE 9. Trial Trajectories I, II, and III, and Optimal Trajectory of System in Figure 7 Using $\sigma_{j,1} = 1.0$, $\sigma_{j,2} = 0.0$, and $\sigma_{j,3} = -1.0$ for $j = 1, 2, 3, 4$

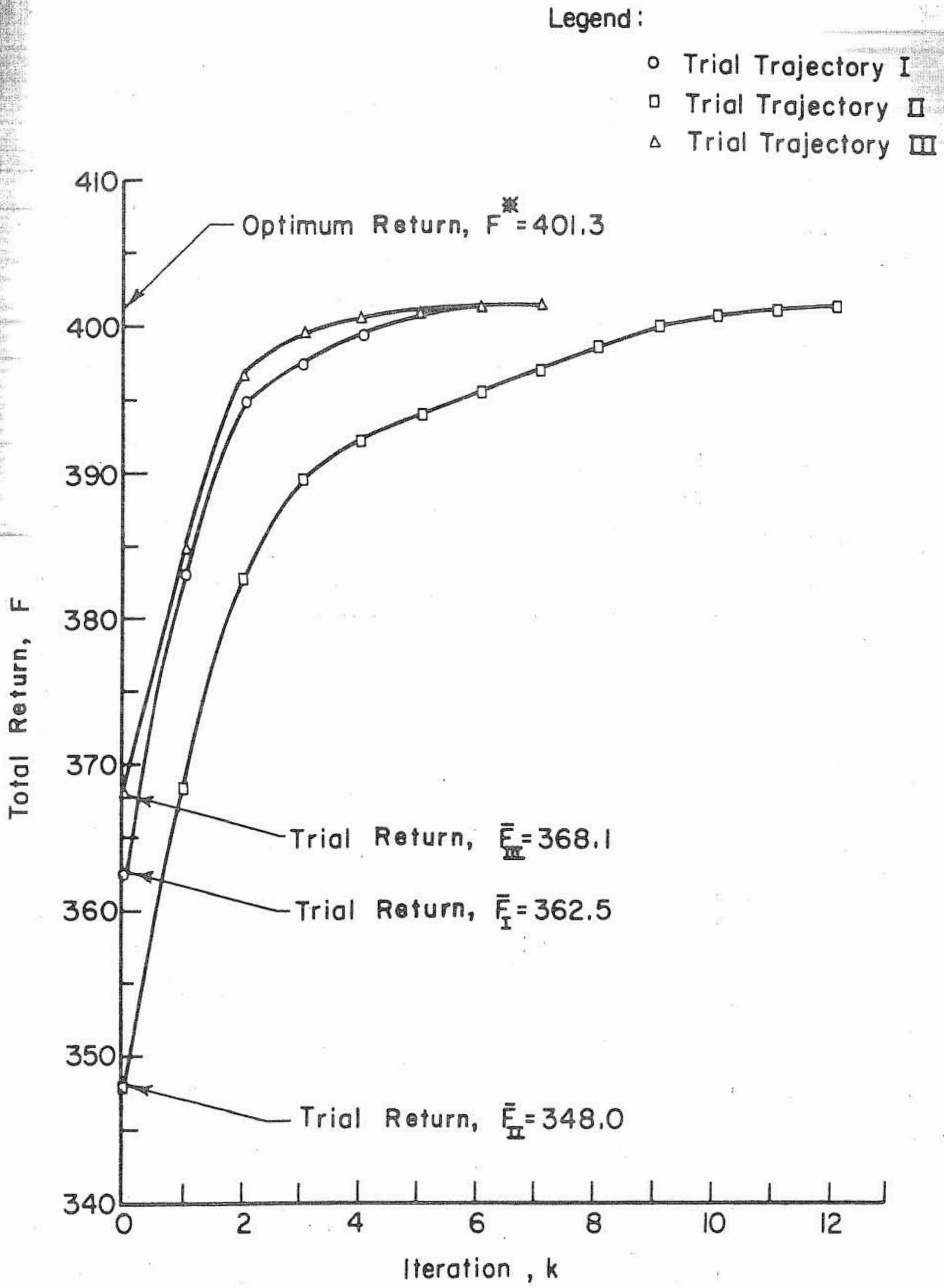


FIGURE 10. Total Return, F, as a Function of Number of Iterations for Trail Trajectories I, II, and III

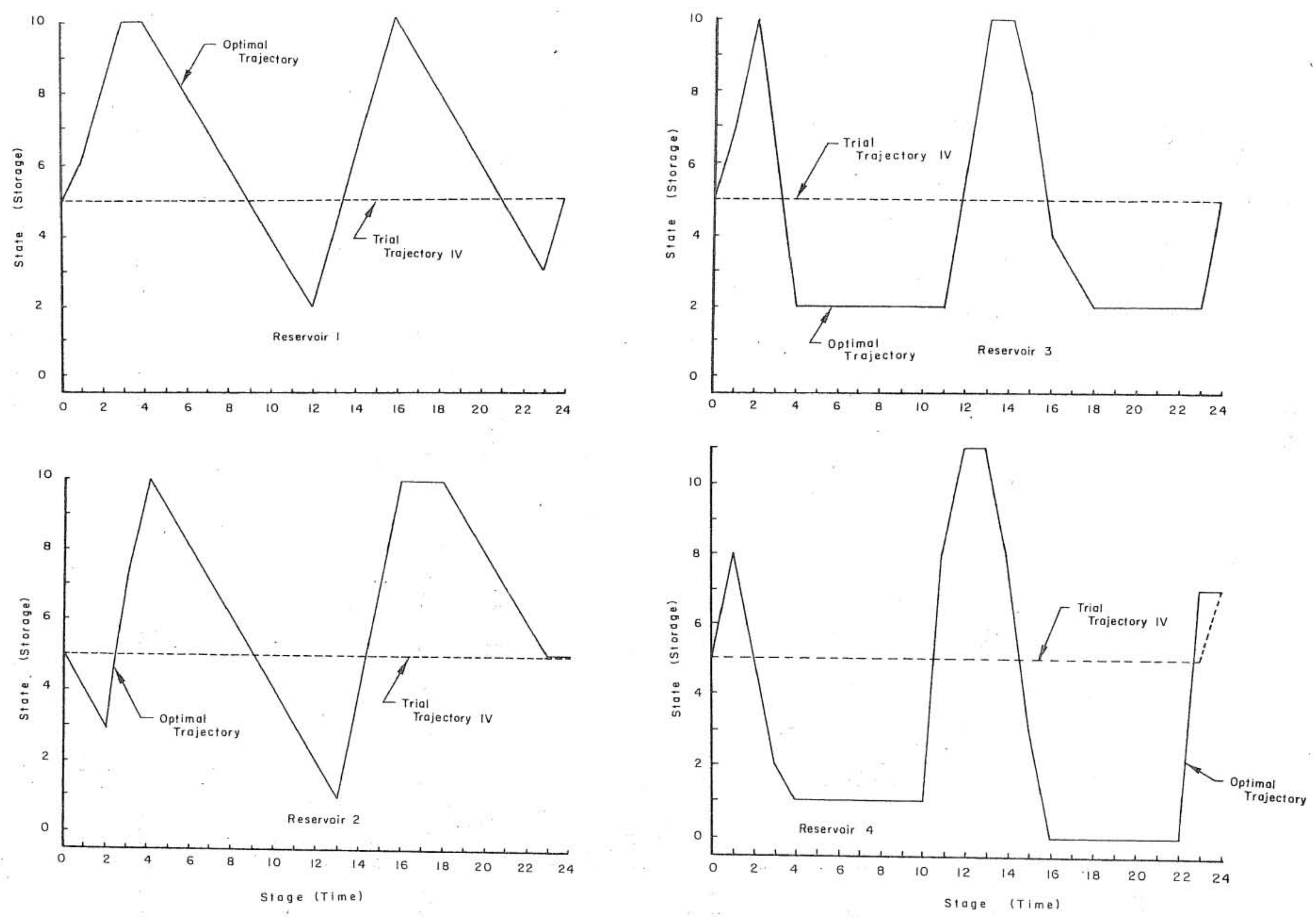


FIGURE 11. Trial and Final Trajectories IV for System in Figure 7 During 24 Periods of Operation

decisions for a trajectory which cannot be improved further may be near optimum rather than optimum. This is simply due to the fact that with the above set of constraints, the change of decision per iteration, Δu, is also constant. This change may not be the optimum change.

In order to overcome this difficulty, it is suggested that $\sigma_{j,t}$ be reduced at certain iterations as described below. A change in $\sigma_{j,t}$ at every iteration may not be efficient as far as the rate of convergence is concerned. In fact, a fast change of $\sigma_{j,t}$ with every iteration may reduce the width of the corridor to near zero, and thus make the iteration process very insufficient. Instead of changing $\sigma_{j,t}$ at every iteration, one may fix the width of the corridor and continue iterating until no more improvements can be made in a corridor of that width. This means that any more changes in decisions caused through equation (44) by changing the state at stage n and n-1 will violate the constraints of equation (47). Now, a reduction in $\sigma_{j,t}$ may be made which will not affect decisions as much as the previous $\sigma_{j,t}$. Again iterations are continued inside the corridor with new fixed widths until no more improvements can be made.

This process of reducing $\sigma_{j,t}$, at iterations beyond which no improvements are possible, may be continued until the widths of the corridor is smaller than some pre-specified constraint.

The reduction of corridor widths in the optimization of the example of Figure 7 did not produce any improvement on the optimal trajectory. The reasons are: (1) the optimal trajectory of this system follows full integer states, (2) the trial trajectories of Figures 9 and 11 were chosen so that they follow full integer states also, and (3) the values of $\sigma_{j,t}$, t = 1, 2, 3, and j = 1, 2, 3, 4 are set at full integers.

In a separate try, trial trajectory I is subjected to the interation process using

$$\sigma_{j,1} = 1.3, \quad \sigma_{j,2} = 0, \quad \text{and } \sigma_{j,3} = -1.3 \qquad \text{for } j = 1, 2, 3, 4 \qquad (54)$$

for all stages starting with iteration 1, and the idea of reducing $\sigma_{j,t}$, $t = 1, 2, 3$ and $j = 1, 2, 3, 4$ is employed. After a total of 18 iterations in 4 corridors the states shown by solid circles in Figure 12 are obtained, producing a return of 399.06 as compared to the optimal return of 401.3. This leads one to conclude that when the optimal values of $\sigma_{j,t}$ are not known, the result may only be considered as an approximation to optimum.

5-2. Advantages and Disadvantages of the Proposed Approach

The major factors which inspired the DDDP approach are the drawbacks which are inherent in the regular dynamic programming, namely; storage space and computer time requirements.

By limiting the optimization to the few lattice points around a trial trajectory, the storage requirements appear to have been curbed substantially. To illustrate this numerically, consider the storage requirements of the problem presented in Section 5-1.

This problem has four state variables, whose admissible ranges are given in equation (46). With values of $\sigma_{j,t}$ given in equation (53) the DDDP requires 243 words of computer memory while the memory requirement of traditional dynamic programming using the same grid size would be 63,888 words. Perhaps it is relevant to mention that the storage requirements of the same problem by successive approximation dynamic programming is even less. However, attemtps to obtain the convergence of this problem by successive approximation dynamic programming failed in this study.

Another major difficulty in applying traditional dynamic programming is the computer time requirement. This is due to the number of computations and comparisons which must be performed at each lattice point. In the above example, at each stage there are 21,296 lattice points. If the domain of the decisions given in equation (47) is divided into lattice points with $\Delta u = 1$ unit, at each state lattice point of each stage a total of

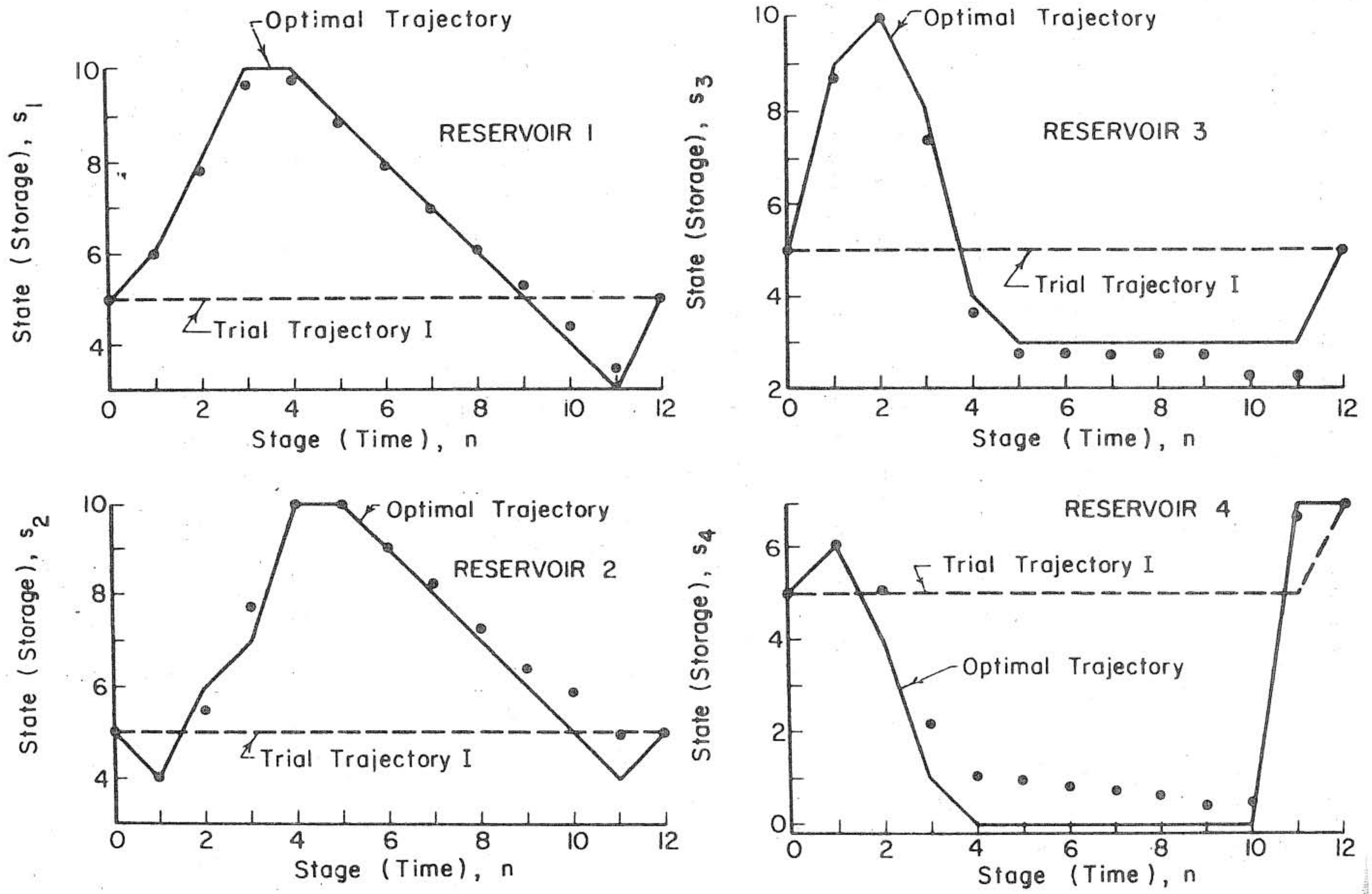


FIGURE 12. Near-Optimal Trajectory Shown by Solid Circles and Optimal Trajectory of System in Figure 7 Using Trial Trajectory I and $\sigma_{j,1} = +1.3$, $\sigma_{j,2} = 0.0$, and $\sigma_{j,3} = -1.3$ for $j = 1, 2, 3, 4$

4 x 5 x 5 x 8 = 800 combinations of decisions must be tested. By limiting the optimization to the neighborhood of a trial trajectory, the number of lattice points are reduced, and therefore, fewer tests will have to be made per state of each stage. Furthermore, if the system is invertible, this efficiency may even be increased. For example, if T = 3 at each stage, then for a four-dimensional invertible problem there are only $3^4 = 81$ possiblities that states at stage n-1 may lead to a particular state at stage n. Therefore, instead of 800 tests at a particular state of stage n only 81 tests may be made.

Table 4 summarizes the computer (IBM 360/75) processing time for the solution of the example of Section 5-1 using the DDDP approach. The number of iterations in this table is one more than what is needed to arrive at optimum results. The last iteration is required to confirm that the optimum results have been reached in the previous iteration.

TABLE 4. Computer (IBM 360/75) Time Requirements of the Proposed Approach for the Solution of System in Figure 7

Trial Trajectories	Operating Periods	No. of Iterations	Total Processing Time (sec)	Processing Time Per Iteration (sec)
I	12	8	35.32	4.42
II	12	13	48.39	3.72
III	12	8	31.04	3.88
IV	24	8	68.70	8.58

The processing time of trial trajectories I and IV may be compared to arrive at an important conclusion. Trial trajectory IV is an extension of I to 24 time periods. Therefore, one would expect that the processing time would double. As seen in Table 4, the processing time of trajectory IV is somewhat less than double that of I.

The disadvantage of the proposed technique appears when it is applied to systems with non-linear objective functions. In these cases, if the $\sigma_{j,t}$ are not chosen properly, it is possible for the improving trajectories to get stuck in a local minima or maxima depending on the objective criterion. In Chapter 6 two approaches, namely; use of a trial trajectory close to the optimum trajectory, and the choice of $\sigma_{j,t}$ by analytical techniques, to limit this disadvantage will be discussed.

5-3. Application of the Proposed Approach to Clearwater River System

In this section the proposed approach is employed to investigate the near optimal operating policy of a multiple-purpose and multiple-dimensional reservoir system.

5-3-1. The Selection of the System

The decision to choose a more realistic model for further investigation has to be made on the basis of the capability of the proposed technique for handling high-dimensional problems, the budget available for computer time, and relevance of the model.

The capability of the proposed technique as compared with the available facilities (IBM 360/75) was roughly estimated to be from 7 to 9 reservoirs, each constituting one state variable. This means that, if there were no constraint on the amount of computer time, this technique, it is believed, could handle the high speed storage requirement of a 7 to 9 dimensional system using the IBM 360/75. However, the optimization of such a system may require considerable amount of computer time which creates a budgetary constraint for the study. Finally, the last criterion, i.e., the relevance of the model, should satisfy the multiple-purpose objective of this study. Based on these criteria, the following system was adopted from Maass, et al. [1962].

5-3-2. The System

The system presented in Figure 13 consists of five streams, four reservoirs, two hydropower plants and a farm. This system is to be operated for N intervals of time to provide irrigation water, and hydroelectric power. The operating procedure of the system during the time interval between stage n and n+1, (n and n+1$\in$N), may be sumarized as follows:

$y_i(n)$ = volume of inflow into reservoir i during the time interval starting at stage n; and

$u_i(n)$ = volume of outflow from reservoir i during time interval starting at stage n; for

$i = 1, 2, 3, 4$ and $n = 0, 1, \ldots, N-1$.

The outflow from reservoir 1, $u_1(n)$, after passing through power plant 1, PP1, joins the outflow from reservoir 2, $u_2(n)$, and river inflow in the reach between the two reservoirs, $y_4(n)$ to form the flow at node 1. Therefore,

$$FL1(n) = u_1(n) + u_2(n) + y_4(n) \tag{55}$$

where FL1(n) is the volume of flow at node 1 during the time interval starting at stage n. Here and in the following mass balance equations the losses due to evaporation, seepage, etc. are ignored.

Depending on irrigation demand part or all of this flow, $u_5(n)$, may be diverted at node 2 towards the farm. The rest of FL1(n), if any, joins with the irrigation water return from the farm, $\alpha(n)u_5(n-1)$, at node 3 to form

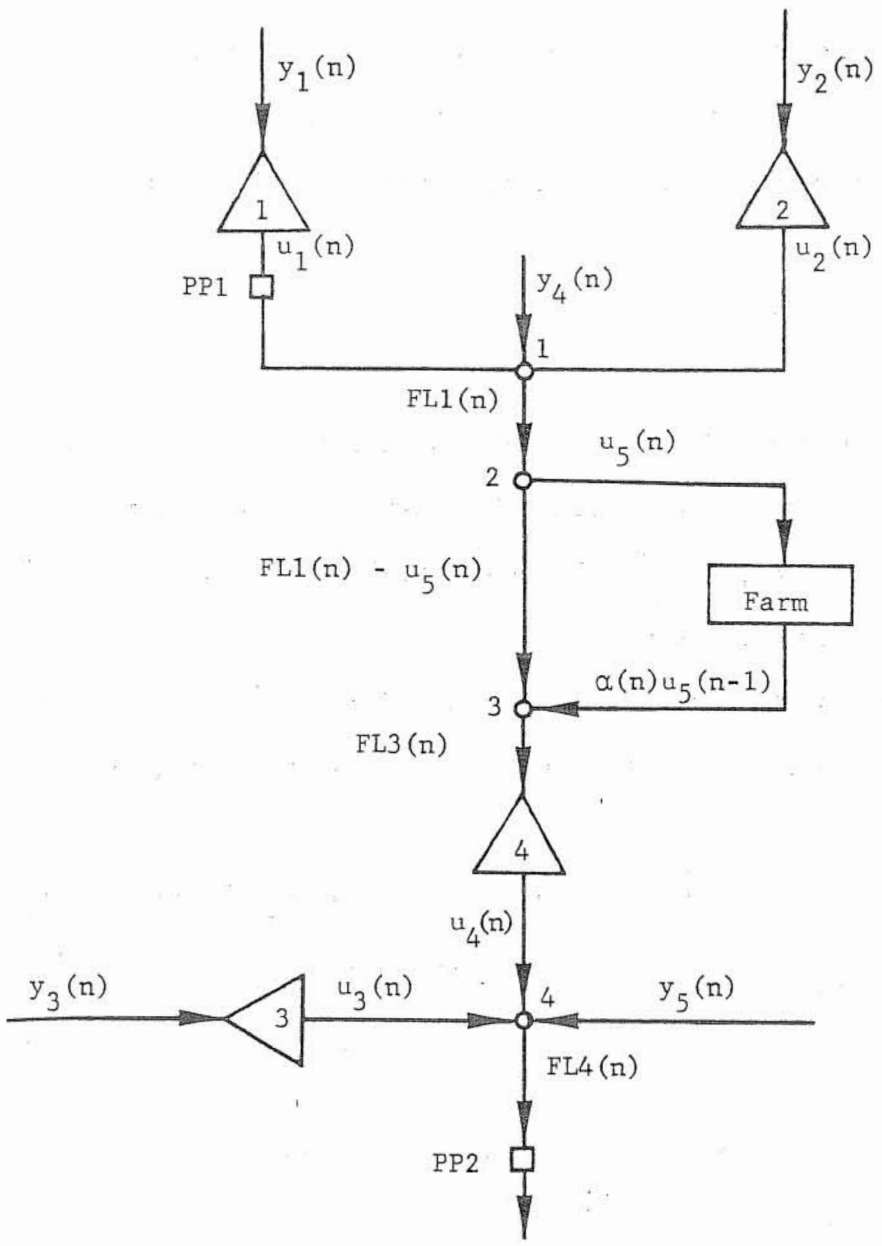


FIGURE 13. Schematic Representation of the Clearwater River System

the inflow into reservoir 4. Here $\alpha(n)$ represents the portion of the water diverted for irrigation during the previous time interval that enters node 3 during the present time interval. Therefore, flow at node 3 is:

$$FL3(n) = FL1(n) - u_5(n) + \alpha(n)u_5(n-1) \tag{56}$$

where FL3(n) is the volume of flow at node 3 during time interval starting at stage n; $u_5(n-1)$ is the total volume of diverted flow for irrigation during the previous time interval; and $\alpha(n)$ is a constant between 0 and 1 for the time interval starting at stage n.

