UM-17

OPTIMUM RESERVOIR OPERATION USING DYNAMIC PROGRAMMING

SATISH CHANDRA

DIRECTOR

STUDY GROUP

P V SEETHAPATHI

S K JAIN

NATIONAL INSTITUTE OF HYDROLOGY
JAL VIGYAN BHAVAN
ROORKEE-247667(UP) INDIA

1985-86

CONTENTS

LIST OF FIGURES

ABSTRACT

Dynamic programming is an optimization technique which has been extensively used for solution of various problems associated with water resources systems. As the original technique had several short comings, a number of modifications have been proposed to overcome the limitations, particularly the curse of dimensionality. One modification, known as discrete differential dynamic programming is particularly suited for the problems related with water resources systems.

A computer programme has been developed for optimization of operation of a reservoir using the technique of discrete differential dynamic programming (DDDP). This programme alongwith the theory of DDDP has been presented in this report. The application of the programme to a hypothetical case is also given to illustrate the programme usage.

1.0 INTRODUCTION

The analysis of a water resources system is a complex problem because of a large number of interacting factors. Among the techniques which are generally used for this purpose now-a-days, optimization techniques are perhaps most common. During the last two decades, significant advances have been made in the optimization techniques and presently these techniques are extensively used for planning, design and operation of water resources projects. The main motivation of search for better techniques for analysis of water resources systems has been the realization of the fact that even a small improvement in the solution of the related problems has high economic value attached to it. Further, the advent of modern fast computer has made it very easy to use these tools and the solution can be obtained quite quickly.

Among the available optimization techniques, two techniques are extensively used for obtaining solution of the problems associated with water resources systems. These are linear programming and dynamic programming. Linear programming problems are those optimization problems whose objective function and constraints are linear functions of decision variables. Once these conditions are satisfied, a very efficient solution technique called simplex method is available which can be applied to any linear programming problem. Now-a-days generalized efficient computer codes are available for solution of linear programming problems. All that a user has to do is to translate the problem in the required form and present the data in the prescribed format. This has been a very big incentive for widespread use of linear programming. Although many physical phenomena are nonlinear in

nature, linear programming has been successfully applied by piecewise linearization of such processes.

The other solution technique which has been extensively used for solution of problems associated with water resources systems is dynamic programming. Dynamic programming is basically an enumeration technique and can be used for objective functions which are linear, nonlinear and even discontinuous. Because of the inherent nature of the technique, generalized computer programmes are not available for it. However, the extensive application of the technique reported in the literature is a proof of the fact that the efforts involved in developing and testing computer programmes for specific applications are insignificant, compared with the benefits. One big advantage in using dynamic programming over other techniques is because of the way constraints are handled in this technique. In the other optimization techniques, the constraints lead to additional computations. However, in dynamic programming the constraints can be utilized for increased computational efficiency since they limit the feasible region.

1.1 Objective and Scope of Present Work

In the present study, a computer programme has been developed which can be used to derive optimal operating policy of a reservoir. This programme alongwith the theory is presented in this report. Sufficient details about the programme are given in the report so that a user can use the programme following them. A case study is also given to substantiate the use of the programme. A listing of the programme alongwith sample input and output is given in the Appendix 1, 2, and 3.

2.0 DYNAMIC PROGRAMMING

Dynamic programming is an enumerative technique developed by Bellman in 1953. This technique was developed to optimize a problem which can be represented as a multistage decision process. The entire formulation of dynamic programming is based upon the Bellman's principle of optimality (ref.figure 1). An optimal policy has the property that 'whatever the initial state and decisions are, the remaining decisions must constitute an optimal policy with respect to the state resulting from the first decision'. The proof can be obtained by contradiction. If the optimal path for going from A to C is I-II then the optimal path from B to C will be II and not II'. In the problem formulation, the dynamic behaviour of the system is expressed by using three types of variables, as described below:

a. State variables - which define the condition of the system. For example, in studies dealing with reservoirs, the amount of water stored in the reservoir may represent its state.

b. Stage variables - which define the order in which events occur in the system. Most commonly, time is taken to be the stage variable. There must be a finite number of possible states at each stage.

c. Control variables - which represent the controls applied at a particular stage and transform the state of the system. For the reservoir operation problem, the release of water from the reservoir is a typical decision variable.