Introduction of $u_5(n-1)$ into equation (56) creates a time lag in one of the input elements of the system. For this system a lag of one represents one month.

Outflows from reservoirs 3 and 4, $u_3(n)$ and $u_4(n)$, join with the river inflow $y_5(n)$ at node 4 to form the inflow into power plant 2, PP2. Therefore,

$$FL4(n) = u_3(n) + u_4(n) + y_5(n) \tag{57}$$

where FL4(n) is the volume of flow at node 4 during the time interval starting at stage n. This flow, FL4(n), passes through the power plant 2, PP2, to generate power.

It is assumed that the events and activities related to water downstream from power plant 2 do not influence the operation of the system. This assumption essentially isolates the system and makes the analysis simpler.

5-3-3. System's Formulation

The system equations describing the continuity principle are:

$$\begin{aligned} s_1(n+1) &= s_1(n) + y_1(n) - u_1(n) \\ s_2(n+1) &= s_2(n) + y_2(n) - u_2(n) \\ s_3(n+1) &= s_3(n) + y_3(n) - u_3(n) \\ s_4(n+1) &= s_4(n) + FL3(n) - u_4(n) \\ &\text{for } n = 1, \ldots, n-1 \end{aligned} \tag{58}$$

where $s_i(n)$ is the state (storage) of reservoir i, i = 1, 2, 3, 4, at stage n. There are four state and four decision variables involved in this system. The decision for irrigation, $u_5(n)$, is made on the basis of irrigation demands and availability of water at node 2. This means that enough water at node 2 is diverted to satisfy these demands. If, after diverting all the available water, still the irrigation demands are not satisfied, steps will be taken to penalize the returns for the shortage, or decisions regarding the outflows from reservoirs 1 and 2 will be modified.

The system's constraints on the states may be presented as follows:

$$\begin{aligned} 0.0 &\leq s_1(n) \leq 15.0 \times 10^6 \text{ ac. ft.} \\ 0.0 &\leq s_2(n) \leq 1.5 \times 10^6 \text{ ac. ft} \\ 0.0 &\leq s_3(n) \leq 13.0 \times 10^6 \text{ ac. ft.} \\ 0.0 &\leq s_4(n) \leq 15.0 \times 10^6 \text{ ac. ft.} \end{aligned} \tag{59}$$

The constraints on the decisions during any operating period are:

$$\begin{aligned}
0 &\le u_1(n) \le s_1(n) + y_1(n) \\
0 &\le u_2(n) \le s_2(n) + y_2(n) \\
0 &\le u_3(n) \le s_3(n) + y_3(n) \\
0 &\le u_4(n) \le s_4(n) + FL3(n) \\
0 &\le u_5(n) \le \min.[\beta(n)(4.0 \times 10^6 \text{ ac. ft.}), FL1(n)]
\end{aligned} \tag{60}$$

where $\beta(n)$ is the monthly irrigation demand relative to the annual irrigation demand and is computed from Table 5; and 4.0×10^6 ac ft is the yearly irrigation demand.

The parameters for power generation are:

$$\begin{aligned}
E &= 2.0 \times 10^9 \text{ kw hr/yr} \\
CPP1 &= 200 \text{ Mw} \\
CPP2 &= 200 \text{ Mw}
\end{aligned} \tag{61}$$

where E is the total energy to be generated per year; CPP1 is the installed capacity of Power Plant 1; and CPP2 is the installed capacity of Power Plant Plant 2.

The total return during a time interval starting at stage n is:

$$R(n) = BIR(n) + BPR(n) \tag{62}$$

where R(n) is the total return from the system; BIR(n) is the gross return from irrigation activities; and BPR(n) is the gross return from hydropower generation.

Therefore, for N periods of operation the total return from the system or performance criterion is:

$$F = \sum_{n=0}^{N-1} R(n) \tag{63}$$

The objective of this analysis is to optimize equation (63) by choosing a set of $u_i(n)$, $i = 1, \ldots, 4$ and hence $u_5(n)$, for $n = 0, 1, \ldots, N-1$, where the time interval between stages is one month, subject to equations (60), (61), and (62). As will be seen later the results of this analysis may only be considered as sub-optimum.

5-3-4. Return and Cost Functions

Most of the return and cost functions presented below are adopted from Maass, et al. [1962]. Some of the assumptions made in the development of these functions are not as realistic as they should be. However, since the emphasis in this study is put on the development of an approach for water resources system analysis, these assumptions do not affect the method of analysis. For the analysis of an actual system, these functions have to be developed by extensive and time consuming studies and surveys directed by specialists in several disciplines of science and engineering.

5-3-4-1. Irrigation

Demand. One of the major activities of the system is to supply irrigation water. The consumptive use and diversion requirements for irrigation were calculated from the climatiological data for irrigation practices in the Lewiston region of Idaho [Maass, et al. 1962]. This area is considered a semiarid area with a growing season of 205 days. The crops grown in this area consist of alfalfa, pasture grass, sugar beets, potatoes, small grains, and fruits. In this problem it is assumed that the crop pattern remains the same from year to year, but the amount of land allocated to farming can vary depending on the annual target irrigation water and land availability. Maass, et al. [1962] calculated the unit net consumptive use during each irrigation season by applying the method presented by Blaney and Criddle [1952] for determining water requirements in irrigated areas to a fixed crop pattern and found it to be about 2.0 feet (2.0 acre feet per acre). Then, asumming that the consumptive use of the cropped area during the nonirrigation seasons was equaled by the average precipitation during that part of the year, and assuming an average conveyance (seepage and deep perculation) loss of 30 percent of the annual diversion requirement, the unit irrigation diversion requirement was calculated to be 5.0 feet (5 acre feet per acre) yearly. This unit irrigation-diversion requirement distributed by months is shown in Table 5.

TABLE 5. Assumed Monthly Distribution of the Annual Target Irrigation Diversion Requirements*

Month	Percentage of Target Annual Diversion for Irrigation
April	12.4
May	14.6
June	16.6
July	19.0
August	18.0
September	12.4
October	7.0
November-March	0.0
Total	100.0

*Source: Maass, et al. [1962]

The distribution of irrigation diversion requirements of Table 5 inherently assumes that the marginal cost of the farm facilities, labor, and other resources required to operate the farm are equal to the marginal benefit of irrigation water diversion. In brief, it is assumed that for each month of the irrigation season the farm irrigation system has been designed and operated at the optimal economic point.

Irrigation Return Flow Into the System. The irrigation return flow may be estimated by subtracting the sum of the net consumptive use and losses during a season from the total water diverted for irrigation during the same season. Assuming that out of every 5.0 feet irrigation diversion 40% (2.0 feet) is the net consumptive use and 30% (1.5 feet) is lost to the adjacent basins and deep perculation, it may be estimated that during every operating period about 30% (1.5 feet) of the total irrigation diversion of the previous operating period will enter the system at node 3.

In this problem a time lag of one time period (one month) was assumed. At the expense of some high-speed memory a time lag of more than one may easily be introduced into the system by carrying along in the computations the optimal decisions of any number of the previous time periods desired. A set of coefficients, specifying the percentage of allocated waters that will return to the system, must also be available. For a time lag of one there is only one coefficient at every stage of computation. Therefore in equation (56) $\alpha(n) = .30$ for $n = 0, 1, \ldots, N-1$.

Irrigation Returns. The unit gross irrigation return for this system is assumed to be a function of the annual target output for irrigation. Therefore, as the annual target output for irrigation increases, the demand for irrigation water and subsequently the unit gross irrigation return decreases. This relationship is shown in Figure 14 and its equations are given in Maass *et al.* [1962] as:

$$\begin{aligned} UIR &= 0.0527A^2 - 0.6412A + 6.44924 && \text{if } A \leq 5.5 \\ UIR &= 4.5 && \text{if } A > 5.5 \end{aligned} \qquad (64)$$

where UIR is the unit gross irrigation return in dollars per acre foot and

A is the annual target output for irrigation water in 10^6 acre feet. The maximum annual target output for irrigation is to be for 1.2 X 10^6 acres of irrigable land which demands 6 X 10^6 acre-feet (ac. ft.) annually (5.0 ac. ft. of water per acre). For this amount of water a unit gross irrigation return is 4.5 dollars/acre-feet. Similarly, for lesser annual target outputs for irrigation the unit gross returns may be obtained from equation (64).

Due to the dynamic and multiple-purpose nature of the system, it is quite possible that during one or several operating periods there are irrigation shortages. Depending on the severity of these shortages, the operators of the farm may lose part or all of their crops. For example, severe shortage of water for irrigation during the germinating season of the small grains, when the presence of moisture is of great importance, can completely destroy the crop. If shortage occurs during the maturing season, the damage may be partial.

Therefore, a penalty function is needed to calculate the damages resulting from certain irrigation shortages. Figure 15 shows the assumed irrigation shortage penalty function used in this analysis. The percent loss of monthly gross irrigation returns passes the 100% mark at a shortage of about 80%. This severe penalty forces the optimizing model to choose, if possible, the policy which allocates more water for irrigation. The highest percentage of shortage beyond which no return is expected for the entire year (total destruction of crops) was arbitrarily set at 80%; i.e. when during any month of irrigation season the shortage surpassed 80% of that month's demand, the entire irrigation return during the season was assumed lost.

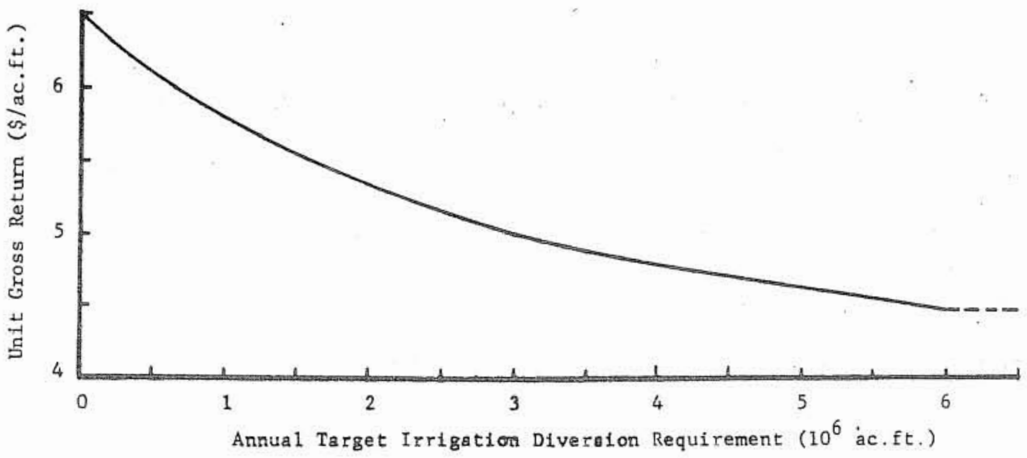


FIGURE 14. Assumed Unit Gross Irrigation Return Function (Source: Maass, et al., 1962)

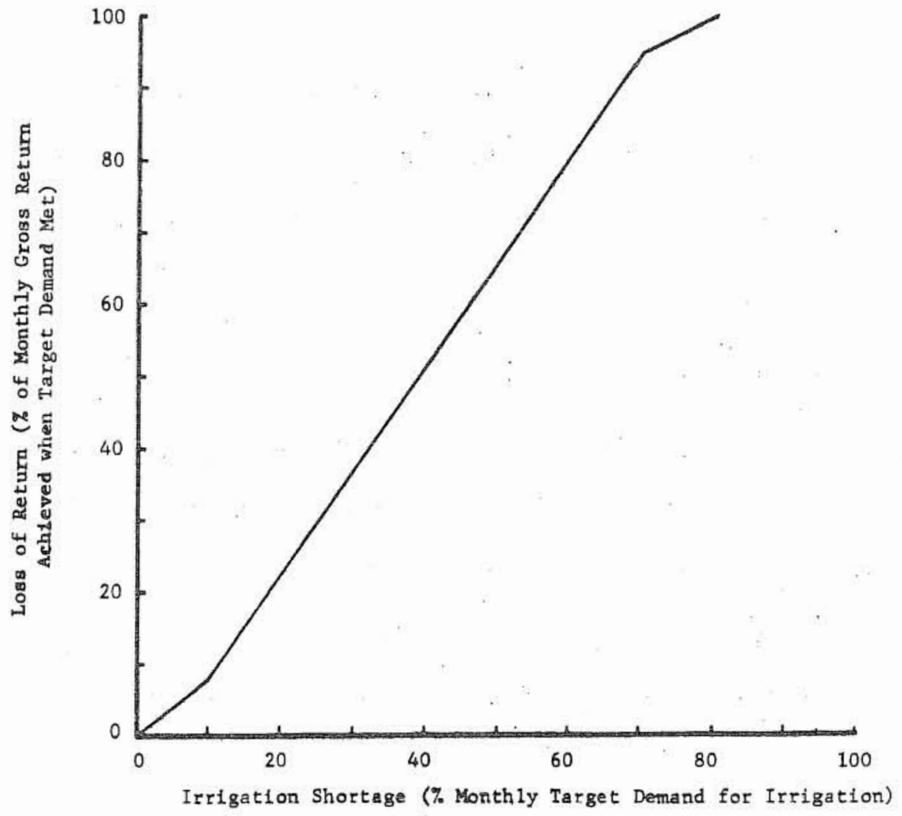


FIGURE 15. Assumed Penalty Function for Irrigation Shortage

The function in Figure 15 implies that the importance of water for irrigation during any month of the year is the same. In order to correct this shortcoming a weight evaluating the importance of a specific operating period was assigned to each of the months in the irrigation season. Thus, by multiplying the loss obtained from Figure 15, for a certain percentage of shortage, by the weight assigned to the month in operation, the importance of the irrigation water during different months was projected in the analysis. The weight describing the importance of any month of irrigation season was calculated by dividing the demand for irrigation during that month by the maximum irrigation demand during any month of the irrigation season. Therefore,

$$\gamma(n) = IR(n)/IR_{max} \qquad n = 0, 1, \ldots, N-1 \qquad (65)$$

where $\gamma(n)$ is the weight assigned to irrigation losses of the period starting at stage n; IR(n) is the demand for irrigation for the period starting at stage n; and IR_{max} is the maximum demand for irrigation water during any period.

5-3-4-2. Hydropower

Demand. Another main activity of the system is hydropower generation. The generated energy is to be distributed among domestic, rural, industrial, commercial, and irrigation activities. The survey of the demand schedule prepared by Maass, et al. [1962] for Clearwater River Basin, Idaho showed the monthly demand distribution presented in Table 6.

TABLE 6. Assumed Monthly Distribution of Annual Total Energy Output*

Month	% of Total Annual Energy Requirement
April	7.7
May	8.3
June	8.9
July	9.1
August	9.3
September	9.1
October	8.2
November	8.1
December	8.3
January	8.2
February	7.5
March	7.3
Total	100.0

*Source: Maass, et al. [1962]

This study assumed that the monthly distribution of energy demands, presented in Table 6, does not vary with the magnitude of target annual energy output. Therefore, a target annual energy output, compatible with the system, was assumed and based on the distribution of Table 6 the monthly energy demands were computed. Due to the monthly operational policies, there may be several months during which there is either a surplus or a shortage of energy. In such cases, steps must be taken to import or export power to the demand areas. The decision as to what sizes of turbines must be considered for power plant 1, PP1, and power plant 2, PP2, was based on the annual target energy output and the following assumptions:

(1) The return from unit power generated by both plants are the same. This assumption enables the incorporation of the energy outputs from the two plants into a single network for distribution.

(2) The turbines are to generate energy at relatively constant rates throughout the month. No stand-by unit was considered for reserve capacity.

Therefore, first an annual target energy output was assumed, and using a load factor of 0.6, the combined installed capacity of the turbines is calculated by:

$$CT = \frac{EO}{f \cdot hy} \qquad (65)$$

where CT is the required installed capacity in kw; f is the load factor; hy is the number of hours in a year; and EO is the annual target energy output in kw-hr. The required installed turbine capacity, CT, should be divided between the two plants in such a way that they contribute to the overall maximization of the net return from the system. For simplicity, it was assumed that each power plant can handle half of the required installed turbine capacity.

Returns from Hydropower. As mentioned before, releases from the reservoirs during any period generate hydropower which may be greater or less than the demand during the period. Surplus energy generated during any period, called dump energy, may be exported from the system, and energy may be imported into the demand area during periods when the amount of energy generated by the system is less than the demand.

In this study it was assumed that the unit return from the energy generated by the system and used in the demand area is constant at 7 mills per kilowatt-hour. The dump energy was assumed to have an unlimited market demand with a return of 1.5 mills per kilowatt-hour and the energy purchased to overcome any deficits was assumed to be available in unlimited supply at 9 mills per kilowatt-hour.

The functions which were used to calculate the energy output of the two power plants are given in Maass et al. [1962] as shown in Table 7.

TABLE 7. Power and Energy Relationships Used in Analysis of Clearwater River System*

Element of System	Dependent Variable	Independent Variables and Constants	Assumed Relationship
PP1	Effective head h(ft.)	Storage in reservoir 1, x (10^6 ac.ft.)	$h = 0.76040039E01 + 0.15802026E03x - 0.31923019E02x^2 + 0.35461025E01x^3 - 0.19495636E00x^4 + 0.41586272E{-}02x^5$

PP1 and PP2	Energy output, E(Mw hr)	u = waterflow through turbines (10^2 ac.ft./month)
		H = effective head (ft.)
		e = turbine efficiency = .85
		c_1 = conversion factor equal to .001024, converts from ft x ac.ft. to Mw hr
			$E = c_1\ euh$

PP1	Energy output per month, Em (Mw hr/month)	h_{av} = average head during month (ft) u_1 = flow through turbines (10^2 ac.ft./month) c_2 = conversion factor equal to 0.0871 which contains c_1, e, and converts from ft X 10^2 ac.ft. to Mw hr.	$E_m = c_2 h_{av} u_1$
PP1	Maximum energy output per month, E_{max} (Mw hr/month)	h_{max} = maximum head (ft) u_{max} = maximum water capacity of turbines (10^2 ac.ft./month) c_2 = as above	$E_{max} = c_2 h_{max} u_{max}$
PP1	E_{max}	CPP1 = rated power capacity of PP1 (Mw) c_3 = conversion factor equal to 730.56, converts months to hours	$E_{max} = c_3 CPP1$
PP1	Water capacity of turbines WCPP1 (10^2 ac.ft./month)	CPP1, h_{av}, h_{max} and c_4 = conversion factor equal to 8387.6, converts from (Mw/ft) to (10^2 ac.ft./month)	$WCPP1 = c_4 CPP1 \times h_{av}^{1/2}/h_{max}^{1/2}$
PP1	Maximum water capacity of turbines $WCPP1_{max}$ (10^2 ac.ft./month)	CPP1, h_{max}, c_4	$WCPP1_{max} = c_4 CPP1/h_{max}$
PP2	Energy output per month, E_m (Mw hr/month)	FL4 = flow at node 4 in Figure 13, (10^2 ac.ft.) WCPP2 = water capacity of turbines (10^2 ac.ft.) c_5 = conversion factor equal to 14.4, converts from 10^2 ac.ft. to Mw hr	If $FL4 < WCPP2$, $E = c_5 FL4$ If $FL4 \geq WCPP2$, $E_m = c_5 WCPP2$
PP2	Water capacity of turbines (10^2 ac.ft./month)	CPP2 = rated power capacity of PP2 (Mw) c_6 = conversion factor equal to 50.73, converts from Mw to 10^2 ac.ft./month	$WCPP2 = c_6 CPP2$

*Source: Maass, et al. [1962]

5-3-5. Hydrology of the System

The system in Figure 13 contains five inflows for which at least 31 years of monthly flows are available. These flows are measured at the following points:

Flow into reservoir 1:	The sum of runoff records at Selway River and Lachsa River near Lowel, Idaho.
Flow into reservoir 2:	The south fork of the Clearwater River near Grangeville, Idaho.
Flow into reservoir 3:	The north fork of the Clearwater River near Ahsahka, Idaho.
Flow at node 1:	The Clearwater River at Kamiah, Idaho.
Flow at node 4:	The Clearwater River at Spalding, Idaho.