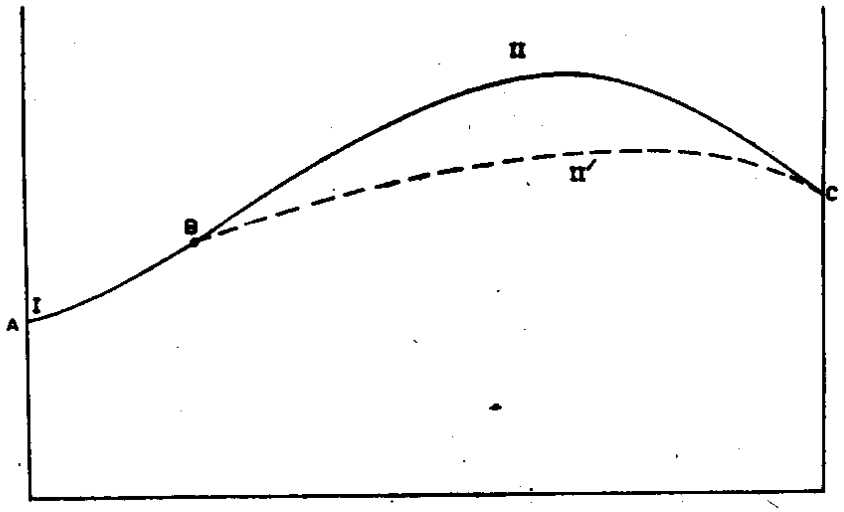

Fig.1 - Illustration of the Principle of Optimality

With each state transformation, a return is associated which may either represent benefits or costs. The state of reservoir will be transformed by releasing a certain amount of water from it. This water can be used for some useful purpose like irrigation and will lead to monetary returns. The water released from a reservoir may also cause flood damages downstream and hence a cost can be associated with these damages. The problem is to find the control variables which optimize the returns. Typically the benefits are maximized and the costs are minimized. The optimal decision made at a particular stage is independent of decisions made at previous stages given the current state of the system. A set of decisions for each time period is called a policy. The particular policy which optimizes the objective function is called the optimal policy. The set of states which result from the application of a policy is called the state trajectory.

The dynamic behaviour of the system is expressed by an equation known as the system equation. It can be written in discrete form as:

$$s(t+1) = f[s(t),u(t),t] \qquad \ldots (1)$$

$$t = 1,2,.....N$$

where s(t) is the state variable at time t,

u(t) is the control applied at time instant t, which last for a duration Δt, Δt, being the length of time in which stage variable is discretized, and

f is the given functional form.

The state of the system at any stage t should lie in the domain of admissible states at that stage. Similarly the control at any stage should also lie in the admissible domain at that stage.

$$s(t) \in S(t) \qquad \ldots (2)$$

$$u(t) \in U(t) \qquad \ldots (3)$$

Where S(t) and U(t) are the domains of admissible states and controls at stage t.

Let R[s(t), u(t),t] be the return obtained if the system is at state s(t) at stage t and the control u(t) is applied at instant t lasting for a duration Δ t. Further, let F[s(N),N]be the sum of returns from application of controls from some initial stage at t = 0 to final stage at t = N. The objective of maximizing the sum of returns from the system can be expressed as

$$\text{Max } F[s(N), N] \qquad \ldots (4)$$

Let the state of system at t = 0, $s(0) \in S(0)$ is known and the returns F[s(0),0] are also known. Let F*[s(0),0] be the optimum value of these returns. Now consider the first stage (of duration Δ t). The optimal return for this period is given by

$$F^*[s(1),1] = \max_{u(0) \in U(0)} R[s(0),u(0),0] + F^*[s(0),0] \qquad \ldots (5)$$

This equation is solved for each discrete level of state at t = 1 as a function of control variables u(0). To do this, the state is discretized into a number fo discrete levels (ref. figure 2). Now a particular lettice point is chosen and all the admissible levels of decision variables which lead to this state are chosen. For each of these decision variables, the return F[s(1),1] is calculated. The maximum among these returns given the value of F*[s(1),1]. This

computation is repeated for each discrete value of s(1) and the results are stored.

The computations are performed in similar fashion for stage 2,3.....N. The recursive equation for any stage t can be written as

$$F^*[s(t),t] = \max_{u(t-1) \in U(t-1)} R[s(t-1), u(t-1), t-1] + F^*[s(t-1), t-1] \quad ...(6)$$

Thus at the end of N^{th} stage, the values of $F^*[s(t),t]$, t=1,2.....N are available. The optimal value of control variables or the optimal policy is obtained by tracing back the values of returns, corresponding to those states which satisfy the initial and final values and the constraints. The optimal state trajectory can be determined by using the system equation once the optimal policy is known.

The above computational scheme of dynamic programming is known as the forward algorithm since the computations start at the initial value of the state variable at stage 1 and move forward. In contrast to this, the computations can also commence at the final value of state variable at the last stage and can move backwards. The optimal policy is retrieved by tracing forward from the returns. This algorithm is called the backward algorithm.