These flows provide y_1, y_2, y_3 for the system equations, equation (58). To calculate y_4, the sum of the natural flows y_1 and y_2 were subtracted from the flow at node 1. The flow y_5 was calculated by subtracting the sum of the natural flows at node 1 and y_3 from the natural flow measured at node 4. These five inflows constitute the stochastic disturbances in the system. Any activity of the system must be related to the joint probability of these stochastic elements. These joint probabilities may be described by multi-variable statistical models such as the ones presented by Anderson [1966].

The analysis in this study used deterministic data, i.e., the mean monthly flows presented in Table 8. Such deterministic data, naturally, limit the applicability of the results. The responses of the system to the mean monthly inflows may be considered as mean or average responses.

TABLE 8. Mean Monthly Stream Flows of Streams in Figure 13*
(All flows in 10^6 ac.ft.)

Stream	April	May	June	July	Aug.	Sept.	Oct.	Nov.	Dec.	Jan.	Feb.	March
y_1	0.7112	1.5197	1.1079	0.2969	0.0895	0.0676	0.1095	0.1386	0.1663	0.1340	0.1308	0.2269
y_2	0.1133	0.1930	0.1239	0.0373	0.0128	0.0106	0.0150	0.0175	0.0193	0.0162	0.0161	0.0326
y_3	0.7675	1.1098	0.6545	0.2167	0.0955	0.0789	0.1102	0.1522	0.2184	0.1847	0.1773	0.3098
y_4	0.0794	0.0928	0.0623	0.0166	0.0053	0.0056	0.0065	0.0112	0.0160	0.0170	0.0242	0.0513
y_5	0.2449	0.1851	0.0892	0.0295	0.0095	0.0071	0.0136	0.0327	0.0720	0.0789	0.1026	0.2090

*Source: USGS Water Supply Papers No. 1317 (1930-1949), and 1737 (1950-1960).

Since irrigation activities start in April and last until the end of October, it was decided to start the operation of the system in April. This procedure guarantees that no irrigation water is left in the system at April as a result of previous year's irrigation activities.

5-3-6. Computer Program

The computer program was written in FORTRAN IV and is presented in Appendix A which contains the notations used in the input data, the program, and sample outputs. The program consists of one main routine and nine subroutines. Some of the subroutines may be integrated with the others to make the compiling time shorter. Following is a brief description of each part:

MAIN:	This is the routine in which most other routines are called.
INPUT:	This subroutine reads all the input data, and, if necessary, lists them.
DETAIL:	Given a trajecotry and policy, this subroutine evaluates and lists the energy, and irrigation activities for each operating period, and sums them for the entire operating horizon.
POWER:	Given a set of outflows from reservoir 1 and at node 4 during any period, this subrouting calculates the total energy generated, compares it with the demand, and evaluates the returns and deficits.
IRRIG:	This subroutine evaluates the returns due to irrigation activities using the functions described by Figures 14 and 15.
STATE:	This subroutine uses the $\sigma_{j,t}$ provided in input data to set up a corridor around the trial trajectory.
DPD:	This subroutine uses the forward deterministic algorithm of dynamic programming to maximize the performance criterion of

equation (63) subject to equations (58), (59), (60), and (61) for the states inside the corridor.

TRACE: Given a final state located inside the corridor this subroutine traces the trajectory which starts with the initial state and ends with the final state.

PLOT 1: Given a trajectory this routine plots state vs. stage for each reservoir using a Calcomp plotter.

5-3-7. Constraints and Parameters Used in the Analysis

In the analysis of this system by the DDDP approach, the following constraints and parameters were assumed:

1. Irrigation target demand was set at 4.0×10^6 ac.ft./yr., and monthly target demands were calculated using Table 5. The percentage of irrigation return was set at $\alpha(n) = .3$, for $n = 0, 1, \ldots, N-1$.
2. Power target demand and installed power plant capacities are given by equation (61) and the monthly target power demands were calculated using Table 6.
3. The maximum and minimum capacity of reservoirs are given by equations (59).
4. The desired initial and final state vectors were set as follows:

$$a(0) = a(N) = \begin{bmatrix} 10.0 \\ 1.0 \\ 9.0 \\ 10.0 \end{bmatrix} \tag{66}$$

5. The state sub-domain at each stage was computed from the following assumed increments from the state variables:

$$\begin{array}{lll} \sigma_{1,1} = -\ 0.3 & \sigma_{1,2} = 0 & \sigma_{1,3} = +\ 0.3 \\ \sigma_{2,1} = -\ 0.03 & \sigma_{2,2} = 0 & \sigma_{2,3} = +\ 0.03 \\ \sigma_{3,1} = -\ 0.3 & \sigma_{3,2} = 0 & \sigma_{3,3} = +\ 0.3 \\ \sigma_{4,1} = -\ 0.3 & \sigma_{4,2} = 0 & \sigma_{4,3} = +\ 0.3 \end{array} \qquad (67)$$

6. Based on the above parameters and constants two trial trajectories, A and B, of Figures 16 and 17 were calculated.

5-3-8. Discussion of Analysis

Using the trial trajectories A and B of Figures 16 and 17, the system was analyzed for 12 monthly operating periods. In order to achieve near optimum results the technique of variable corridor width described in Section 5-1-5 was used. First a set of state sub-domains computed from equations (67) were adopted, and iterations continued in six corridors each with a width equal to 0.7 that of the previous corridor. At the end of each iteration a trajectory satisfying the boundary conditions in equation (66) was traced. Figures 16 and 17 present the trial trajectories and their respective final trajectories obtained after 30 iterations. Figures 18 and 19 show the rate of convergence to the final trajectory for different trial trajectories. Figures 20 and 21 show the results for trajectory C which is the extension of trajectory B to 24 months of operation. Table 9 summarizes the results of optimization for the three trial trajectories.

Comparison of the final trajectories A and B reveals that even though they are rather similar, differences exist between them. Results of Table 9 indicate that final trajectory B provides about \$57,000 (about 0.17%)

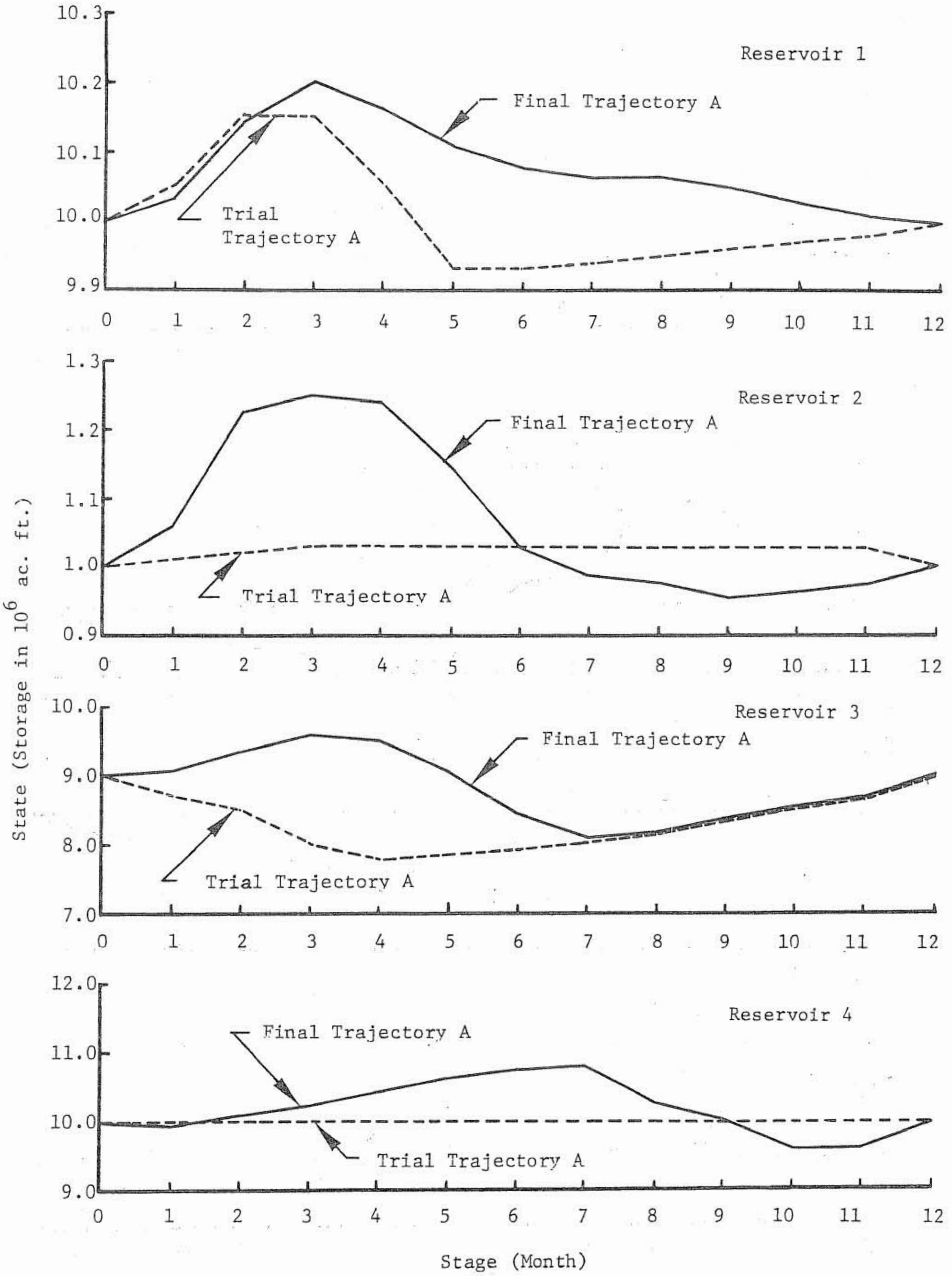


FIGURE 16. Trial and Final Trajectories A for System in Figure 13 During 12 Months of Operation

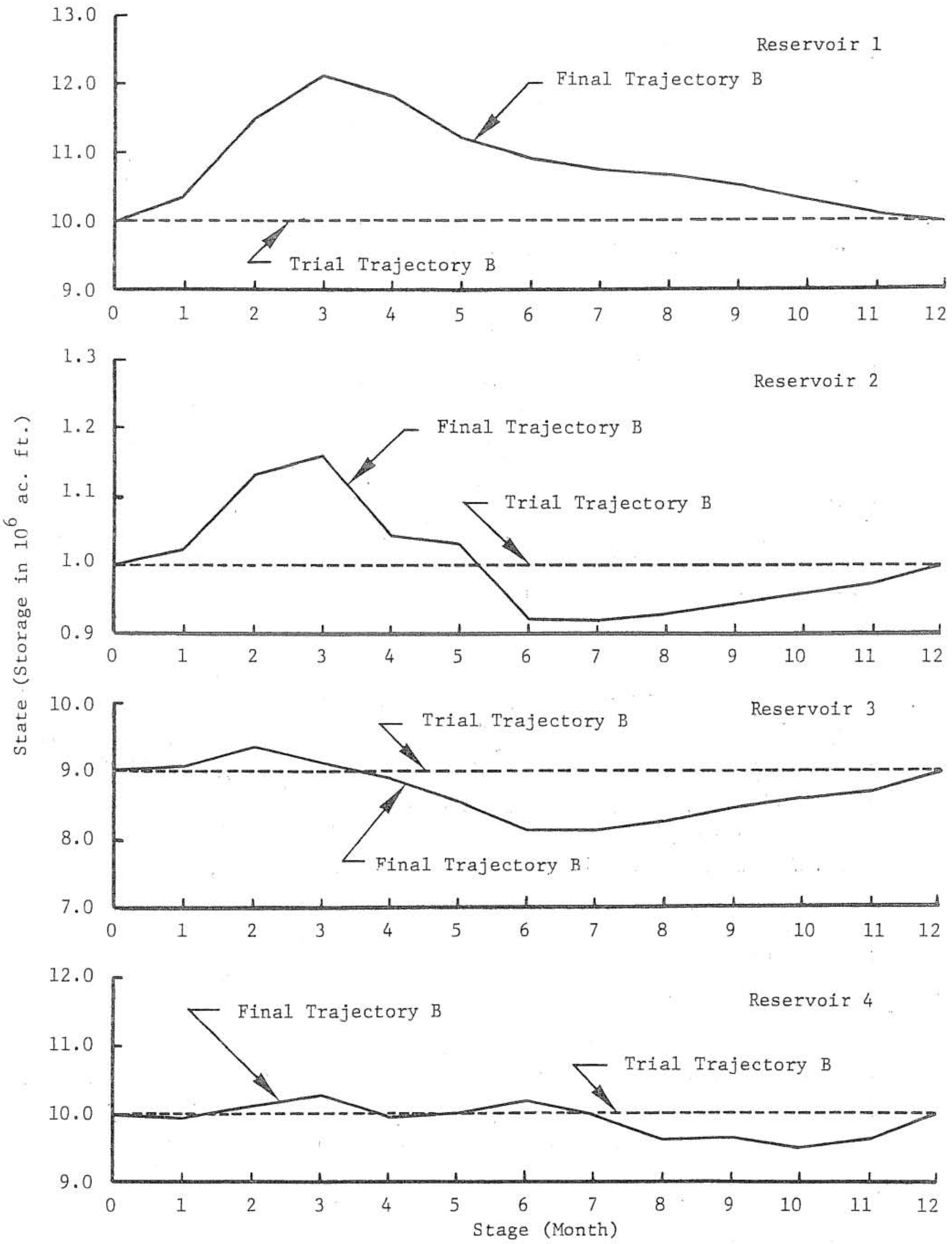


FIGURE 17. Trial and Final Trajectories B for System in Figure 13 During 12 Months Operation

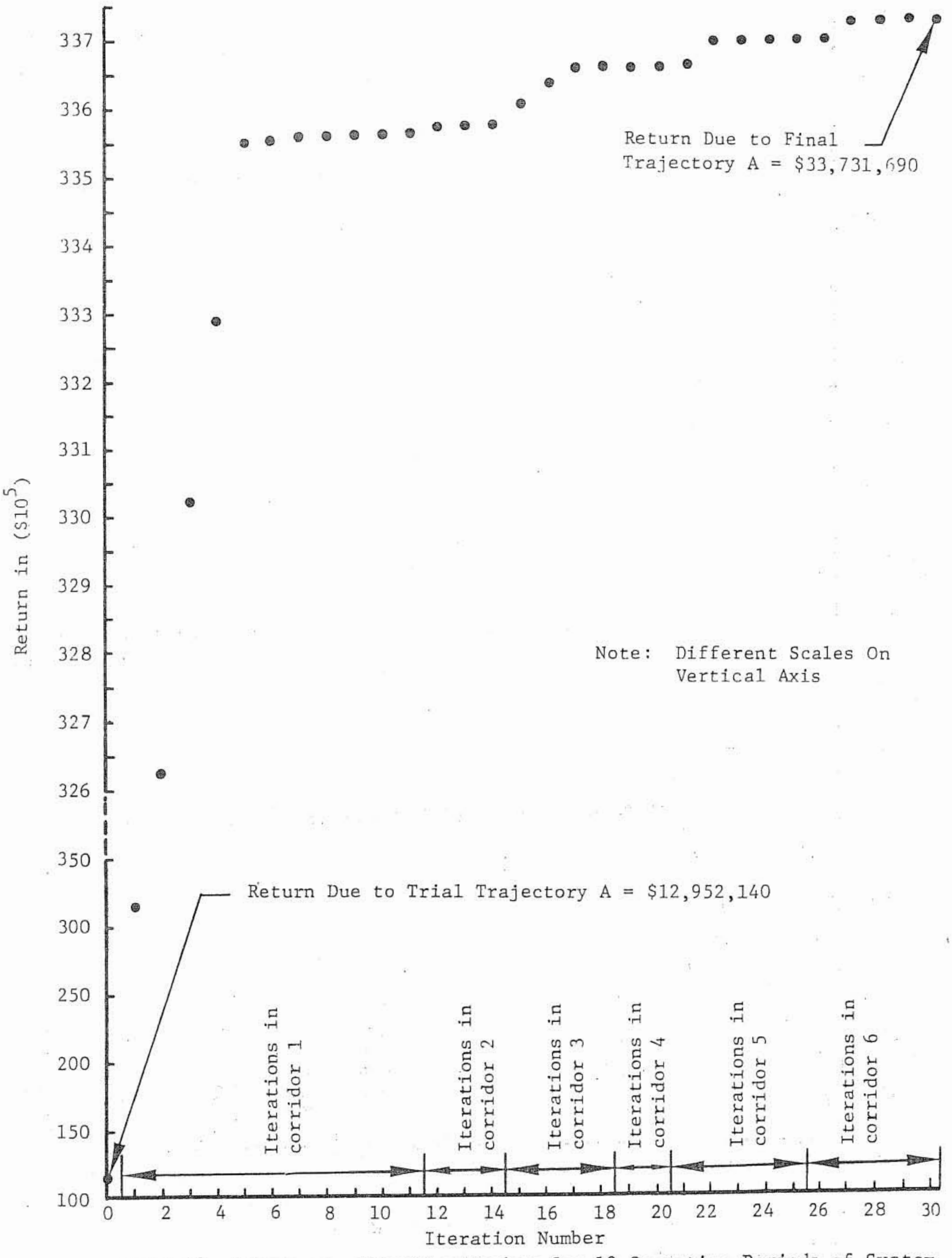


FIGURE 18. Return vs. Iteration Number for 12 Operating Periods of System in Figure 13 Using Trial Trajectory A

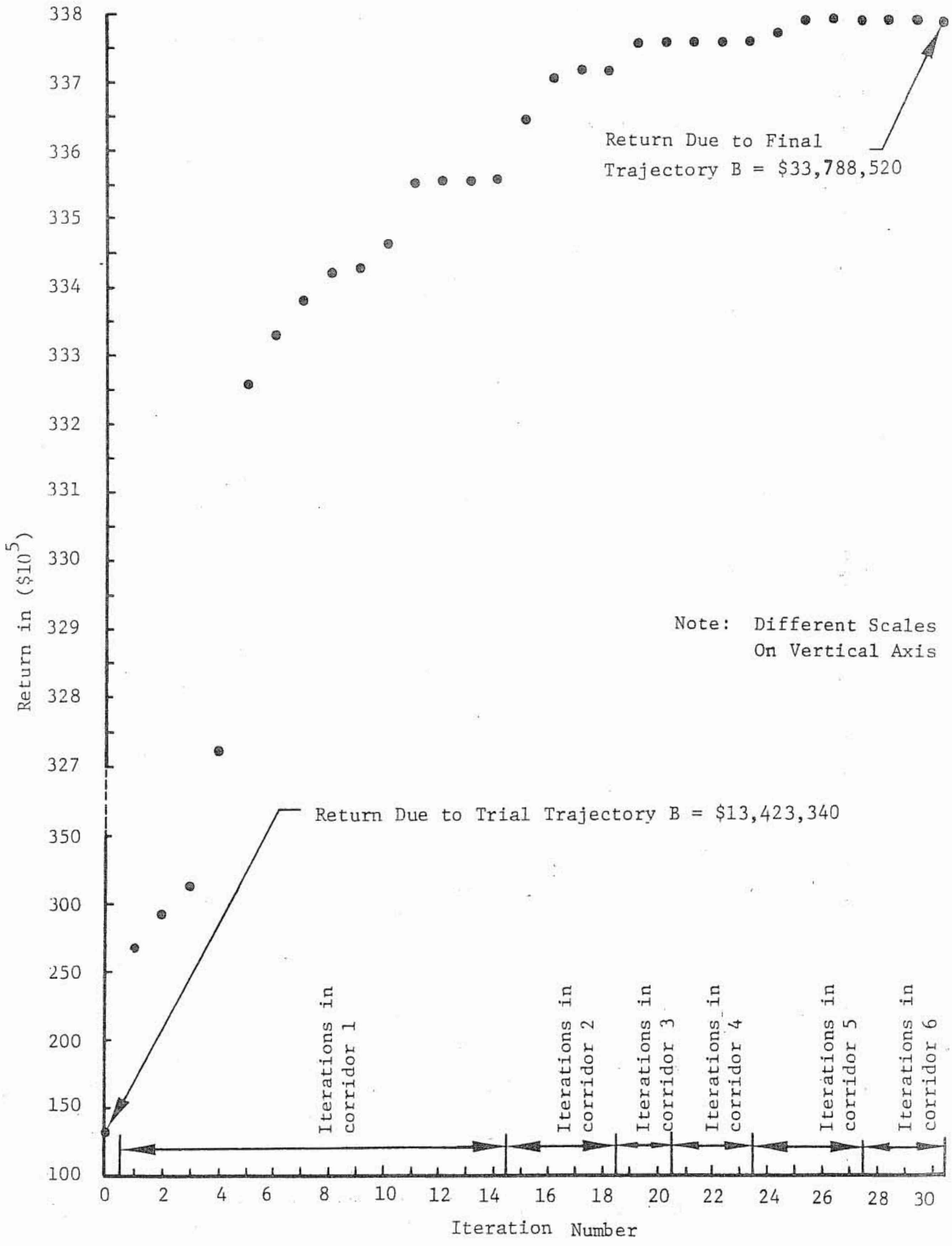


FIGURE 19. Return vs. Iteration Number for 12 Operating Periods of System in Figure 13 Using Trial Trajectory B

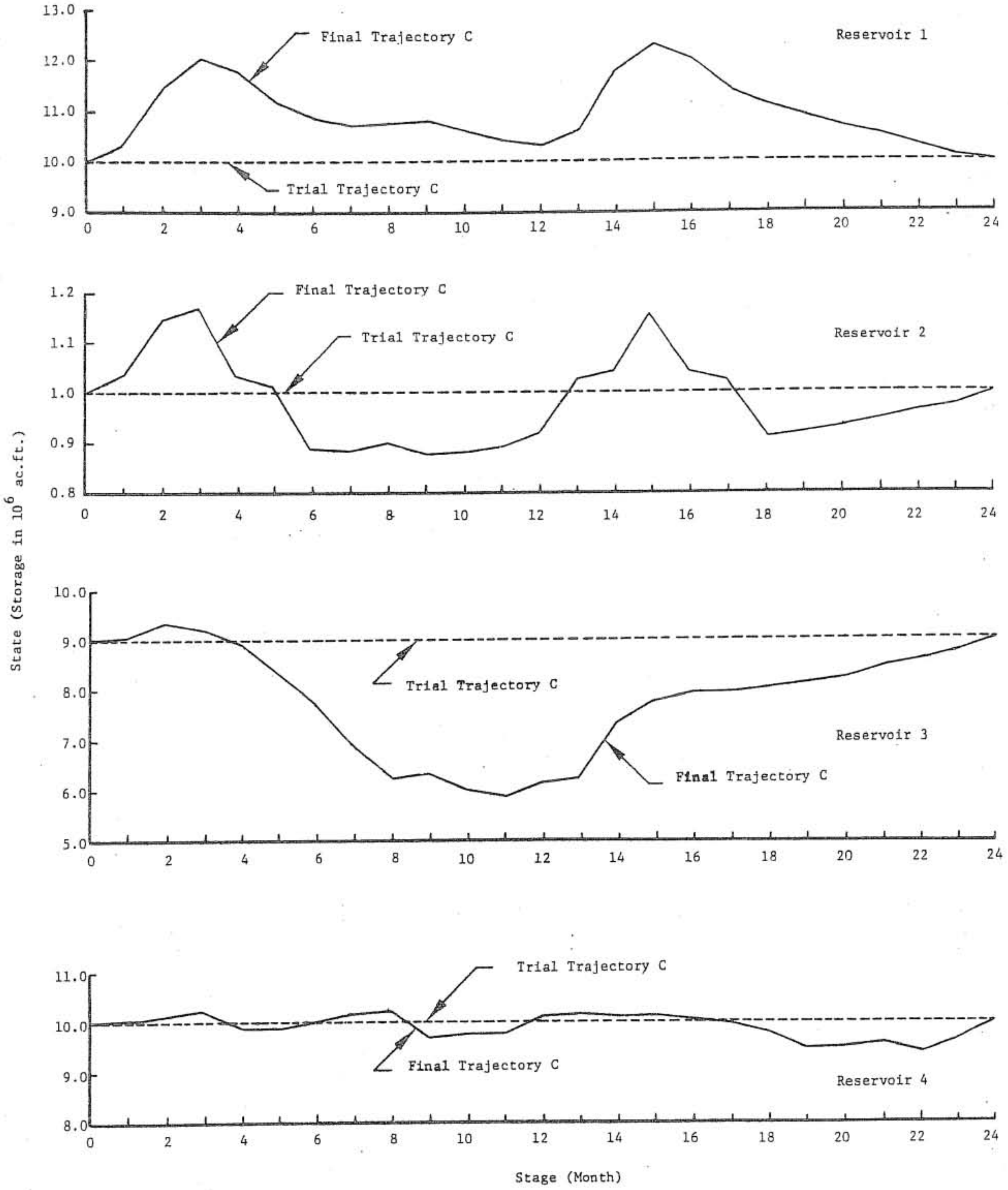


FIGURE 20. Trial and Final Trajectories C for System in Figure 13 During 24 Months of Operation

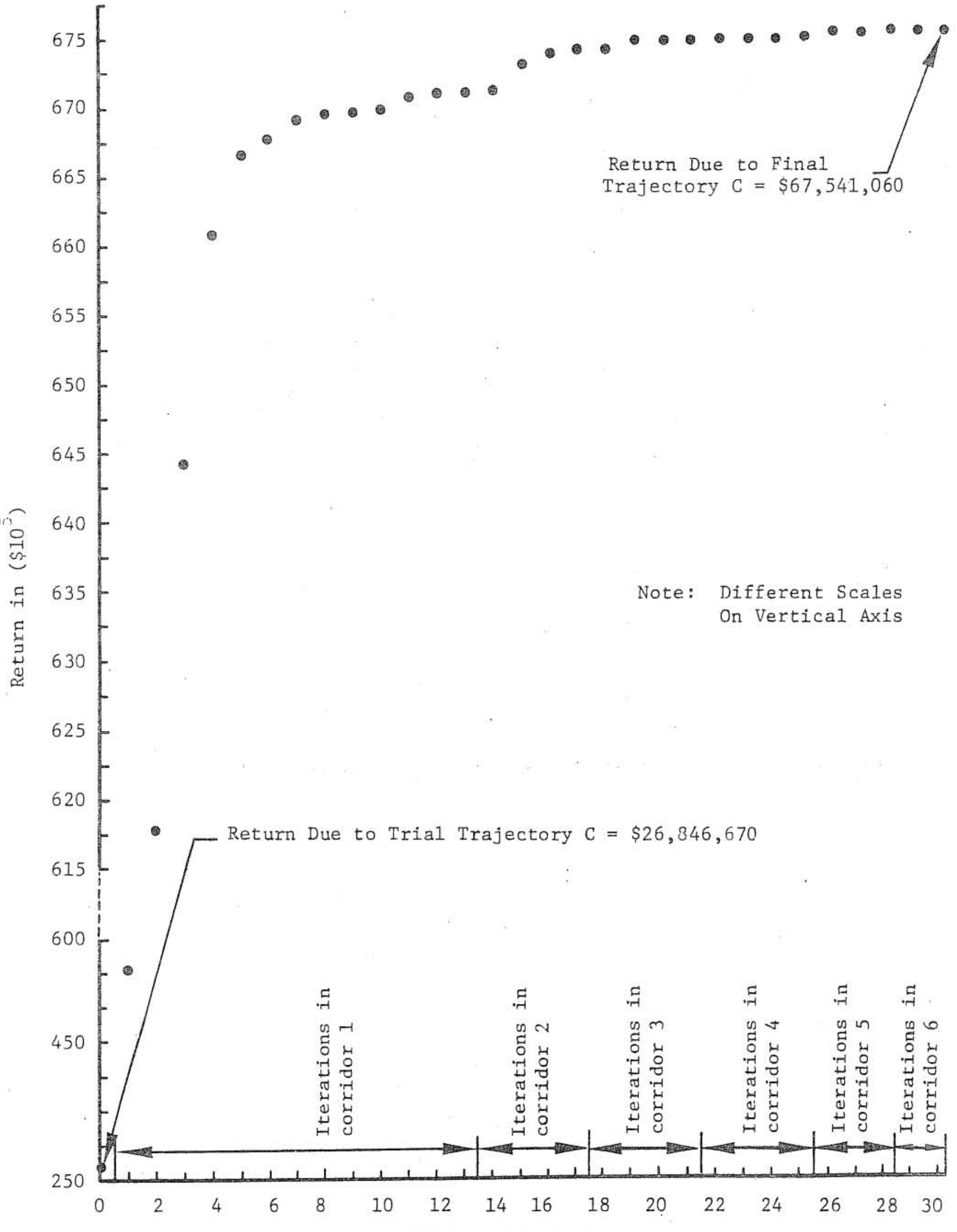


FIGURE 21. Return vs. Iteration Number for 24 Operating Periods of System in Figure 13 Using Trial Trajectory C

per year more return than A. The differences between these two final trajectories may be attributed to the choice of the trial trajectory and the state sub-domains. In both cases a state sub-domain computed from equations (67) was used for the trial trajectory. It may be more effecient computationally to determine the state sub-domain as a function of stage and iteration number as suggested by Mayne [1966] and Jacobson [1968a]. See Section 6-2-1 for discussion regarding computations of the state sub-domain.

Table 9 shows that all final trajectories satisfied the annual target irrigation and power demands, and that there was a surplus power of about 27.2%, 29.1%, and 28.5% with respect to the demand for final trajectories A, B, and C respectively. Tables A-1, 2, 3, 4, 5, 6 in Apendix A show that monthly irrigation and power demands are satisfied. In addition, surplus power was sold as dump energy (priced at 1.5 mills per kilowatt hour) and, therefore, the total return would have been more had the annual energy demand been set higher than 2.0 X 10^9 kw hr. This conclusion suggests that the parameters and demands set in this analysis do not represent the optimum capability of the system. Therefore, in further analysis, the response of the system to higher demands must be evaluated.

TABLE 9. Production and Returns Due to Different Trajectories for System in Figure 13

Trajectory	Sum of Water for Irrigation (10^6 ac.ft)	Sum of Returns from Irrigation (\$$10^6$)	Sum of Power Generated (Mw hr)	Sum of Returns from Power (\$$10^6$)	Sum of Returns (\$$10^6$)	Time Used on the Central Processing Unit (CPU) of IBM 360/75
Trial A	3.2367	0.0	1583831	12.952	12.952	17.7 min
Final A	4.0000	18.916	2544013	14.816	33.732	
Trial B	2.4172	0.0	1804347	13.423	13.423	16.9 min
Final B	4.0000	18.916	2581903	14.872	33.789	
Trial C	4.8344	0.0	3608694	26.847	26.847	40.7 min
Final C	8.0000	37.831	5139859	29.710	67.541	

6. CONCLUDING REMARKS

6-1. Summary and Conclusions

This study has been primarily aimed at the presentation of an approach (based on existing optimization techniques) which is suitable for analysis of operation policies of multiple-purpose and multiple-unit water resources systems. While investigating the characteristics of these systems, it was concluded that dynamic programming is applicable for the optimization of their operating policies. However, further investigation of standard dynamic programming revealed a dimensionality problem which limits its application to low-order systems. Attempts to curb the dimensionality of the standard dynamic programming led to an approach which originally had been inspired by Larson's [1968] state increment dynamic programming, but whose theoretical justification was found to be that of differential dynamic programming presented by Mayne [1966] and Jacboson [1968a, b, c]. The proposed DDDP approach which deals with the optimization of trajectories, was applied to two four-dimensional reservoir systems (a simplified system, and Clearwater River System) whose solutions could not be obtained using standard dynamic programming with the available computer facilities and budget allocated for computer time. In the analysis of these systems it was discovered that the invertibility of system equations in water resources can be used to reduce computation time.

For the first system the solution by the proposed approach with three different trial trajectories was found to be the exact solutions obtained by two other techniques. However, it was concluded that this exactness in results is due to the choice of the trial trajectory and the state sub-domains. The selection of a different set of state sub-domains proved to provide near optimum results.

The difficulties in the choice of the trial trajectory and the state sub-domains became more apparent in the solution of the second system which is much more realistic than the previous one, but which must still be considered a simplified system with respect to actual systems. It was observed that the choice of two different trial trajectories led to two different but close final trajectories. The total return of final trajectory A is about 0.17% less than that of final trajectory B. This difference was attributed to the choice of the trial trajectories and the state sub-domains which remained the same for both trial trajectories.

Extensions of the time horizons of the above systems, without any changes in demand and objective function, from 12 to 24 time periods produced operating policies in each of the two sets of 12 time periods very similar to those obtained in the analysis for a time horizon of only 12 time periods. The results of the analysis of the Clearwater River system indicate that it is capable of answering higher demands than are set in the analysis.

It must be mentioned that the solution of these systems became possible by reducing them to a class of problems generally referred to as two-point boundary-valued problems. This approach naturally excludes many other solutions. However, in water resources analysis many of these solutions are not considered operationally feasible. For example, the solution of cases where the reservoirs are full or empty at the beginning and end of a certain time period may never be operationally feasible and therefore may be excluded from the analysis. There are cases due to system constraints, such as flood control pool, recreational, and power head requirements which make the solution of a general problem wasteful, i.e., a problem which provides answers for every initial and final state.

6-2. Recommendations for Future Studies

The following recommendations and suggestions are the direct result of the realization that this study has not investigated all of the possible improvements that can be made to the proposed approach and its applications to actual systems.

6-2-1. Regarding the Approach

In the application of the proposed approach two major difficulties were observed which may be improved as follows.

6-2-1-1. Choice of Trial Trajectory

The choice of the trial taajectory can reduce the computing time noticeably, simply because the closer the trial trajectory is to the optimal trajectory, the fewer are the iterations that are required to arrive at or near the optimal trajectory. Larson's [1968] successive approximation dynamic programming algorithm based on Bellman's [1957, 1961] work may be employed to arrive at a trial trajectory. In fact, in applying this technique (successive approximation dynamic programming) to the problem presented in Section 5-1 a near optimal trajectory was obtained in this study. Another approach in setting up the trial trajectory is the use of engineering judgment. This approach is particularly applicable to water resources systems where most of them are presently being operated on the basis of such engineering judgment.

6-2-1-2. Choice of the State Sub-Domains

The choice of a set of state sub-domains at the start of each iteration as a function of stages may be considered as the most important single factor in the rate of convergence. In selection of these sub-domains works of Mayne [1966], Jacobson [1968a], and Jacobson and Mayne [1970] may be employed. These works form the Hamiltonian of the return function and optimize it by calculus of variation to estimate the best changes in decision and states as functions of time in the neighborhood of the trial trajectory. Both of these algorithms require that the second derivative of the Hamiltonian with respect to the decision variable be negative-definite for a maximization problem. This requirement may be difficult to fulfill in water resources systems where some objective functions, such as flood control return functions, are not continuously differentiable. However, approximation of these functions by a proper order polynomial function which satisfies the above requirement may generate a close approximation to the optimal $\delta u(t)$ and $\delta s(t)$ in equations (21) and (22). Mayne's algorithm, which is an approximation to Jacobson's, exhibits one step convergence for control problems with linear equations and quadratic performance criterion. In this context, Mayne's algorithm is found to be considerably more efficient than Jacobson's algorithm [Jacobson, 1968a]. Therefore, perhaps an approximation of the objective function by a second order polynomial is suitable for an approximation of the state sub-domains.

6-2-2. Choice of Return Functions

In the course of this study it was observed that the compatibility of return functions play an important role in the final results. Assume that storage in a reservoir provides recreation, and outflow from it generates hydropower and may cause flood damage. If the return and damage functions are assumed without substantial amount of economic and engineering studies, the result of optimization may not show any logical conclusion. For example, if an unjustifiably high penalty is assumed for the hydropower shortage, the optimizing model selects decisions which may not be realistic as far as flood control and recreation projects are concerned. Therefore, one of the prime objectives of any future study should be the development of realistic return functions for the water resources system under study.

6-2-3. Regarding the Hydrology

In this study the economic response of the physical system was measured in terms of the mean monthly flows. However, in a more realistic analysis, statistical and stochastic models describing the joint probability distributions of the different river flows must be formulated and incorporated into the recursive equation of dynamic programming. Perhaps one should be warned of the sharp increase in the computer time requirements with this type of data.

6-2-4. Regarding the System Optimization

In a complete optimization of the water resources system, the operating phase must be optimized in conjunction with the planning and allocation phases. The proposed optimization procedure may be repeated for different combinations of planning and allocation decisions to form a maximum return response surface. Then, this surface may be optimized by a static optimization technique such as the steepest ascent in an attempt to obtain the most suitable combination of decisions, i.e., planning, allocation, and operation.

7. REFERENCES

Anderson, T. W., An Introduction to Multivariate Statistical Analysis, John Wiley & Sons, Inc., New York, 1966.

Bellman, R., An introduction to the theory of dynamic programming, Rand Report R-245, The Rand Corporation, Santa Monica, California, June, 1953.

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

Bellman, R., Adaptive Control Processes, Princeton University Press, Princeton, New Jersey, 1961.

Bellman, R., A new type of approximation leading to reduction of dimensionality in control processes, J. Mathematical Analysis and Applications, 27, 454-459, 1969.

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

Bellman, R., and R. Kalaba, Reduction of dimensionality, dynamic programming and control processes, J. Basic Engineering, Trans. Am. Soc. Mech. Engrs., 83, 82-84, March, 1961.

Blaney, H. F., and W. D. Criddle, Determining water requirements in irrigated areas from climatological and irrigation data, U.S. Soil Conservation Service Technical Publication 96, 1952.

Bryson, A. E., and Yu-Chi Ho, Applied Optimal Control, Blaisdell Publishing Co., Waltham, Mass., 131-141, 1969.

Chow, V. T., and D. D. Meredith, Water resources systems analysis: Part II. Annotated bibliography on programming techniques, Civil Engineering Studies, Hydraulic Engineering Series No. 20, University of Illinois, Urbana, Illinois, July, 1969a.

Chow, V. T., and D. D. Meredith, Water resources systems analysis: Part IV. Review of programming techniques, Civil Engineering Studies, Hydraulic Engineering Series No. 22, University of Illinois, Urbana, Illinois, July, 1969b.

Eveleigh, V. W., Adaptive Control and Optimization Techniques, McGraw-Hill, New York, 1967.

Fel'dbaum, A. A., Optimal Control Systems, Academic Press, New York, 1965.

Fletcher, R., and M.J.D. Powell, A rapidly convergent descent method for minimization, The Computer Journal, VI, 163-168, 1963.

Fletcher, R., and C. M. Reeves, Function minimization by conjugate gradients, The Computer Journal, VII, 149-154, 1964.

Gablinger, M., and D. P. Loucks, Markov models for flow regulation, J. Hydraulics Div., Am. Soc. Civil Engrs., 96(HY1), 165-181, January, 1970.

Hall, W. A., R. C. Garboe, W. W.-G. Yeh, and A. J. Askew, Optimum firm power output from a two reservoir system by incremental dynamic programming, Contribution No. 130, Water Resources Center, University of California, Los Angeles, October, 1969.

Jacobson, D. H., Second-order and second-variation methods for determining optimal control: A comparative study using differential dynamic programming, International Journal of Control, 7(2), 175-196, 1968a.

Jacobson, D. H., New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach, J. Optimization Theory and Applications, 2(6), 1968b.

Jacobson, D. H., Differential dynamic programming methods for solving bang-bang control problems, IEEE Trans. Automatic Control, AC-13(6), 661-675, December, 1968c.

Jacobson, D. H., and D. Mayne, Differential Dynamic Programming, American Elseveir Pub. Co., Inc., New York, 1970.

Korsak, A. J., and R. E. Larson, A dynamic programming successive approximation technique with convergence proofs - Part II: Convergence proofs, IFAC Automatica, 6(2), 261-270, March, 1970.

Larson, R. E., State Increment Dynamic Programming, American Elsevier Publiching Co., Inc., New York, 1968.

Larson, R. E., and W. G. Keckler, Applications of dynamic programming to water resources problems, Presented at IFAC Haifa Conference on Computer Control of Natural Resources and Public Utilities, Haifa, Israel, September, 1967.

Larson, R. E., and W. G. Keckler, Applications of dynamic programming to control of water resources systems, Automatics, 5, 15-26, 1969.

Lee, S. E., Dynamic programming, quasilinearization and the dimensionality difficulty, J. Mathematical Analysis and Applications, 27, 303-322, 1969.

Maass, A., M. M. Hufschmidt, R. Dorfman, H. E. Thomas, S. A. Marglin, and G. M. Fair, Design of Water Resources Systems, Harvard University Press, Cambridge, Massachusetts, 1962.

Mayne, D., A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems, International Journal of Control, 3(1), 85-95, 1966.

Roberts, S. M., Dynamic Programming in Chemical Engineering and Process Control, Academic Press, New York, 1964.

Wong, P. J., and D. G. Luenberger, Reducing the memory requirement of dynamic programming, Operations Research, 16, 1115-1125, November-December, 1968.

APPENDIX A. COMPUTER PROGRAMS AND DESCRIPTION OF PROGRAM INPUT AND OUTPUT

A-1. Notation Used to Input Data for Program to Analyze Clearwater River System

IUNIT	= Low-speed storage unit used to store computed data of the previous stages.
NN	= Number of stages in analysis.
ITER	= Maximum number of iterations to be performed in each corridor width.
NCOR	= Number of corridors to be considered in the analysis.
IFLAG	= A flag which specified whether stochastic dynamic programming or deterministic dynamic programming should be used. If IFLAG = 1, then analyze stochastically. If IFLAG = 2, then analyze deterministically. Note: Since the subroutine for stochastic dynamic programming is not included in the following program, this flag should always be set equal to 2 in the input data.
IPRINT	= A flag specifying whether or not a listing of the input data is required. If IPRINT = 0, then no listing of input data will be produced. If IPRINT = 1, then the input data will be listed.
IPLOT	= A flag specifying whether or not a plot of trajectory is required. If IPLOT = 0, then no plot will be produced. If IPLOT = 1, then plots will be produced.
ALPHA	= A constant equal to 0.3 specifying the percentage of irrigation water return.
CONST	= A negative constant, such as -1000000, replacing all the variables to be calculated at the start of computations.
CR(I)	= Capacity of reservoir I, I = 1,2,3,4, (10^6 ac.ft.).