2.1 Advantages and Disadvantages of Dynamic Programming

Dynamic programming is essentially an enumerative technique which is specially suited to multistage decision problems. There are a number of advantages in using this technique particularly for analysis of a water resources system. Some of the advantages are:

i) The dynamic programming formulation is same for linear as

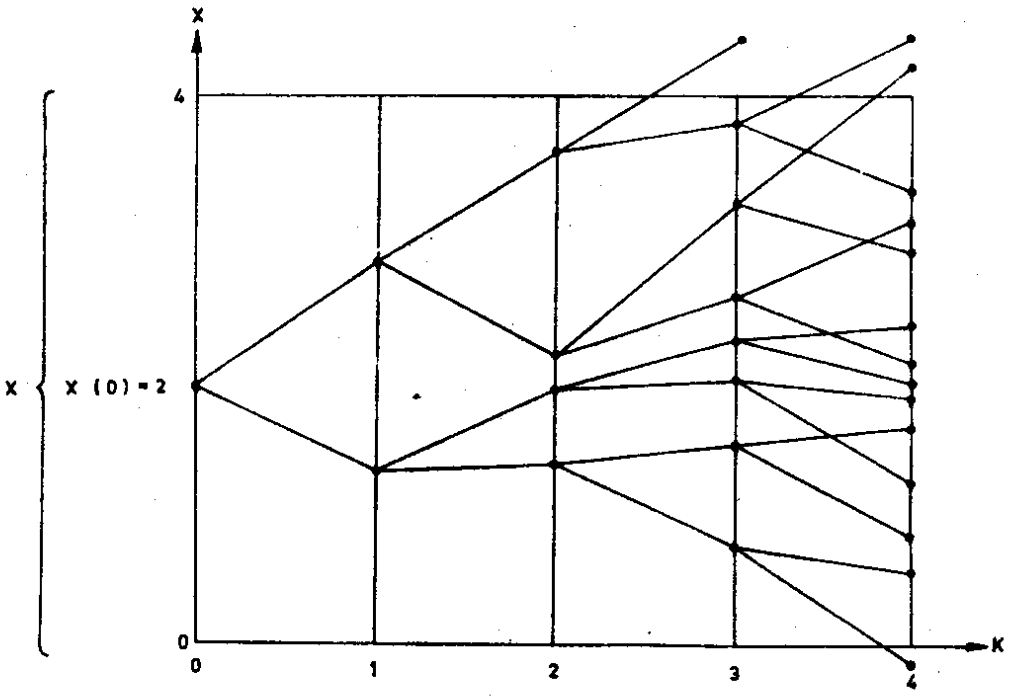

Fig. 2 - Tree Generated by Enumeration

well as nonlinear problems. Thus, no extra effort is required for nonlinear problems. This property is very useful in case of water resources systems since many related problems cannot be realistically linearized.

ii) The incorporation of constraints in linear and nonlinear programming problems is more difficult than in dynamic programming problems. In case of dynamic programming, the constraints serve a useful purpose. They do limit the feasible region and thus may lead to reduction in computational time requirement.

iii) The stochastic nature of a problem can be easily considered in the dynamic programming formulation. The algorithm developed for a deterministic problem does not have to be significantly changed to incorporate stochasticity. This is in contrast with other techniques where incorporation of stochasticity requires too much change in the algorithm and significant increase in computational time.

Besides the above advantages, there are some disadvantages in using the dynamic programming formulation. These include:

i) The dynamic programming is not basically tailored in such a fashion that generalized programmes can be written using it. Thus a new computer programme has to be developed or an existing progrmme has to be significantly modified and tested for each new application of the technique. On the other hand standard computer programmes are widely available for the linear programming technique.

ii) It was stated above that to solve a particular problem, the state and control variables are discretized at each stage and these discretized values are then used to compute returns. Thus for the purpose of computation, these values have to be stored in the computer memory from where they can be drawn as and when required. The number of

the knowledge of one variable enables the determination of the other. The initial values must always satisfy the constraints.

Now a set of incremental values of stage variable is assumed. When these incremental values are added and substracted from the trial trajectory at a particular stage, a subdomain is formed around the trial trajectory. This subdomain is called a corridor. At this stage optimization is performed constrained to this corridor and a better value of the trajectory is found. For the next iteration, this trajectory is considered as the trial trajectory. The computations are performed by varying the composition of the corridor in such a way that the algorithm converges towards the optimal solution.

One such corridor is shown in figure 3. In this case the trial trajectory lies at the centre of the corridor though this is not a necessary condition. More than one quantized states on either side of the trajectory may be chosen but the choice of three quantized states at each stage is most suitable for computational efficiency.