CPP1	= Capacity of power plant 1 (Mw).
CPP2	= Capacity of power plant 2 (Mw).
TIRR	= Target irrigation water demand (10^6 ac.ft.).
TPWR	= Target power demand (Mw hr.).
BPUFE	= Return per unit of firm energy (\$/Mw hr.).
CPUDEF	= Cost per unit of deficit energy (\$/Mw hr.).
BPUDUP	= Return per unit of dump energy (\$/Mw hr.).
SI(I)	= Initial storage of reservoir I, I = 1,2,3,4 (10^6 ac.ft.).
SF(I)	= Final storage of reservoir I, I = 1,2,3,4 (10^6 ac.ft.).
DS(I)	= One half of the initial corridor's width for reservoir I, I = 1,2,3,4 (10^6 ac.ft.).
S(IS,N)	= Initial trajectory of reservoir IS, IS = 1,2,3,4 at stage N, N = 1,2, ..., NN (10^6 ac.ft.).
EY(I,N)	= Expected inflow of stream I, I = 1,2, ..., (NN-1) (10^6 ac.ft.).
DMIN(IS,N)	= Lower bound on decision IS, IS = 1,2,3,4 during month N, N = 1,2, ..., 12 (10^6 ac.ft.).
SMIN(IS,N)	= Lower bound on storage IS, IS = 1,2,3,4 at the beginning of month N, N = 1,2, ..., 12 (10^6 ac.ft.).
SMAX(IS,N)	= Upper bound on storage IS, IS = 1,2,3,4 at the beginning of month N, N = 1,2, ..., 12 (10^6 ac.ft.).
PPR(MO)	= Percentage of the power demand during month MO, MO = 1,2, ..., 12.
PIR(MO)	= Percentage of the irrigation water demand during month MO, MO = 1, 2, ..., 12.
A(1,I)	= Coefficients of the polynomial describing the effective power head-storage relationship for reservoir 1 in Table 7.

A-2. FORTRAN IV Program used to Analyze the Clearwater River System

```
C     MAIN
      COMMON NNS(4,121),SI(4),SF(4),SL(4,121,3),S1NM1(81),S2NM1(81)
     1,S3NM1(81),S4NM1(81),FN(81),FNM1(81),D5N(81),D5NM1(2,81),FY(5
     2,120),YL(5,12,3),DMIN(4,12),DMAX(4,12),DIR(12),DPR(12),BETA(1
     32),S(4,121),DS(4),SMIN(4,13),SMAX(4,13),A(1,6),D(5,121),PPR(1
     42),REPWR(12),PIR(12),RETURN(13),CR(4),ALPHA,CONST,CPP1,CPP2,H
     51MAX,WCPP2,BPUFE,CPUDFF,BPUDUP,UPIR,PRJO(243)
      CALL INPUT(IUNIT,NN,ITER,NPR,NCOR,NNM1,IFLAG,NNP1,IPLOT)
C     CALCULATE RETURN DUE TO NOMINAL TRAJECTORY
      IT=0
      IC=1
      CALL DETAIL(NN,FLAG2,IT,IC,IFLAG)
      IF (FLAG2.EQ.1.)GO TO 114
      PRINT 2004,IT,IC
      DO 101 N=1,NNM1
      PRINT 2005,N,S(1,N),S(2,N),S(3,N),S(4,N),D(1,N),D(2,N),D(3,N)
     1,D(4,N),D(5,N)
  101 CONTINUE
      PRINT 2005,NN,S(1,NN),S(2,NN),S(3,NN),S(4,NN)
      PRINT 2006,RETURN(IC)
      FNITM1=RETURN(IC)
      IF(IPLOT.LT.1)GO TO 102
C     PLOT STATE VS. STAGE
      CALL PLOT1(NN,IT,WIDTH,IC,CR,S)
  102 CONTINUE
C     START ITERATION
      DO 112 IC=2,NCOR
      DO 109 IT=1,ITER
C     SET UP CORRIDORS
      CALL STATE(S,SMIN,SMAX,DS,SL,NNS,NN)
      PRINT 2007,(DS(I),I=1,4)
C     OPTIMIZE INSIDE CORRIDORS
      GO TO(103,104),IFLAG
C     OPTIMIZE STOCHASTICALLY
  103 CONTINUE
      GO TO 105
C     OPTIMIZE DETERMINISTICALLY
  104 CALL DPD(NN,IUNIT)
  105 CONTINUE
C     TRACE OPTIMAL TRAJ. INSIDE CORRIDORS
      CALL TRACE(NN,IUNIT,MNN,NNM1,NNP1,IFLAG)
C     PRINT THE NEW TRAJECTORY
      PRINT 2004,IT,IC
      DO 106 N=1,NNM1
      PRINT 2005,N,S(1,N),S(2,N),S(3,N),S(4,N),D(1,N),D(2,N),D(3,N)
     1,D(4,N),D(5,N)
  106 CONTINUE
      PRINT 2005,NN,S(1,NN),S(2,NN),S(3,NN),S(4,NN)
      PRINT 2006,FN(MNN)
      IF(FN(MNN).LE.FNITM1)GO TO 110
      FNITM1=FN(MNN)
  109 CONTINUE
  110 RETURN (IC)=FN(MNN)
      FNITM1=FN(MNN)
      DO 111 IS=1,4
  111 DS(IS)=DS(IS)*.7
  112 CONTINUE
```

```
  113 CONTINUE
C     CALCULATE & LIST DETAIL LIST OF ACTIVITIES
      CALL DETAIL(NN,FLAG2,IT,IC,IFLAG)
      IF(IPLOT.LT.1)GO TO 114
C     PLOT STATE VS. STAGE
      CALL PLOT1(NN,IT,WIDTH,IC,CR,S)
  114 CONTINUE
 2004 FORMAT(1H1,//,20X,'TRAJECTORY AFTER ',I2,' ITERATIONS IN CORR
     1IDOR ',I2,//,9X,'N',7X,'S1',8X,'S2',8X,'S3',8X,'S4',8X,'D1',8
     2X,'D2',8X,'D3',8X,'D4',8X,'D5',//)
 2005 FORMAT(1X,I9,9F10.4)
 2006 FORMAT(1X,'RETURN ($1000)=',F15.2)
 2007 FORMAT(///,1X,'DS(1)=',F7.4,10X,'DS(2)=',F7.4,10X,'DS(3)=',F7
     1.4,10X,'DS(4)=',F7.4)
 2008 FORMAT(1H1)
      PRINT 2008
      STOP
      END
```

```
      SUBROUTINE INPUT(IUNIT,NN,ITER,NPR,NCOR,NNM1,IFLAG,NNP1,
     1IPLOT)
      COMMON NNS(4,121),SI(4),SF(4),SL(4,121,3),S1NM1(81),S2NM1(81)
     1,S3NM1(81),S4NM1(81),FN(81),FNM1(81),D5N(81),D5NM1(2,81),FY(5
     2,120),YL(5,12,3),DMIN(4,12),DMAX(4,12),DIR(12),DPR(12),BETA(1
     32),S(4,121),DS(4),SMIN(4,13),SMAX(4,13),A(1,6),D(5,121),PPR(1
     42),REPWR(12),PIR(12),RETURN(13),CR(4),ALPHA,CONST,CPP1,CPP2,H
     51MAX,WCPP2,BPUFE,CPUDEF,BPUDUP,UPIR,PRJO(243)
C     READ INPUT DATA
      READ 1000,IUNIT,NN,ITER,NCOR,NPR,IFLAG,IPRINT,IPLOT
C     IF IFLAG=1 ANALYZE STOCHASTICALLY
C     IF IFLAG=2 ANALYZE DETERMINISTICALLY
      NNP1=NN+1
      NNM1=NN-1
      READ 1002,ALPHA,CONST
      READ 1002,(CR(I),I=1,4),CPP1,CPP2,TIRR,TPWR
      READ 1002,BPUFF,CPUDEF,BPUDUP
      READ 1002,(SI(I),I=1,4)
      READ 1002,(SF(I),I=1,4)
      READ 1002,(DS(I),I=1,4)
      DO 93 IS=1,4
   93 READ 1005,(S(IS,N),N=1,NN)
      GO TO(94,96),IFLAG
   94 DO 95 I=1,5
      READ 1004,(FY(I,MO),MO=1,12)
      DO 95 IP=1,NPR
      READ 1004,(YL(I,MO,IP),MO=1,12)
   95 CONTINUE
```

```
      NIP=NPR**1
      READ 1004,(PRJO(IP),IP=1,NIP)
      GO TO 98
   96 DO 97 I=1,5
   97 READ 1004,(FY(I,N),N=1,NNM1)
   98 DO 99 IS=1,4
      READ 1003,(DMIN(IS,N),N=1,12)
      READ 1003,(SMIN(IS,N),N=1,13)
   99 READ 1003,(SMAX(IS,N),N=1,13)
      READ 1003,(PPR(MO),MO=1,12)
      READ 1003,(PIR(MO),MO=1,12)
      READ 1001,(A(1,I),I=1,6)
C     CALCULATE MONTHLY IRRIGATION(10**6 A.F./MO) AND
C     POWER (MW-HR/MO) DEMANDS
      DO 100 MO=1,12
      DIR(MO)=TIRR*PIR(MO)/100.
  100 DPR(MO)=TPWR*PPR(MO)/100.
C     CALCULATE BENEFIT FROM POWER IF ALL DEMANDS WERE SATISFIED
      DO 101 MO=1,12
  101 REPWR(MO)=DPR(MO)*BPUFF
C     CALCULATE BETA
      DIRMAX=0.
      DO 102 MO=1,12
      IF (DIR(MO).LE.DIRMAX)GO TO 102
      DIRMAX=DIR(MO)
  102 CONTINUE
      DO 103 MO=1,12
  103 BETA(MO)=DIR(MO)/DIRMAX
C     CALCULATE H1MAX=MAX. HEAD (FT.) OF RES. 1
      CR1=CR(1)
      H1MAX=A(1,1)+A(1,2)*CR1+A(1,3)*(CR1**2)+A(1,4)*(CR1**3)+A(1,5
     1)*(CR1**4)+A(1,6)*(CR1**5)
C     CALCULATE WCPP2=WATER CAPACITY OF PP2 IN (10**6 AF/MO)
      WCPP2=.005073*CPP2
C     CALCULATE PRICE OF IRRIGATION WATER, UPIR, IN ($/10**6 AF)
      IF(TIRR.LE.5.5)GO TO 104
      UPIR=4500000.
      GO TO 105
  104 UPIR=(.05278*(TIRR**2)-.6412*TIRR+6.44924)*1000000.
  105 CONTINUE
      IF(IPRINT.EQ.0)GO TO 999
C     PRINT INPUT DATA
      PRINT 2000,NN,ITER,ALPHA,CONST
      PRINT 2001
      PRINT 2002,(CR(I),I=1,4),CPP1,CPP2,TIRR,TPWR
      PRINT 2003,BPUFF,CPUDFF,BPUDUP
      PRINT 2004,(SI(I),SF(I),I=1,4)
      PRINT 2005,(DS(I),I=1,4)
      DO 200 IS=1,4
      PRINT 2006,IS,(DMIN(IS,N),N=1,12)
      PRINT 2007,IS,(DMAX(IS,N),N=1,12)
      PRINT 2008,IS,(S(IS,N),N=1,NN)
      PRINT 2009,IS,(SMIN(IS,N),N=1,13)
  200 PRINT 2010,IS,(SMAX(IS,N),N=1,13)
      GO TO(201,203),IFLAG
  201 DO 202 I=1,5
      PRINT 2011,I,(FY(I,MO),MO=1,12)
      DO 202 IP=1,NPR
      PRINT 2018,I,IP,(YL(I,MO,IP),MO=1,12)
  202 CONTINUE
      PRINT 2019,(PRJO(IP),IP=1,NIP)
      GO TO 205
  203 DO 204 I=1,5
  204 PRINT 2011,I,(FY(I,N),N=1,NNM1)
  205 PRINT 2012,(PPR(MO),MO=1,12)
      PRINT 2013,(DPR(MO),MO=1,12)
      PRINT 2017,(REPWR(MO),MO=1,12)
      PRINT 2014,(DIR(MO),MO=1,12)
      PRINT 2015,(BETA(MO),MO=1,12)
      PRINT 2016,H1MAX,WCPP2,UPIR
 1000 FORMAT(16I5)
 1001 FORMAT(3F20.8)
 1002 FORMAT(8F10.0)
 1003 FORMAT(16F5.0)
 1004 FORMAT(8X,12F6.4)
 1005 FORMAT(13F6.4)
 2000 FORMAT(1H1,  60X,'INPUT DATA',/ ,1X,'NO. OF STAGES=',I5,5X,'N
     1O. OF ITER=',I3,5X,'ALPHA=',F4.2,5X,'CONST=',F10.0,/)
 2001 FORMAT(1X,'ALL VOLUME UNITS IN 10**6 A.F.',/ )
 2002 FORMAT(1X,'PARAMETERS OF SYSTEM:  ','CR1(10**6 A.F.)=',F5.2,2
     1X,'CR2(10**6 A.F.)=',F5.2,2X,'CR3(10**6 A.F.)=',F5.2,2X,'CR4(
     210**6 A.F.)=',F5.2,//,25X,'CPP1(MW)=',F6.1,2X,'CPP2(MW)=',F6.
     31,2X,'TIRR(10**6 A.F.)=',F6.1,2X,'TPWR(MW-HR)=',F10.1,//)
 2003 FORMAT(1X,'PRICE OF ENERGY:',3X,'FIRM ENERGY($/MW-HR)=',F5.2,
     13X,'DEFICIT ENERGY($/MW-HR)=',F5.2,3X,'DUMP ENERGY($/MW-HR)='
     2,F5.2,/)
 2004 FORMAT(1X,'SI(1)=',F5.2,4X,'SF(1)=',F5.2,4X,'SI(2)=',F5.2,4X,
     1'SF(2)=',F5.2,4X,'SI(3)=',F5.2,4X,'SF(3)=',F5.2,4X,'SI(4)=',F
     25.2,4X,'SF(4)=',F5.2)
 2005 FORMAT(1X,'DELS(1)=',F5.2,4X,'DELS(2)=',F5.2,4X,'DELS(3)=',F5
     1.2,4X,'DELS(4)=',F5.2,//)
 2006 FORMAT(1X,'DMIN(',I1,',MO)=',13F8.3)
 2007 FORMAT(1X,'DMAX(',I1,',MO)=',13F8.3)
 2008 FORMAT(1X,'S(',I1,',N)=',4X,13F8.3)
 2009 FORMAT(1X,'SMIN(',I1,',N)=',1X,13F8.3)
 2010 FORMAT(1X,'SMAX(',I1,',N)=',1X,13F8.3,//)
 2011 FORMAT(1X,'Y(',I1,',MO)=',12F9.4)
 2012 FORMAT(/,1X,'PPR(MO)=',5X,12F9.1)
 2013 FORMAT(1X,'DPR(MW-HR/MO)',F8.0,11F9.0)
 2014 FORMAT(1X,'DIR(M-AF/MO)',12F9.4)
 2015 FORMAT(1X,'BETA(MO)=',3X,12F9.4)
 2016 FORMAT(/ ,1X,'H1MAX (FT)=',F6.1,10X,'WCPP2 (10**6 AF/MO)=',F5
     1.2,10X,'UPIR ($/10**6 AF)=',F8.0)
 2017 FORMAT(1X,'REPWR ($/MO)',12F9.0,/ )
 2018 FORMAT(1X,'YL(',I1,',MO,',I1,')',1X,12F9.4)
 2019 FORMAT(1X,'JOINT PROB.',18F6.4)
  999 CONTINUE
      RETURN
      END
      SUBROUTINE DETAIL(NN,FLAG2,IT,IC,IFLAG)
      COMMON NNS(4,121),SI(4),SF(4),SL(4,121,3),S1NM1(81),S2NM1(81)
     1,S3NM1(81),S4NM1(81),FN(81),FNM1(81),D5N(81),D5NM1(2,81),EY(5
     2,120),YL(5,12,3),DMIN(4,12),DMAX(4,12),DIR(12),DPR(12),BETA(1
     32),S(4,121),DS(4),SMIN(4,13),SMAX(4,13),A(1,6),D(5,121),PPR(1
     42),RFPWR(12),PIR(12),RETURN(13),CR(4),ALPHA,CONST,CPP1,CPP2,H
     51MAX,WCPP2,BPUFE,CPUDEF,BPUDUP,UPIR,PRJO(243)
      DIMENSION FLAG1(12)
      PRINT 1000,IT,IC
      FLAG2=0.
      MO=0
      SBPRYR=0.
      SBPRHR=0.
      SBIRYR=0.
      SBIRHR=0.
      SWIRYR=0.
      SWIRHR=0.
      SPRYR=0.
      SPRHR=0.
      SBPYR=0.
      SBHR=0.
      DO 99 MO=1,12
   99 FLAG1(MO)=0.
      DO 110 N=2,NN
      NM1=N-1
      MO=MO+1
      IF(MO.GT.12)MO=1
      IF(IFLAG.EQ.1)NM1=MO
      D(1,N-1)=S(1,N-1)+EY(1,NM1)-S(1,N)
      IDE=1
      IF(D(1,N-1).LT.DMIN(1,MO))GO TO 111
      D(2,N-1)=S(2,N-1)+EY(2,NM1)-S(2,N)
      IDE=2
      IF(D(2,N-1).LT.DMIN(2,MO))GO TO 111
      D(3,N-1)=S(3,N-1)+EY(3,NM1)-S(3,N)
      IDE=3
      IF(D(3,N-1).LT.DMIN(3,MO))GO TO 111
      FLO2=D(1,N-1)+D(2,N-1)+EY(4,NM1)
      IF(FLO2.GE.DIR(MO))GO TO 100
      D(5,N-1)=FLO2
      DEFIR=DIR(MO)-FLO2
      PIRSH=(DEFIR/DIR(MO))*100.
      GO TO 101
  100 D(5,N-1)=DIR(MO)
      DEFIR=0.
      PIRSH=0.
  101 CONTINUE
      IF(N.GT.2)GO TO 102
      RUNOFF=0.
      GO TO 103
  102 RUNOFF=ALPHA*D(5,N-2)
  103 D(4,N-1)=S(4,N-1)-S(4,N)+FLO2-D(5,N-1)+RUNOFF
      IDE=4
      IF(D(4,N-1).LT.DMIN(4,MO))GO TO 111
      FLO4=D(3,N-1)+D(4,N-1)+EY(5,NM1)
      CALL POWER(S(1,N-1),S(1,N),D(1,N-1),FLO4,MO,CPP1,H1MAX,WCPP2,
     1CPUDEF,BPUDUP,BPR,TOTP,A,DPR,RFPWR,PPRSH,PPRSP)
      IF(PIRSH.GT.80.)GO TO 104
      CALL IRRIG(PIRSH,BIR,MO,DIR,BETA,UPIR)
      GO TO 105
  104 BIR=0.
      FLAG1(MO)=1.
  105 CONTINUE
      DO 106 M=1,MO
      IF(FLAG1(M).EQ.1.)GO TO 107
  106 CONTINUE
      SBPYR=SBPYR+BIR+BPR
      SBIRYR=SBIRYR+BIR
      GO TO 108
  107 SBPYR=SBPYR+BPR-SBIRYR
      SBIRYR=0.
  108 CONTINUE
      SWIRYR=SWIRYR+D(5,N-1)
      SPRYR=SPRYR+TOTP
      SBPRYR=SBPRYR+BPR
      PRINT 1001,N,MO,D(5,N-1),PIRSH,SWIRYR,BIR,SBIRYR,TOTP,PPRSP,P
     1PRSH,SPRYR,BPR,SBPRYR,SBPYR
      IF(MO.LT.12)GO TO 110
      DO 109 M=1,12
  109 FLAG1(M)=0.
      SBIRHR=SBIRHR+SBIRYR
      SBPRHR=SBPRHR+SBPRYR
      SWIRHR=SWIRHR+SWIRYR
      SPRHR=SPRHR+SPRYR
      SBHR=SBHR+SBPYR
      SBIRYR=0.
      SBPRYR=0.
      SWIRYR=0.
      SPRYR=0.
      SBPYR=0.
  110 CONTINUE
      RETURN(IC)=SBHR
      PRINT 1003,SWIRHR,SBIRHR
      PRINT 1004,SPRHR,SBPRHR
      GO TO 112
  111 PRINT 1002,N,MO,IDE
      FLAG2=1.
 1000 FORMAT(1H1,//,10X,'SUMMARY OF IRRIGATION, PWER, AND FLOOD ACT
     1IVITIES (TRAJ. OF ITER ',I2,' IN CORR. ',I2,')',//,2X,'N',2X,
     2'MO',3X,'D5',3X,'PIRSH',1X,'SWIR/YR',5X,'BIR',4X,'SBIR/YR',5X
     3,'TOTP',3X,'PPRSP',1X,'PPRSH',2X,'SPR/YR',7X,'BPR',4X,'SBPR/Y
     4R'               ,3X,'STB/YR',/,8X,'(M-AF)',2X,'(%)',2X,'(M-AF)
     5',5X,'(K-$)',4X,'(K-$)',3X,'(MW-HR)',3X,'(%)',3X,'(%)',3X,'(M
     6W-HR)',5X,'(K-$)',4X,'(K-$)',             5X,'(K-$)',/)
 1001 FORMAT(1X,I3,1X,I2,F7.4,F6.2,F8.4,3F10.1, F6.2,5F10.1)
 1002 FORMAT(1X,'AT N=',I3,' AND MO=',I2,' D',I1,' IS LT. DMIN')
 1003 FORMAT(//,5X,'SWIRHR (M-AF)=',F10.4,10X,'SBIRHR (K-$)=',
     1F10.1)
 1004 FORMAT(/,5X,'SPRHR (MW-HR)=',F10.1,10X,'SBPRHR (K-$)=',F10.1)
  112 CONTINUE
      RETURN
      END
      SUBROUTINE DPD(NN,IUNIT)
C     FORWARD DP APPLIED TO CLEAR WATER BASIN
      COMMON NNS(4,121),SI(4),SF(4),SL(4,121,3),S1NM1(81),S2NM1(81)
     1,S3NM1(81),S4NM1(81),FN(81),FNM1(81),D5N(81),D5NM1(2,81),EY(5
     2,120),YL(5,12,3),DMIN(4,12),DMAX(4,12),DIR(12),DPR(12),BETA(1
     32),S(4,121),DS(4),SMIN(4,13),SMAX(4,13),A(1,6),D(5,121),PPR(1
     42),RFPWR(12),PIR(12),RETURN(13),CR(4),ALPHA,CONST,CPP1,CPP2,H
     51MAX,WCPP2,BPUFE,CPUDEF,BPUDUP,UPIR,PRJO(243)
      DIMENSION FLAG1(2,81),NOD(4),SBIR(2,81)
      REWIND IUNIT
      N=1
      NDIM1=NNS(1,N)*NNS(2,N)*NNS(3,N)*NNS(4,N)
      DO 7 M1=1,NDIM1
      FNM1(M1)=CONST
      SBIR(1,M1)=CONST
      FLAG1(1,M1)=0.
    7 D5NM1(1,M1)=CONST
      DO 10 IS=1,4
      NNOD=NNS(IS,N)
      DO 8 I=1,NNOD
      II=I
      IF(SI(IS).EQ.SL(IS,N,I))GO TO 9
    8 CONTINUE
    9 NOD(IS)=II
   10 CONTINUE
      M=(NOD(1)-1)*NNS(2,N)*NNS(3,N)*NNS(4,N)+(NOD(2)-1)*NNS(3,N)*N
     1NS(4,N)+(NOD(3)-1)*NNS(4,N)+NOD(4)
      FNM1(M)=0.
      D5NM1(1,M)=0.
      SBIR(1,M)=0.
      MO=0
      DO 34 N=2,NN
      MO=MO+1
      IF(MO.GT.12)MO=1
      NI2=NNS(1,N)
      NJ2=NNS(2,N)
      NK2=NNS(3,N)
      NL2=NNS(4,N)
      NI1=NNS(1,N-1)
      NJ1=NNS(2,N-1)
      NK1=NNS(3,N-1)
      NL1=NNS(4,N-1)
      NDIM2=NI2*NJ2*NK2*NL2
      K1L1=NK1*NL1
      J1K1L1=K1L1*NJ1
      NDIM1=J1K1L1*NI1
      DO 12 M2=1,NDIM2
      FN(M2)=CONST
      S1NM1(M2)=CONST
      S2NM1(M2)=CONST
      S3NM1(M2)=CONST
      S4NM1(M2)=CONST
      D5N(M2)=CONST
      D5NM1(2,M2)=CONST
      SBIR(2,M2)=CONST
      FLAG1(2,M2)=CONST
   12 CONTINUE
      M2=0
      DO 32 I2=1,NI2
      D1T=EY(1,N-1)-SL(1,N,I2)
      DO 31 J2=1,NJ2
      D2T=EY(2,N-1)-SL(2,N,J2)
      DO 30 K2=1,NK2
      D3T=EY(3,N-1)-SL(3,N,K2)
      DO 29 L2=1,NL2
      M2=M2+1
      DO 28 I1=1,NI1
      D1N=SL(1,N-1,I1)+D1T
      IF(D1N.LT.DMIN(1,MO))GO TO 28
      IA1=(I1-1)*J1K1L1
      DO 27 J1=1,NJ1
      D2N=SL(2,N-1,J1)+D2T
      IF(D2N.LT.DMIN(2,MO))GO TO 27
      IB1=(J1-1)*K1L1
      DO 26 K1=1,NK1
      D3N=SL(3,N-1,K1)+D3T
      IF(D3N.LT.DMIN(3,MO))GO TO 26
      IC1=(K1-1)*NL1
      DO 25 L1=1,NL1
      M1=IA1+IB1+IC1+L1
      IF(FNM1(M1).EQ.CONST)GO TO 25
      FLO2=D1N+D2N+EY(4,N-1)
      IF(FLO2.GE.DIR(MO))GO TO 13
      U5N=FLO2
      DEFIR=DIR(MO)-FLO2
      PIRSH=(DEFIR/DIR(MO))*100.
      GO TO 14
   13 U5N=DIR(MO)
      DEFIR=0.
      PIRSH=0.
   14 RUNOFF=ALPHA*D5NM1(1,M1)
   16 D4N=SL(4,N-1,L1)-SL(4,N,L2)+FLO2-U5N+RUNOFF
      IF(D4N.LT.DMIN(4,MO))GO TO 25
      FLO4=D3N+D4N+EY(5,N-1)
      SBIRM1=SBIR(1,M1)
C     CALCULATE BENEFITS
      SN1=SL(1,N-1,I1)
      SNP1=SL(1,N,I2)
      CALL POWER(SN1,SNP1,D1N,FLO4,MO,CPP1,H1MAX,WCPP2,CPUDEF,BPUDU
     1P,BPR,TOTP,A,DPR,RFPWR,PPRSH,PPRSP)
      IF(PIRSH.GT.80..OR.FLAG1(1,M1).EQ.1.)GO TO 17
      CALL IRRIG(PIRSH,BIR,MO,DIR,BETA,UPIR)
      TAGM2=0.
      TBN=BIR+BPR+FNM1(M1)
      SBIRM2=SBIRM1+BIR
      GO TO 21
   17 BIR=0.
      TAGM2=1.
      TBN=BPR+FNM1(M1)-SBIRM1
      SBIRM2=0.
   21 CONTINUE
      IF(TBN.LE.FN(M2))GO TO 24
      FN(M2)=TBN
      S1NM1(M2)=SL(1,N-1,I1)
      S2NM1(M2)=SL(2,N-1,J1)
      S3NM1(M2)=SL(3,N-1,K1)
      S4NM1(M2)=SL(4,N-1,L1)
      D5N(M2)=U5N
      D5NM1(2,M2)=D5NM1(1,M1)
      FLAG1(2,M2)=TAGM2
      SBIR(2,M2)=SBIRM2
   24 CONTINUE
   25 CONTINUE
   26 CONTINUE
   27 CONTINUE
   28 CONTINUE
   29 CONTINUE
   30 CONTINUE
   31 CONTINUE
   32 CONTINUE
      WRITE(IUNIT)N,(S1NM1(M2),S2NM1(M2),S3NM1(M2),S4NM1(M2),D5N(M2
     1),D5NM1(2,M2),M2=1,NDIM2)
      DO 33 M2=1,NDIM2
      FNM1(M2)=FN(M2)
      D5NM1(1,M2)=D5N(M2)
      FLAG1(1,M2)=FLAG1(2,M2)
      SBIR(1,M2)=SBIR(2,M2)
   33 CONTINUE
      IF(MO.LT.12)GO TO 34
      DO 331 M2=1,NDIM2
      FLAG1(1,M2)=0.
      SBIR(1,M2)=0.
  331 CONTINUE
   34 CONTINUE
      ENDFILE IUNIT
      RETURN
      END
```

```
      SUBROUTINE IRRIG(PIRSH,BIR,MO,DIR,BETA,UPIR)
      DIMENSION DIR(12),BETA(12)
C        CALCULATE GROSS IRRIGATION BENEFITS IN($1000.)
      IF(PIRSH.GT.10.)GO TO 15
      PLGB=0.8*PIRSH
      GO TO 18
   15 IF(PIRSH.GT.70.)GO TO 16
      PLGB=1.45*PIRSH-6.5
      GO TO 18
   16 PLGB=0.5*PIRSH+60.0
   18 BIR=(DIR(MO)*UPIR-(DIR(MO)*UPIR*PLGB/100.)*BETA(MO))/1000.
   19 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TRACE(NN,IUNIT,MNN,NNM1,NNP1,IFLAG)
C     TRACE TRAJECTORY WHICH SATISFIES THE BOUNDARY CONDITIONS
C     SI AND SF
      COMMON NNS(4,121),SI(4),SF(4),SL(4,121,3),S1NM1(81),S2NM1(81)
     1,S3NM1(81),S4NM1(81),FN(81),FNM1(81),D5N(81),D5NM1(2,81),EY(5
     2,120),YL(5,12,3),DMIN(4,12),DMAX(4,12),DIR(12),DPR(12),BETA(1
     32),S(4,121),DS(4),SMIN(4,13),SMAX(4,13),A(1,6),D(5,121),PPR(1
     42),RFPWR(12),PIR(12),RETURN(13),CR(4),ALPHA,CONST,CPP1,CPP2,H
     51MAX,WCPP2,BPUFE,CPUDEF,BPUDUP,UPIR,PRJO(243)
      DIMENSION NOD(4)
      DO 10 IS=1,4
   10 S(IS,NN)=SF(IS)
      MO=NN
      DO 15 NR=1,NNM1
      N=NNP1-NR
      MO=MO-1
      IF(MO.LT.1)MO=12
      IF(IFLAG.EQ.1)NM1=MO
      NM1=N-1
      NDIM=NNS(1,N)*NNS(2,N)*NNS(3,N)*NNS(4,N)
      DO 13 IS=1,4
      NNOD=NNS(IS,N)
      DO 11 I=1,NNOD
      II=I
      IF(S(IS,N).EQ.SL(IS,N,I))GO TO 12
   11 CONTINUE
   12 NOD(IS)=II
   13 CONTINUE
      M=(NOD(1)-1)*NNS(2,N)*NNS(3,N)*NNS(4,N)+(NOD(2)-1)*NNS(3,N)*N
     1NS(4,N)+(NOD(3)-1)*NNS(4,N)+NOD(4)
      IF(N.EQ.NN)MNN=M
      DO 14 IR=1,2
      BACKSPACE IUNIT
   14 CONTINUE
      READ(IUNIT)NP,(S1NM1(M2),S2NM1(M2),S3NM1(M2),S4NM1(M2),D5N(M2
     1),D5NM1(2,M2),M2=1,NDIM)
      S(1,N-1)=S1NM1(M)
      S(2,N-1)=S2NM1(M)
      S(3,N-1)=S3NM1(M)
      S(4,N-1)=S4NM1(M)
      D(1,N-1)=S(1,N-1)-S(1,N)+EY(1,NM1)
      D(2,N-1)=S(2,N-1)-S(2,N)+EY(2,NM1)
      D(3,N-1)=S(3,N-1)-S(3,N)+EY(3,NM1)
      D(4,N-1)=S(4,N-1)-S(4,N)+D(1,N-1)+D(2,N-1)+EY(4,NM1)+ALPHA*D5
     1NM1(2,M)-D5N(M)
      D(5,N-1)=D5N(M)
   15 CONTINUE
   16 CONTINUE
      RETURN
      END
      SUBROUTINE PLOT1(NN,IT,WIDTH,IC,CR,S)
      DIMENSION CR(4),S(4,121),YY(121),XX(121),TX(2),TY(2),FL(4)
      FIC=IC
      FIT=IT
      FNN=NN-1
      WIDTH=6.0
      FL(1)=7.5
      FL(2)=7.5
      FL(3)=6.5
      FL(4)=7.5
      IF(IT.GT.0)GO TO 102
C     GENERATE ARRAY X
      DO 101 N=1,NN
      FN=N
  101 XX(N)=FN-1.
  102 DO 107 IS=1,4
C     DEFINE REFERENCE POINT
      CALL CCP1PL(.5,.5,-3)
      FIS=IS
C     DEFINE SCALING FACTORS
      TX(1)=0.
      TX(2)=FNN/WIDTH
      TY(1)=0.
      TY(2)=CR(IS)/FL(IS)
      DO 103 N=1,NN
  103 YY(N)=S(IS,N)
      IF(IT.GT.0)GO TO 104
C     DRAW AND LABEL X AND Y AXIS
      CALL CCP5AX(TX(1),0.,'STAGE (MONTH)',-13,WIDTH,0.0,TX)
      CALL CCP5AX(TX(1),0.,'STATE (STORAGE IN 10**6 AF)',27,FL(IS)+
     1.5,90.0,TY)
C     DRAW CURVE
  104 CALL CCP6LN(XX,YY,NN,1,TX,TY)
C     LABEL CURVE
      IF(IT.GT.0)L=(NN-5)/2
      IF(IT.EQ.0)L=(NN-1)/2
      XLABEL=(XX(L)-TX(1))/TX(2)
      YLABEL=(YY(L)-TY(1))/TY(2)
      CALL CCP1PL(XLABEL,YLABEL,3)
      Y1=YLABEL+.3
      CALL CCP1PL(XLABEL+.3,Y1,2)
      CALL CCP1PL(XLABEL+.4,Y1,2)
      Y1=Y1-.04
      IF(IT.GT.0)GO TO 105
      CALL SYMBOL(XLABEL+.5,Y1,.08,'NOM. TRAJ.',0.,10)
      XTITLE=((XX(1)+.3)-TX(1))/TX(2)
      YTITLE=(CR(IS)-TY(1))/TY(2)
      CALL SYMBOL(XTITLE,YTITLE,.1,'TRAJ. OF RES. ',0.,14)
      CALL NUMBER(XTITLE+1.1,YTITLE,.1,FIS,0.,-1)
      GO TO 106
  105 CALL SYMBOL(XLABEL+.6,Y1,.08,'TRAJ. OF ITER. ',0.,15)
      CALL NUMBER(XLABEL+1.6,Y1,.08,FIT,0.,-1)
      CALL SYMBOL(XLABEL+1.8,Y1,.08,'IN CORRIDOR ',0.,12)
      CALL NUMBER(XLABEL+2.6,Y1,.08,FIC,0.,-1)
  106 CONTINUE
      CALL CCP1PL(WIDTH+.5,-.5,-3)
  107 CONTINUE
      IF(IT.GT.0)GO TO 108
      XOR=-4.*(WIDTH+1.)
      CALL CCP1PL(XOR,0.,-3)
  108 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE STATE(S,SMIN,SMAX,DS,SL,NNS,NN)
      DIMENSION S(4,121),SMIN(4, 13),SMAX(4, 13),DS(4),SL(4,121,3),
     1NNS(4,121)
C     SETUP THE CORRIDORS AROUND EACH TRAJECTORY.
      M=0
      DO 54 N=1,NN
      IF(N.EQ.NN)M=13
      M=M+1
      IF(M.GT.12)M=1
      DO 53 IS=1,4
      IF(S(IS,N).EQ.SMIN(IS,M).OR.S(IS,N).EQ.SMAX(IS,M))GO TO 50
      SD=S(IS,N)-DS(IS)
      IF(SD.LT.SMIN(IS,M))SD=SMIN(IS,M)
      SU=S(IS,N)+DS(IS)
      IF(SU.GT.SMAX(IS,M))SU=SMAX(IS,M)
      SL(IS,N,1)=SD
      SL(IS,N,2)=S(IS,N)
      SL(IS,N,3)=SU
      GO TO 52
   50 IF(S(IS,N).EQ.SMAX(IS,M))GO TO 51
      SL(IS,N,1)=S(IS,N)
      SL(IS,N,2)=S(IS,N)+DS(IS)
      SL(IS,N,3)=SL(IS,N,2)+DS(IS)
      GO TO 52
   51 SL(IS,N,3)=S(IS,N)
      SL(IS,N,2)=S(IS,N)-DS(IS)
      SL(IS,N,1)=SL(IS,N,2)-DS(IS)
   52 NNS(IS,N)=3
   53 CONTINUE
   54 CONTINUE
      RETURN
      END
      SUBROUTINE POWER(SN1,SNP1,D1N,FLO4,MO,CPP1,H1MAX,WCPP2,CPUDEF
     1,BPUDUP,BPR,TOTP,A,DPR,RFPWR,PPRSH,PPRSP)
      DIMENSION A(1,6),DPR(12),RFPWR(12)
C     CALCULATE BENEFITS FROM POWER IN($1000)
C     CALCULATE EPP1, ENERGY GENERATED BY PP1 IN(MW-HR/MO.)
      SN2=SN1*SN1
      SN3=SN2*SN1
      SN4=SN3*SN1
      SN5=SN4*SN1
      SNP12=SNP1*SNP1
      SNP13=SNP12*SNP1
      SNP14=SNP13*SNP1
      SNP15=SNP14*SNP1
      HN=A(1,1)+A(1,2)*SN1+A(1,3)*SN2+A(1,4)*SN3+A(1,5)*SN4+A(1,6)*
     1SN5
      HNP1=A(1,1)+A(1,2)*SNP1+A(1,3)*SNP12+A(1,4)*SNP13+A(1,5)*SNP1
     14+A(1,6)*SNP15
      AVH=(HN+HNP1)/2.
      WCPP1=.83876*CPP1*(SQRT(AVH))/(H1MAX**1.5)
C     WCPP1=WATER CAPACITY OF PP1 UNDER AVH IN(10**6 AF/MO.)
      IF(D1N.GT.WCPP1)GO TO 154
      EPP1=871.*AVH*D1N
      GO TO 155
  154 EPP1=871.*AVH*WCPP1
C     CALCULATE EPP2, ENERGY OUTPUT OF PP2 IN(MW-HR/MO.)
  155 IF(FLO4.GE.WCPP2)GO TO 161
      EPP2=144000.*FLO4
      GO TO 162
  161 EPP2=144000.*WCPP2
C     CALCULATE TOTAL BENEFITS FROM PP1 & PP2 IN($1000.)
  162 TOTP=EPP1+EPP2
      IF(TOTP.GE.DPR(MO))GO TO 163
      BPR=(RFPWR(MO)-(DPR(MO)-TOTP)*CPUDEF)/1000.
      PPRSP=0.
      PPRSH=((DPR(MO)-TOTP)/DPR(MO))*100.
      GO TO 164
  163 PPRSP=((TOTP-DPR(MO))/DPR(MO))*100.
      PPRSH=0.
      BPR=(RFPWR(MO)+(TOTP-DPR(MO))*BPUDUP)/1000.
  164 CONTINUE
      RETURN
      END
```

A-3. Notation Used in Output Tables for Clearwater River System

N	= Stage
MO	= Month
s_i	= Storage (state) or reservoir i, i = 1,2,3,4 in 10^6 ac.ft.
u_i	= Outflow (decision) from reservoir i, i = 1,2,3,4 in 10^6 ac.ft.
u_5	= Irrigation diversion in 10^6 ac.ft.
PIRSH	= Percent monthly irrigation shortage.
SWIR/YR	= Sum of the irrigation diversion per year from the beginning of the irrigation season to the end of month in question in 10^6 ac.ft.
BIR	= Monthly return due to irrigation activities in $1000.
SBIR/YR	= Sum of the returns per year from the beginning of the irrigation season to the end of the month in question in $1000.
TOTP	= Total monthly power generated by PP1 and PP2 in Mw hr.
PPRSP	= Percent monthly power surplus.
PPRSH	= Percent monthly power shortage.
SPR/YR	= Total power per year from the beginning of the operating year to the end of the month in question in Mw hr.
BPR	= Monthly return from hydropower.
SBPR/YR	= Total return per year from hydropower from the beginning of the operating year to the end of the month in question in $1000.
STB/YR	= Sum of the total returns per year from the beginning of the operating year to the end of the month in question in $100.
(M-AF)	= 10^6 ac.ft.
(K-$)	= $1000.
SWIRHR	= Sum of the total water diverted for irrigation during the entire time horizon in 10^6 ac.ft.
SBIRHR	= Sum of the total returns due to irrigation activities during the entire time horizon in 10^6 ac.ft.
SPRHR	= Sum of the total power generated during the entire time horizon in Mw hr.
SBPRHR	= Sum of the total returns due to power generated during the entire time horizon in $1000.

TABLE A-1. Trial Trajectory A and Detailed List of Activities

Trial Trajectory A (all units in 10^6 ac.ft.)

N	s_1	s_2	s_3	s_4	u_1	u_2	u_3	u_4	u_5
1	10.0000	1.0000	9.0000	10.0000	0.2112	0.1033	1.0675	0.0	0.3939
2	10.5000	1.0100	8.7000	10.0000	0.5197	0.1830	1.3098	0.3297	0.5840
3	11.5000	1.0200	8.5000	10.0000	1.1079	0.1139	1.1545	0.7953	0.6640
4	11.5000	1.0300	8.0000	10.0000	1.2969	0.0373	0.4267	0.7900	0.7600
5	10.5000	1.0300	7.7900	10.0000	1.2895	0.0128	0.0055	0.8156	0.7200
6	9.3000	1.0300	7.8800	10.0000	0.0676	0.0106	0.0089	0.2160	0.0838
7	9.3000	1.0300	7.9500	10.0000	0.0095	0.0150	0.0002	0.0251	0.0310
8	9.4000	1.0300	8.0600	10.0000	0.0386	0.0175	0.0022	0.0766	0.0
9	9.5000	1.0300	8.2100	10.0000	0.0663	0.0193	0.0084	0.1016	0.0
10	9.6000	1.0300	8.4200	10.0000	0.0340	0.0162	0.0047	0.0672	0.0
11	9.7000	1.0300	8.6000	10.0000	0.0308	0.0161	0.0773	0.0711	0.0
12	9.8000	1.0300	8.7000	10.0000	0.0269	0.0626	0.0098	0.1408	0.0
13	10.0000	1.0000	9.0000	10.0000					

Return ($1000) = 12952.14

Summary of Irrigation and Power Generation Activities (Trial Trajectory A)

N	MO	D5 (M-AF)	PIRSH (%)	SWIR/YR (M-AF)	BIR (K-$)	SBIR/YR (K-$)	TOTP (MW-HR)	PPRSP (%)	PPRSH (%)	SPR/YR (MW-HR)	BPR (K-$)	SBPR/YR (K-$)	STB/YR (K-$)
2	1	0.3939	20.58	0.3939	1988.1	1988.1	221645.0	43.93	0.0	221645.0	1179.5	1179.5	3167.6
3	2	0.5840	0.0	0.9779	2761.7	4749.8	276266.8	66.43	0.0	497911.8	1327.4	2506.9	7256.7
4	3	0.6640	0.0	1.6419	3140.0	7889.8	278475.7	56.45	0.0	776387.4	1396.7	3903.6	11793.4
5	4	0.7600	0.0	2.4019	3594.0	11483.8	276266.8	51.80	0.0	1052654.0	1415.4	5319.0	16802.8
6	5	0.7200	0.0	3.1219	3404.8	14888.6	244344.3	31.37	0.0	1296998.0	1389.5	6708.5	21597.1
7	6	0.0838	83.11	3.2057	0.0	0.0	56926.4	0.0	68.72	1353924.0	1023.9	7732.2	7732.3
8	7	0.0310	88.93	3.2367	0.0	0.0	8917.5	0.0	94.56	1362841.0	837.8	8570.2	8570 2
9	8	0.0	0.0	3.2367	0.0	0.0	29548.0	0.0	81.76	1392388.0	869.1	9439.3	9439.3
10	9	0.0	0.0	3.2367	0.0	0.0	49454.2	0.0	70.21	1441842.0	928.9	10368.2	10368.2
11	10	0.0	0.0	3.2367	0.0	0.0	33671.9	0.0	79.47	1475513.0	887.3	11255.5	11255.5
12	11	0.0	0.0	3.2367	0.0	0.0	47007.9	0.0	68.66	1522520.0	844.0	12099.5	12099.5
13	12	0.0	0.0	3.2367	0.0	0.0	61311.2	0.0	58.01	1583831.0	852.6	12952.1	12952.1

SWIRHR (M-AF) = 3.2367 SBIRHR ($1000) = 0.0

SPRHR (MW-HR) = 1583831.0 SBPRHR ($1000) = 12952.1

TABLE A-2. Final Trajectory A and Detailed List of Activities

Final Trajectory A (all units in 10^6 ac.ft.)

N	s_1	s_2	s_3	s_4	u_1	u_2	u_3	u_4	u_5
1	10.0000	1.0000	9.0000	10.0000	0.3642	0.0539	0.6985	0.0519	0.4960
2	10.3470	1.0594	9.0690	9.9496	0.4693	0.0280	0.8191	0.0079	0.5840
3	11.3974	1.2244	9.3597	10.0966	0.5079	0.0092	0.4291	0.0036	0.6640
4	11.9974	1.2491	9.5851	10.2436	0.6969	0.0476	0.2948	0.0149	0.7600
5	11.5974	1.2388	9.5070	10.4290	0.6175	0.1082	0.5617	0.0290	0.7200
6	11.0694	1.1434	9.0408	10.6390	0.3676	0.1310	0.6801	0.1059	0.4960
7	10.7694	1.0230	8.4396	10.7573	0.2374	0.0541	0.4443	0.1360	0.2800
8	10.6415	0.9839	8.1055	10.7881	0.1070	0.0257	0.0825	0.7704	0.0
9	10.6731	0.9757	8.1752	10.2456	0.3175	0.0394	0.0456	0.5928	0.0
10	10.5219	0.9556	8.3480	10.0257	0.3430	0.0065	0.0047	0.7961	0.0
11	10.3129	0.9653	8.5280	9.5961	0.3308	0.0064	0.0053	0.3374	0.0
12	10.1129	0.9750	8.7000	9.6201	0.3398	0.0076	0.0098	0.0188	0.0
13	10.0000	1.0000	9.0000	10.0000					

Return ($1000) = 33731.69

Summary of Irrigation and Power Generation Activities (Final Trajectory A)

N	MO	D5 (M-AF)	PIRSH (%)	SWIR/YR (M-AF)	BIR (K-$)	SBIR/YR (K-$)	TOTP (MW-HR)	PPRSP (%)	PPRSH (%)	SPR/YR (MW-HR)	BPR (K-$)	SBPR/YR (K-$)	STB/YR (K-$)
2	1	0.4960	0.0	0.4960	2345.5	2345.5	269639.1	75.09	0.0	269639.1	1251.5	1251.5	3597.0
3	2	0.5840	0.0	1.0800	2761.7	5107.2	275323.2	65.86	0.0	544962.3	1326.0	2577.4	7684.7
4	3	0.6640	0.0	1.7440	3140.0	8247.2	212562.9	19.42	0.0	757525.2	1297.8	3875.3	12122.5
5	4	0.7600	0.0	2.5040	3594.0	11841.2	182327.4	0.18	0.0	939852.6	1274.5	5149.8	16991.0
6	5	0.7200	0.0	3.2240	3404.8	15246.0	218098.9	17.26	0.0	1157951.0	1350.1	6499.9	21745.9
7	6	0.4960	0.0	3.7200	2345.5	17591.0	244126.1	34.14	0.0	1402077.0	1367.2	7867.1	25458.7
8	7	0.2800	0.0	4.0000	1324.1	18915.7	171440.6	4.54	0.0	1573517.0	1159.2	9026.3	27941.9
9	8	0.0	0.0	4.0000	0.0	18915.7	166205.9	2.60	0.0	1739722.0	1140.3	10166.6	29082.2
10	9	0.0	0.0	4.0000	0.0	18915.7	216901.8	30.66	0.0	1956623.0	1238.4	11404.9	30320.6
11	10	0.0	0.0	4.0000	0.0	18915.7	249919.5	52.39	0.0	2206542.0	1276.9	12681.8	31597.4
12	11	0.0	0.0	4.0000	0.0	18915.7	182341.2	21.56	0.0	2388883.0	1098.5	13780.3	32695.9
13	12	0.0	0.0	4.0000	0.0	18915.7	155130.2	6.25	0.0	2544013.0	1035.7	14816.0	33731.6

SWIRHR (M-AF) = 4.0000 SBIRHR ($1000) = 18915.7

SPRHR (MW-HR) = 2544013.0 SBPRHR ($1000) = 14816.0

TABLE A-3. Trial Trajectory B and Detailed List of Activities

Trial Trajectory B (all units in 10^6 ac.ft.)

N	s_1	s_2	s_3	s_4	u_1	u_2	u_3	u_4	u_5
1	10.0000	1.0000	9.0000	10.0000	0.7112	0.1133	0.7675	0.4079	0.4960
2	10.0000	1.0000	9.0000	10.0000	1.5197	0.1930	1.1098	1.3703	0.5840
3	10.0000	1.0000	9.0000	10.0000	1.1079	0.1239	0.6545	0.8053	0.6640
4	10.0000	1.0000	9.0000	10.0000	0.2969	0.0373	0.2167	0.1992	0.3508
5	10.0000	1.0000	9.0000	10.0000	0.0895	0.0128	0.0955	0.1052	0.1076
6	10.0000	1.0000	9.0000	10.0000	0.0676	0.0106	0.0789	0.0323	0.0838
7	10.0000	1.0000	9.0000	10.0000	0.1095	0.0150	0.1102	0.0251	0.1310
8	10.0000	1.0000	9.0000	10.0000	0.1386	0.0175	0.1522	0.2066	0.0
9	10.0000	1.0000	9.0000	10.0000	0.1663	0.0193	0.2184	0.2016	0.0
10	10.0000	1.0000	9.0000	10.0000	0.1340	0.0162	0.1847	0.1672	0.0
11	10.0000	1.0000	9.0000	10.0000	0.1308	0.0161	0.1173	0.1711	0.0
12	10.0000	1.0000	9.0000	10.0000	0.2269	0.0326	0.3098	0.3108	0.0
13	10.0000	1.0000	9.0000	10.0000					

Return ($1000) = 13423.34

Summary of Irrigation and Power Generation Activities (Trial Trajectories B)

N	MO	D5 (M-AF)	PIRSH (%)	SWIR/YR (M-AF)	BIR (K-$)	SBIR/YR (K-$)	TOTP (MW-HR)	PPRSP (%)	PPRSH (%)	SPR/YR (MW-HR)	BPR (K-$)	SBPR/YR (K-$)	STB/YR (K-$)
2	1	0.4960	0.0	0.4960	2345.5	2345.5	271523.1	76.31	0.0	271523.1	1254.3	1254.3	3599.8
3	2	0.5840	0.0	1.0800	2761.7	5107.2	271523.1	63.57	0.0	543046.3	1320.3	2574.6	7681.8
4	3	0.6640	0.0	1.7440	3140.0	8247.2	271523.1	52.54	0.0	814569.4	1386.3	3960.8	12208.1
5	4	0.3508	53.84	2.0948	1021.7	9269.0	169622.1	0.0	6.80	984191.4	1249.2	5210.1	14479.0
6	5	0.1076	85.06	2.2024	0.0	0.0	62072.5	0.0	66.63	1046263.9	1054.1	6264.2	6264.2
7	6	0.0838	83.11	2.2862	0.0	0.0	41049.3	0.0	77.45	1087313.0	992.1	7256.3	7256.3
8	7	0.1310	53.21	2.4172	979.4	0.0	60351.1	0.0	63.20	1147664.0	940.7	8197.0	8197.0
9	8	0.0	0.0	2.4172	0.0	0.0	105618.3	0.0	34.80	1253282.0	1021.2	9218.3	9218.3
10	9	0.0	0.0	2.4172	0.0	0.0	129931.8	0.0	21.73	1383213.0	1089.9	10308.1	10308.1
11	10	0.0	0.0	2.4172	0.0	0.0	109643.3	0.0	33.14	1492856.0	1039.3	11347.4	11347.4
12	11	0.0	0.0	2.4172	0.0	0.0	111415.0	0.0	25.72	1604271.0	972.8	12320.2	12320.2
13	13	0.0	0.0	2.4172	0.0	0.0	200076.3	37.04	0.0	1804347.0	1103.1	13423.3	13423.3

SWIRHR (M-AF) = 2.4172 SBIRHR ($1000) = 0.0

SPRHR (MW-HR) = 1804347.0 SBPRHR ($1000) = 13423.3

TABLE A-4. Final Trajectory B and Detailed List of Activities

Final Trajectory B (all units in 10^6 ac.ft.)

N	s_1	s_2	s_3	s_4	u_1	u_2	u_3	u_4	u_5
1	10.0000	1.0000	9.0000	10.0000	0.3392	0.0914	0.7279	0.0356	0.4960
2	10.3720	1.0219	9.0396	9.9784	0.4097	0.0826	0.8101	0.0029	0.5840
3	11.4820	1.1323	9.3393	10.1254	0.5079	0.0989	0.8964	0.0273	0.6640
4	12.0820	1.1573	9.0974	10.2785	0.5969	0.1551	0.4447	0.5403	0 7600
5	11.7820	1.0394	8.8694	9.9460	0.6895	0.0253	0.4238	0.1783	0.7200
6	11.1820	1.0269	8.5411	9.9958	0.3676	0.1234	0.5259	0.0282	0.4960
7	10.8820	0.9141	8.0941	10.1842	0.2625	0.0150	0.0969	0.3881	0.2800
8	10.7290	0.9141	8.1074	9.9489	0.2148	0.0042	0.0132	0.6638	0.0
9	10.6528	0.9274	8.2464	9.5994	0.3133	0.0046	0.0084	0.3000	0.0
10	10.5058	0.9421	8.4564	9.6333	0.3269	0.0030	0.0377	0.5151	0.0
11	10.3129	0.9553	8.6034	9.4651	0.3408	0.0014	0.0807	0.2122	0.0
12	10.1029	0.9700	8.7000	9.6193	0.3298	0.0026	0.0098	0.0030	0.0
13	10.0000	1.0000	9.0000	10.0000					

Return ($1000) = 33788.52

Summary of Irrigation and Power Generation Activities (Final Trajectory B)

N	MO	D5 (M-AF)	PIRSH (%)	SWIR/YR (M-AF)	BIR (K-$)	SBIR/YR (K-$)	TOTP (MW-HR)	PPRSP (%)	PPRSH (%)	SPR/YR (MW-HR)	BPR (K-$)	SBPR/YR (K-$)	STB/YR (K-$)
2	1	0.4960	0.0	0.4960	2345.5	2345.5	266323.4	72.94	0.0	266323.4	1246.5	1246.5	3592.0
3	2	0.5840	0.0	1.0800	2761.7	5107.2	273542.3	64.78	0.0	539865.7	1323.3	2569.8	7677.0
4	3	0.6640	0.0	1.7440	3140.0	8247.2	279256.1	56.89	0.0	819121.8	1397.9	3967.7	12214.9
5	4	0.7600	0.0	2.5040	3594.0	11841.2	280059.6	53.88	0.0	1099181.0	1421.1	5388.8	17230.0
6	5	0.7200	0.0	3.2240	3404.8	15246.0	220328.3	18.46	0.0	1319509.0	1353.5	6742.3	21988.3
7	6	0.4960	0.0	3.7200	2345.5	17591.6	211230.9	16.06	0.0	1530739.0	1317.8	8060.1	25651.7
8	7	0.2800	0.0	4.0000	1324.1	18915.7	167024.4	1.84	0.0	1697763.0	1152.5	9212.6	28128.3
9	8	0.0	0.0	4.0000	0.0	18915.7	179916.8	11.06	0.0	1877679.0	1160.9	10373.5	29289.2
10	9	0.0	0.0	4.0000	0.0	18915.7	167814.3	1.09	0.0	2045493.0	1164.7	11538.2	30453.9
11	10	0.0	0.0	4.0000	0.0	18915.7	208404.1	27.08	0.0	2253897.0	1214.6	12752.8	31668.5
12	11	0.0	0.0	4.0000	0.0	18915.7	178731.4	19.15	0.0	2432628.0	1093.1	13845.9	32761.6
13	12	0.0	0.0	4.0000	0.0	18915.7	149275.3	2.24	0.0	2581903.0	1026.9	14872.8	33788.5

SWIRHR (M-AF) = 4.0000 SBIRHR ($1000) = 18915.7

SPRHR (MW-HR) = 2581903.0 SBPRHR ($1000) = 14872.8

TABLE A-5. Trail Trajectory C and Detailed List of Activities

Trial Trajectory C (all units in 10^6 ac.ft.)

N	s_1	s_2	s_3	s_4	u_1	u_2	u_3	u_4	u_5
1	10.0000	1.0000	9.0000	10.0000	0.7112	0.1133	0.7675	0.4079	0.4960
2	10.0000	1.0000	9.0000	10.0000	1.5197	0.1930	1.1098	1.3703	0.5840
3	10.0000	1.0000	9.0000	10.0000	1.1079	0.1239	0.6545	0.8053	0.6640
4	10.0000	1.0000	9.0000	10.0000	0.2969	0.0373	0.2167	0.1992	0.3508
5	10.0000	1.0000	9.0000	10.0000	0.0895	0.0128	0.0955	0.1052	0.1076
6	10.0000	1.0000	9.0000	10.0000	0.0676	0.0106	0.0789	0.0323	0.0838
7	10.0000	1.0000	9.0000	10.0000	0.1095	0.0150	0.1102	0.0251	0.1310
8	10.0000	1.0000	9.0000	10.0000	0.1386	0.0175	0.1522	0.2066	0.0
9	10.0000	1.0000	9.0000	10.0000	0.1663	0.0193	0.2184	0.2016	0.0
10	10.0000	1.0000	9.0000	10.0000	0.1340	0.0162	0.1847	0.1672	0.0
11	10.0000	1.0000	9.0000	10.0000	0.1308	0.0161	0.1773	0.1711	0.0
12	10.0000	1.0000	9.0000	10.0000	0.2269	0.0326	0.3098	0.3108	0.0
13	10.0000	1.0000	9.0000	10.0000	0.7112	0.1133	0.7675	0.4079	0.4960
14	10.0000	1.0000	9.0000	10.0000	1.5197	0.1930	1.1098	1.3703	0.5840
15	10.0000	1.0000	9.0000	10.0000	1.1079	0.1239	0.6545	0.8053	0.6640
16	10.0000	1.0000	9.0000	10.0000	0.2969	0.0373	0.2167	0.1992	0.3508
17	10.0000	1.0000	9.0000	10.0000	0.0895	0.0128	0.0955	0.1052	0.1076
18	10.0000	1.0000	9.0000	10.0000	0.0676	0.0106	0.0789	0.0323	0.0838
19	10.0000	1.0000	9.0000	10.0000	0.1095	0.0150	0.1102	0.0251	0.1310
20	10.0000	1.0000	9.0000	10.0000	0.1386	0.0175	0.1522	0.2066	0.0
21	10.0000	1.0000	9.0000	10.0000	0.1663	0.0193	0.2184	0.2016	0.0
22	10.0000	1.0000	9.0000	10.0000	0.1340	0.0162	0.1847	0.1672	0.0
23	10.0000	1.0000	9.0000	10.0000	0.1308	0.0161	0.1773	0.1711	0.0
24	10.0000	1.0000	9.0000	10.0000	0.2269	0.0326	0.3098	0.3108	0.0
25	10.0000	1.0000	9.0000	10.0000					

Return ($1000) = 26846.67

Summary of Irrigation and Power Generation Activities (Trial Trajectory C)

N	MO	D5 (M-AF)	PIRSH (%)	SWIR/YR (M-AF)	BIR (K-$)	SBIR/YR (K-$)	TOTP (MW-HR)	PPRSP (%)	PPRSH (%)	SPR/YR (MW-HR)	BPR (K-$)	SBPR/YR (K-$)	STB/YR (K-$)
2	1	0.4960	0.0	0.4960	2345.5	2345.5	271523.1	76.31	0.0	271523.1	1254.3	1254.3	3599.8
3	2	0.5840	0.0	1.0800	2761.7	5107.2	271523.1	63.57	0.0	543046.3	1320.3	2574.6	7681.8
4	3	0.6640	0.0	1.7440	3140.0	8247.2	271523.1	52.54	0.0	814569.4	1386.3	3960.8	12208.1
5	4	0.3508	53.84	2.0948	1021.7	9269.0	169622.1	0.0	6.80	984191.4	1249.2	5210.1	14479.0
6	5	0.1076	85.06	2.2024	0.0	0.0	62072.5	0.0	66.63	1046263.9	1054.1	6264.2	6264.2
7	6	0.0838	83.11	2.2862	0.0	0.0	41049.3	0.0	77.45	1087313.0	992.1	7256.3	7256.3
8	7	0.1310	53.21	2.4172	979.4	0.0	60351.0	0.0	63.20	1147664.0	940.7	8197.0	8197.0
9	8	0.0	0.0	2.4172	0.0	0.0	105618.3	0.0	34.80	1253282.0	1021.2	9218.3	9218.3
10	9	0.0	0.0	2.4172	0.0	0.0	129931.8	0.0	21.73	1383213.0	1089.9	10308.1	10308.1
11	10	0.0	0.0	2.4172	0.0	0.0	109643.3	0.0	33.14	1492856.0	1039.3	11347.4	11347.4
12	11	0.0	0.0	2.4172	0.0	0.0	111415.0	0.0	25.72	1604271.0	972.8	12320.2	12320.2
13	12	0.0	0.0	2.4172	0.0	0.0	200076.3	37.04	0.0	1804347.0	1103.1	13423.3	13423.3
14	1	0.4960	0.0	0.4960	2345.5	2345.5	271523.1	76.31	0.0	271523.1	1254.3	1254.3	3599.8
15	2	0.5840	0.0	1.0800	2761.7	5107.2	271523.1	63.57	0.0	543046.3	1320.3	2574.6	7681.8
16	3	0.6640	0.0	1.7440	3140.0	8247.2	271523.1	52.54	0.0	814569.4	1386.3	3960.8	12208.1
17	4	0.3508	53.84	2.0948	1021.7	9269.0	169622.1	0.0	6.80	984191.4	1249.2	5210.1	14479.0
18	5	0.1076	85.06	2.2024	0.0	0.0	62072.5	0.0	66.63	1046263.9	1054.1	6264.2	6264.2
19	6	0.0838	83.11	2.2862	0.0	0.0	41049.3	0.0	77.45	1087313.0	992.1	7256.3	7256.3
20	7	0.1310	53.21	2.4172	979.4	0.0	60351.1	0.0	63.20	1147664.0	940.7	8197.0	8197.0
21	8	0.0	0.0	2.4172	0.0	0.0	105618.3	0.0	34.80	1253282.0	1021.2	9218.3	9218.3
22	9	0.0	0.0	2.4172	0.0	0.0	129931.8	0.0	21.73	1383213.0	1089.9	10308.1	10308.1
23	10	0.0	0.0	2.4172	0.0	0.0	109643.3	0.0	33.14	1492856.0	1039.3	11347.4	11347.4
24	11	0.0	0.0	2.4172	0.0	0.0	111415.0	0.0	25.72	1604271.0	972.8	12320.2	12320.2
25	12	0.0	0.0	2.4172	0.0	0.0	200076.3	37.04	0.0	1804347.0	1103.1	13423.3	13423.3

SWIRHR (M-AF) = 4.8344 SBIRHR ($1000) = 0.0

SPRHR (MW-HR) = 3608694.0 SBPRHR ($1000) = 26846.7

TABLE A-6. Final Trajectory C and Detailed List of Activities

Final Trajectory C (all units in 10^6 ac.ft.)

N	s_1	s_2	s_3	s_4	u_1	u_2	u_3	u_4	u_5
1	10.0000	1.0000	9.0000	10.0000	0.3392	0.0792	0.7495	0.0018	0.4960
2	10.3720	10.841	9.0180	10.0000	0.4097	0.0823	0.7832	0.0242	0.5840
3	11.4820	1.1448	9.3446	10.1254	0.5079	0.0983	0.8294	0.0636	0.6640
4	12.0820	1.1704	9.1697	10.2415	0.5969	0.1767	0.4795	0.5045	0.7600
5	11.7820	1.0310	8.9069	9.9664	0.6895	0.0353	0.6850	0.2696	0.7200
6	11.1820	1.0085	8.3174	9.9349	0.3676	0.1328	0.6915	0.0910	0.4960
7	10.8820	0.8863	7.7048	10.0699	0.2625	0.0190	0.9142	0.0098	0.2800
8	10.7290	0.8823	6.9008	10.2169	0.0665	0.0008	0.8178	0.1379	0.0
9	10.8011	0.8990	6.2352	10.2415	0.1292	0.0440	0.1087	0.6926	0.0
10	10.8382	0.8743	6.3449	9.7381	0.3440	0.0102	0.5306	0.2200	0.0
11	10.6282	0.8803	5.9990	9.8893	0.3408	0.0065	0.3135	0.4496	0.0
12	10.4182	0.8899	5.8628	9.8112	0.3298	0.0007	0.0127	0.0319	0.0
13	10.3153	0.9218	6.1599	10.1611	0.4112	0.0089	0.6470	0.0035	0.4960
14	10.6153	1.0262	6.2804	10.1611	0.3197	0.1791	0.0628	0.2161	0.5840
15	11.8153	1.0401	7.3274	10.1014	0.5979	0.0090	0.2488	0.1624	0.6640
16	12.3253	1.1550	7.7331	10.1194	0.5969	0.1570	0.0067	0.3168	0.7600
17	12.0253	1.0353	7.9431	10.0123	0.6895	0.0281	0.1043	0.2618	0.7200
18	11.4253	1.0200	7.9343	9.9814	0.3676	0.1234	0.0068	0.4011	0.4960
19	11.1253	0.9072	8.0064	9.7969	0.3437	0.0041	0.0235	0.5532	0.2800
20	10.8911	0.9181	8.0931	9.4668	0.3486	0.0075	0.0493	0.4237	0.0
21	10.6811	0.9281	8.1960	9.4944	0.3322	0.0046	0.0084	0.2622	0.0
22	10.5152	0.9428	8.4060	9.5850	0.3402	0.0037	0.0377	0.5313	0.0
23	10.3090	0.9553	8.5530	9.4146	0.3369	0.0014	0.0303	0.1270	0.0
24	10.1029	0.9700	8.7000	9.6501	0.3298	0.0026	0.0098	0.0338	0.0
25	10.0000	1.0000	9.0000	10.0000					

Return ($1000) = 67541.06

Summary of Irrigation and Power Generation Activities (Final Trajectory C)

N	MC	D5 (M-AF)	PIRSH (%)	SWIR/YR (M-AF)	BIR (K-$)	SBIR/YR (K-$)	TOTP (MW-HR)	PPRSP (%)	PPRSH (%)	SPR/YR (MW-HR)	BPR (K-$)	SBPR/YR (K-$)	STB/YR (K-$)
2	1	0.4960	0.0	0.4960	2345.5	2345.5	264578.1	71.80	0.0	264578.1	1243.9	1243.9	3589.4
3	2	0.5840	0.0	1.0800	2761.7	5107.2	272731.0	64.30	0.0	537309.1	1322.1	2566.0	7673.2
4	3	0.6640	0.0	1.7440	3140.0	8247.2	274841.6	54.41	0.0	812150.7	1391.3	3957.2	12204.4
5	4	0.7600	0.0	2.5040	3594.0	11841.2	279921.6	53.80	0.0	1092072.0	1420.9	5378.1	17219.3
6	5	0.7200	0.0	3.2240	3404.8	15246.0	271089.3	45.75	0.0	1363161.0	1429.6	6807.7	22053.8
7	6	0.4960	0.0	3.7200	2345.5	17591.6	244121.3	34.13	0.0	1607282.0	1367.2	8174.9	25766.5
8	7	0.2800	0.0	4.0000	1324.1	18915.7	230245.8	40.39	0.0	1837527.0	1247.4	9422.3	28337.9
9	8	0.0	0.0	4.0000	0.0	18915.7	166430.8	2.74	0.0	2003957.0	1140.6	10562.9	29478.6
10	9	0.0	0.0	4.0000	0.0	18915.7	172643.4	4.00	0.0	2176600.0	1172.0	11734.9	30650.5
11	10	0.0	0.0	4.0000	0.0	18915.7	244028.7	48.80	0.0	2420628.0	1268.0	13002.9	31918.6
12	11	0.0	0.0	4.0000	0.0	18915.7	247444.8	64.96	0.0	2668072.0	1196.2	14199.1	33114.7
13	12	0.0	0.0	4.0000	0.0	18915.7	154864.3	6.07	0.0	2822936.0	1035.3	15234.4	34150.0
14	1	0.4960	0.0	0.4960	2345.5	2345.5	256724.5	66.70	0.0	256724.5	1232.1	1232.1	3577.6
15	2	0.5840	0.0	1.0800	2761.7	5107.2	183765.9	10.70	0.0	440490.4	1188.6	2420.7	7528.0
16	3	0.6640	0.0	1.7440	3140.0	8247.2	206503.4	16.01	0.0	646993.8	1288.8	3709.5	11956.7
17	4	0.7600	0.0	2.5040	3594.0	11841.2	185653.7	2.01	0.0	832647.4	1279.5	4989.0	16830.2
18	5	0.7200	0.0	3.2240	3404.8	15246.0	187281.6	0.69	0.0	1019929.0	1303.9	6292.9	21538.9
19	6	0.4960	0.0	3.7200	2345.5	17591.6	191215.4	5.06	0.0	1211144.0	1287.8	7580.7	25172.3
20	7	0.2800	0.0	4.0000	1324.1	18915.7	210274.3	28.22	0.0	1421418.0	1217.4	8798.1	27713.8
21	8	0.0	0.0	4.0000	0.0	18915.7	199227.2	22.98	0.0	1620645.0	1189.8	9987.9	28903.6
22	9	0.0	0.0	4.0000	0.0	18915.7	169246.1	1.96	0.0	1789891.0	1166.9	11154.8	30070.5
23	10	0.0	0.0	4.0000	0.0	18915.7	215517.9	31.41	0.0	2005408.0	1225.3	12380.1	31295.7
24	11	0.0	0.0	4.0000	0.0	18915.7	157800.6	5.20	0.0	2163208.0	1061.7	13441.8	32357.4
25	12	0.0	0.0	4.0000	0.0	18915.7	153715.4	5.28	0.0	2316923.0	1033.6	14475.4	33391.0

SWIRHR (M-AF) = 8.0000 SBIRHR ($1000) = 37831.3

SPRHR (MW-HR) = 5139859.0 SBPRHR ($1000) = 29709.7

A-4. FORTRAN IV Program used to Analyze the System Described in Figure 7.

```
      COMMON S(4,13),DS(4,13),SMIN(4,13),SMAX(4,13),SL(4,13,3),NNS(4,13)
     1,SI(4),SF(4),DMIN(4),DMAX(4),C1(12),C2(12),C3(12),C4(12),FN(81),U(
     2324),D(4,13),FU(13,324)
      READ 1002,NN,ITER,NCOR
      PRINT 4000,NN,ITER,NCOR
      NNM1=NN-1
      READ 1003,(DMIN(I),I=1,4)
      PRINT 4001,(DMIN(I),I=1,4)
      READ 1003,(DMAX(I),I=1,4)
      PRINT 4001,(DMAX(I),I=1,4)
      READ 1003,(SI(I),I=1,4)
      PRINT 4001,(SI(I),I=1,4)
      READ 1003,(SF(I),I=1,4)
      PRINT 4001,(SF(I),I=1,4)
      DO 100 IS=1,4
      READ 1003,(S(IS,N),N=1,NN)
      READ 1003,(  DS(IS,N),N=1,NN)
      PRINT 4001,(  DS(IS,N),N=1,NN)
      READ 1003,(SMIN(IS,N),N=1,NN)
      PRINT 4001,(SMIN(IS,N),N=1,NN)
      READ 1003,(SMAX(IS,N),N=1,NN)
      PRINT 4001,(SMAX(IS,N),N=1,NN)
  100 CONTINUE
      DO 101 N=1,NNM1
      READ 1003,C1(N),C2(N),C3(N),C4(N)
      PRINT 4001,C1(N),C2(N),C3(N),C4(N)
  101 CONTINUE
      IC=0
      IT=0
      PRINT 4003,IT,IC
      BIN=0.
      DO 99 N=1,NNM1
      D(1,N)=S(1,N)+2.-S(1,N+1)
      D(2,N)=S(2,N)+3.-S(2,N+1)
      D(3,N)=S(3,N)+D(2,N)-S(3,N+1)
      D(4,N)=S(4,N)+D(1,N)+D(3,N)-S(4,N+1)
      BIN=BIN+C1(N)*D(1,N)+C2(N)*D(2,N)+C3(N)*D(3,N)+C4(N)*D(4,N)
      PRINT 4004,S(1,N),S(2,N),S(3,N),S(4,N),D(1,N),D(2,N),D(3,N),D(4,N)
   99 CONTINUE
      PRINT 4004,S(1,NN),S(2,NN),S(3,NN),S(4,NN)
      PRINT 4005,BIN
      FNITM1=BIN
      DO 505 IC=1,NCOR
      DO 500 IT=1,ITER
C
      CALL STATE(NN)
      DO 105 IS=1,4
      DO 105 N=1,NN
      LLL=NNS(IS,N)
      PRINT 4004,(SL(IS,N,L9),L9=1,LLL)
  105 CONTINUE
C
C
      CALL DP(NN)
C
      CALL TRACE(NN,MNN,NNM1)
C
      PRINT 4003,IT,IC
      DO 103 N=1,NNM1
      PRINT 4004,S(1,N),S(2,N),S(3,N),S(4,N),D(1,N),D(2,N),D(3,N),D(4,N)
  103 CONTINUE
      PRINT 4004,S(1,NN),S(2,NN),S(3,NN),S(4,NN)
      PRINT 4005,FN(MNN)
      DO 106 IS=1,4
  106 PRINT 4007,(  DS(IS,N),N=1,NN)
      IF(FN(MNN).LE.FNITM1)GO TO 502
  104 FNITM1=FN(MNN)
  500 CONTINUE
  502 CONTINUE
      DO 503 IS=1,4
      DO 503 N=1,NN
  503 DS(IS,N)=DS(IS,N)*.7
  505 CONTINUE
  506 CONTINUE
 1002 FORMAT(16I5)
 1003 FORMAT(16F5.0)
 1004 FORMAT(10F8.3)
 4000 FORMAT(1X,20I6)
 4001 FORMAT(1X,20F6.1)
 4002 FORMAT(1X,3I6,17F6.1)
 4003 FORMAT(1H,/////,28X,'TRAJECTORIES AFTER ',I2,' ITERATIONS IN CORRI
     1DOR ',I2,///)
 4004 FORMAT(1X,8F10.4)
 4005 FORMAT(1X,'BENEFIT=',F10.2)
 4006 FORMAT(1X,'N=',I2,5X,'NP=',I2)
 4007 FORMAT(1X,13F7.4)
      STOP
      END
      SUBROUTINE DP(NN)
      COMMON S(4,13),DS(4,13),SMIN(4,13),SMAX(4,13),SL(4,13,3),NNS(4,13)
     1,SI(4),SF(4),DMIN(4),DMAX(4),C1(12),C2(12),C3(12),C4(12),FN(81),U(
     2324),D(4,13),FU(13,324)
      DIMENSION NOD(4),FNM1(81)
      N=1
      NDIM1=NNS(1,N)*NNS(2,N)*NNS(3,N)*NNS(4,N)
      DO 7 M1=1,NDIM1
    7 FNM1(M1)=-1.E5
      DO 10 IS=1,4
      NNOD=NNS(IS,N)
      DO 8 I=1,NNOD
      II=I
      IF(SI(IS).EQ.SL(IS,N,I))GO TO 9
    8 CONTINUE
    9 NOD(IS)=II
   10 CONTINUE
      M=(NOD(1)-1)*NNS(2,N)*NNS(3,N)*NNS(4,N)+(NOD(2)-1)*NNS(3,N)*NNS(4,
     1N)+(NOD(3)-1)*NNS(4,N)+NOD(4)
      PRINT 4002,M
      FNM1(M)=0.
      DO 30 N=2,NN
      NI2=NNS(1,N)
      NJ2=NNS(2,N)
      NK2=NNS(3,N)
      NL2=NNS(4,N)
      NI1=NNS(1,N-1)
      NJ1=NNS(2,N-1)
      NK1=NNS(3,N-1)
      NL1=NNS(4,N-1)
      K2L2=NK2*NL2
      J2K2L2=K2L2*NJ2
      NDIM2=J2K2L2*NI2
      DO 11 M2=1,NDIM2
      FN(M2)=-1.E5
      U(M2)=-1.E5
      U(NDIM2+M2)=-1.E5
      U(2*NDIM2+M2)=-1.E5
      U(3*NDIM2+M2)=-1.E5
   11 CONTINUE
      K1L1=NK1*NL1
      J1K1L1=K1L1*NJ1
      DO 28 I2=1,NI2
      IA2=(I2-1)*J2K2L2
      D1T=2.-SL(1,N,I2)
      DO 27 J2=1,NJ2
      IB2=(J2-1)*K2L2
      D2T=3.-SL(2,N,J2)
      DO 26 K2=1,NK2
      IC2=(K2-1)*NL2
      DO 25 L2=1,NL2
      M2=IA2+IB2+IC2+L2
      DO 24 I1=1,NI1
      D1N=SL(1,N-1,I1)+D1T
      IF(D1N.LT.DMIN(1).OR.D1N.GT.DMAX(1))GO TO 24
      IA1=(I1-1)*J1K1L1
      DO 23 J1=1,NJ1
      D2N=SL(2,N-1,J1)+D2T
      IF(D2N.LT.DMIN(2).OR.D2N.GT.DMAX(2))GO TO 23
      IB1=(J1-1)*K1L1
      DO 22 K1=1,NK1
      D3N=SL(3,N-1,K1)+D2N-SL(3,N,K2)
      IF(D3N.LT.DMIN(3).OR.D3N.GT.DMAX(3))GO TO 22
      IC1=(K1-1)*NL1
      DO 21 L1=1,NL1
      D4N=SL(4,N-1,L1)+D3N+D1N-SL(4,N,L2)
      IF(D4N.LT.DMIN(4).OR.D4N.GT.DMAX(4))GO TO 21
      M1=IA1+IB1+IC1+L1
      IF((-1.E5).EQ.FNM1(M1))GO TO 21
      B=C1(N-1)*D1N+C2(N-1)*D2N+C3(N-1)*D3N+C4(N-1)*D4N+FNM1(M1)
      IF(B.LE.FN(M2))GO TO 20
      FN(M2)=B
      U(M2)=D1N
      U(NDIM2+M2)=D2N
      U(2*NDIM2+M2)=D3N
      U(3*NDIM2+M2)=D4N
   20 CONTINUE
   21 CONTINUE
   22 CONTINUE
   23 CONTINUE
   24 CONTINUE
   25 CONTINUE
   26 CONTINUE
   27 CONTINUE
   28 CONTINUE
      MM=4*NDIM2
      DO 31 M2=1,MM
   31 FU(N,M2)=U(M2)
 4002 FORMAT(1X,I2,20F5.2)
 4003 FORMAT(1X,24F5.2)
      DO 29 M2=1,NDIM2
      FNM1(M2)=FN(M2)
   29 CONTINUE
   30 CONTINUE
      RETURN
      END
      SUBROUTINE STATE(NN)
      COMMON S(4,13),DS(4,13),SMIN(4,13),SMAX(4,13),SL(4,13,3),NNS(4,13)
     1,SI(4),SF(4),DMIN(4),DMAX(4),C1(12),C2(12),C3(12),C4(12),FN(81),U(
     2324),D(4,13),FU(13,324)
      DO 205 N=1,NN
      DO 204 IS=1,4
      FLAG=0.
      SD=S(IS,N)-DS(IS,N)
      IF(SD.LT.SMIN(IS,N))SD=SMIN(IS,N)
      IF(SD.EQ.S(IS,N))GO TO 200
      SL(IS,N,1)=SD
      SL(IS,N,2)=S(IS,N)
      FLAG=1.
      GO TO 201
  200 SL(IS,N,1)=S(IS,N)
  201 SU=S(IS,N)+DS(IS,N)
      IF(SU.GT.SMAX(IS,N))SU=SMAX(IS,N)
      IF(SU.EQ.S(IS,N))GO TO 202
      IF(FLAG.EQ.1.)GO TO 203
      SL(IS,N,2)=SU
  202 NNS(IS,N)=2
      GO TO 204
  203 SL(IS,N,3)=SU
      NNS(IS,N)=3
  204 CONTINUE
  205 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TRACE(NN,MNN,NNM1)
      COMMON S(4,13),DS(4,13),SMIN(4,13),SMAX(4,13),SL(4,13,3),NNS(4,13)
     1,SI(4),SF(4),DMIN(4),DMAX(4),C1(12),C2(12),C3(12),C4(12),FN(81),U(
     2324),D(4,13),FU(13,324)
      DIMENSION NOD(4)
      DO 10 IS=1,4
   10 S(IS,NN)=SF(IS)
      DO 15 NR=1,NNM1
      N=NN+1-NR
      NDIM=NNS(1,N)*NNS(2,N)*NNS(3,N)*NNS(4,N)
      MM=4*NDIM
      DO 13 IS=1,4
      NNOD=NNS(IS,N)
      DO 11 I=1,NNOD
      II=I
      IF(S(IS,N).EQ.SL(IS,N,I))GO TO 12
   11 CONTINUE
   12 NOD(IS)=II
   13 CONTINUE
      M=(NOD(1)-1)*NNS(2,N)*NNS(3,N)*NNS(4,N)+(NOD(2)-1)*NNS(3,N)*NNS(4,
     1N)+(NOD(3)-1)*NNS(4,N)+NOD(4)
      IF(N.EQ.NN)MNN=M
      D(1,N-1)=FU(N,M)
      D(2,N-1)=FU(N,NDIM+M)
      D(3,N-1)=FU(N,2*NDIM+M)
      D(4,N-1)=FU(N,3*NDIM+M)
      PRINT 1000,N,M,FU(N,M),FU(N,NDIM+M),FU(N,2*NDIM+M),FU(N,3*NDIM+M)
 1000 FORMAT(1X,2I5,4F10.2)
      S(1,N-1)=S(1,N)+D(1,N-1)-2.
      S(2,N-1)=S(2,N)+D(2,N-1)-3.
      S(3,N-1)=S(3,N)+D(3,N-1)-D(2,N-1)
      S(4,N-1)=S(4,N)+D(4,N-1)-D(3,N-1)-D(1,N-1)
   15 CONTINUE
   16 CONTINUE
      RETURN
      END
```

APPENDIX B. NOTATIONS

a(o); a(N)	= m-dimensional vectors specifying the desired states of system at t_0 and t_f respectively
b(n)	= return function for a time increment starting at stage n
BIR(n)	= return from irrigation activities during a time increment starting at stage n
BPR(n)	= return from hydropower generation during a time increment starting at stage n
c; e	= discrete levels in state vector
C_k	= corridor formed by all D(n), n = 0,1, ..., N for k-th iteration
CT	= capacity of turbines in kw
CPP1, CPP2	= capacities of power plants 1 and 2 in Mw
D(n)	= sub-domain formed at stage n
E{ }	= expected value of the terms in the bracket
EO	= annual target energy output in Kw hr
f	= load factor
F	= sum of the returns for N time increments
F^*	= optimum sum of the returns for N time increments
$\overline{F}$	= sum of the returns for N time increments due to $\overline{u}$ and $\overline{s}$
F_s	$= (\frac{\partial F}{\partial s_1}, \frac{\partial F}{\partial s_2}, \ldots, \frac{\partial F}{\partial s_m})$
F_s^*	$= (\frac{\partial^2 F^*}{\partial s_i \partial s_j}, i, j = 1,2, \ldots, m)$
g	= a penalty function
hy	= number of hours in a year
h.o.t.	= higher order terms
H_j	= number of discrete levels in the j-th component of decision vector
L_i	= number of discrete levels in the i-th component of state vector

m	= the order of the system, i.e., the number of state variables
n	= beginning of a time increment called a stage
N	= total number of time increments in the time horizon
$p[y(n), v]$	= probability of the v-th discrete level of y during a time increment starting at stage n
q	= number of decision variables in the system
R	= return from the system in one time increment
$s(n)$	= an m-dimensional state (storage) vector at stage n
$s^{*}(n)$	= vector of the optimal state at stage n
$\overline{s}(n)$	= vector of the trial state at stage n
$S(n)$	= an m-dimensional vector representing the admissible state domain at stage n
$\dot{s}$	$= \frac{d}{dt}[s,u,t]$
t	= time
t_0	= beginning of the time horizon
t_f	= end of the time horizon
T	= total number of assumed increments from the state domain
$u(n)$	= a q-dimensional decision (release) vector at stage n
$u^{*}(n)$	= vector of the optimal decision at stage n
$\overline{u}(n)$	= vector of the trial decision at stage n
$U(n)$	= q-dimensional vector representing the admissible decision domain at stage n
V	= number of discrete levels in the probability space of y
$y(n)$	= inflow during the time increment starting at stage n
z	= a discrete level in decision vector
$\alpha(n)$	= a constant between 0 to 1 representing the fraction of the irrigation water return to the system
$\beta(n)$	= irrigation demand relative to the annual irrigation demand during a time increment starting at stage n

$\gamma(n)$	= a value between 0 and 1 describing the importance or irrigation water demand during a time increment starting at stage n as compared to the maximum irrigation demand during any period
$\delta s_{i,j}(n)$	= j-th component of the m-dimensional incremental vector $\Delta s_i(n)$ at stage n
$\delta u(t)$	= change in decision vector at time t
Δt	= an increment of time
$\Delta s_i(n)$	= the i-th m-dimensional incremental vector formed at stage n
ε	= a constant
η	= a dummy variable representing time
θ; ϕ	= function describing the dynamic behavior of the system
$\sigma_{j,t}$	= value of t-th assumed increment for j-th state variable in the state domain
ψ	= a function presenting decision as a function of states only