To obtain good convergence, two criterion were suggested by Hall (1968). These are guidelines about the increments to the state vector. The first is that the increments to the state variables must be kept small and constant throughout any iteration. The second is that the size of increments should be reduced as the iterations proceed. However, the size of increments should be chosen such that entire feasible region could be inspected if required. There is a strong correlation between the number of iterations required for good convergence and the size of increments at each iteration. It was also suggested by Yeh (1982) that several iterations with a small increment should be allowed at the end of each computation cycle to improve the value of objective function.

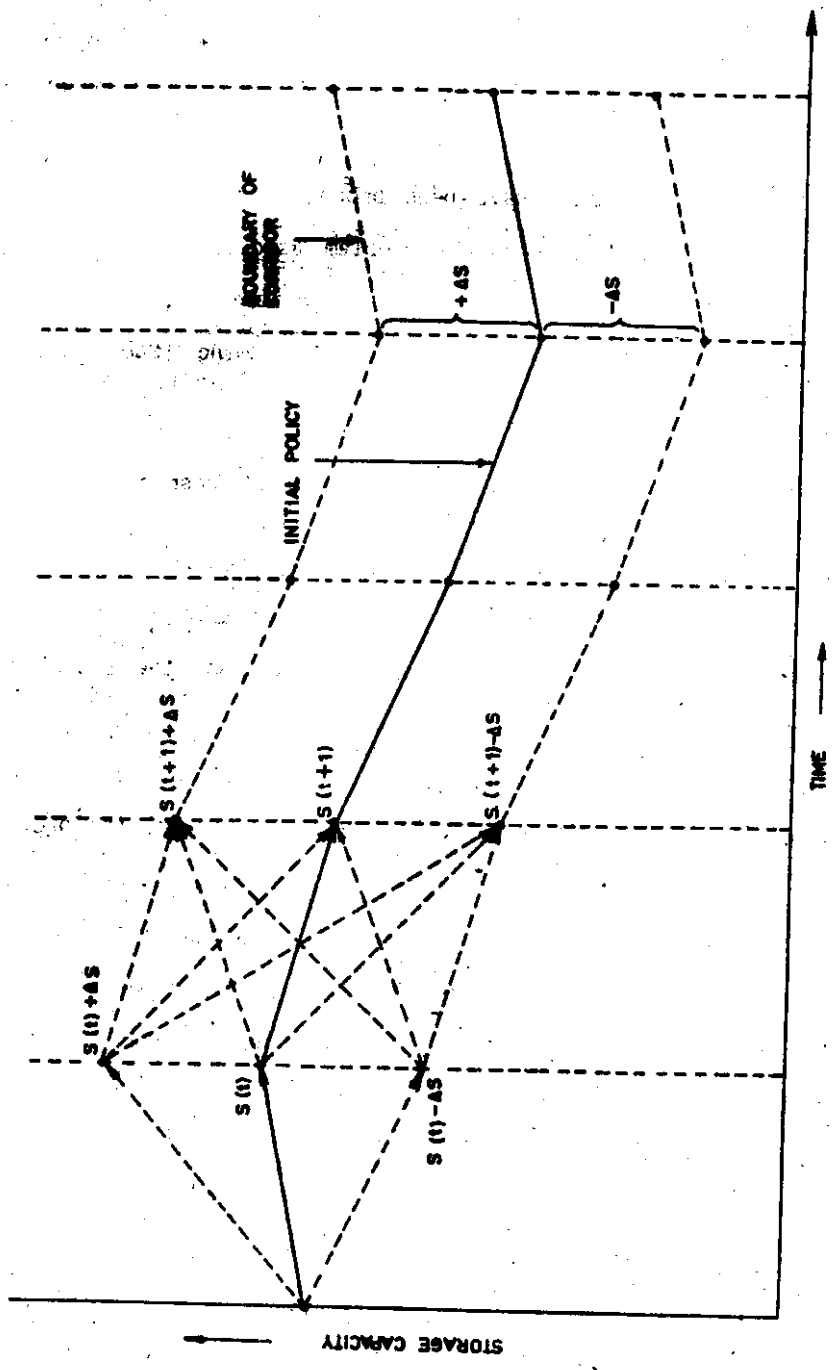

Fig.3 - Discrete Differential Dynamic Programming

discretized values goes on increasing with the fineness of the discretization. For large problems, the memory requirement becomes a major limitation. This requires judicious choice to be made for the accuracy requirement, computer memory available and computational time available. Chow et al (1975) have discussed in detail the computer time and memory requirements for analysis of water resources system.

Several techniques have been proposed by different investigators to reduce the dimensionality problem associated with analysis of water resources systems using dynamic programming technique. One of these, the Discrete Differential Dynamic Programming (DDDP) proposed by Heidari et al (1971) is being described here.

2.2 Discrete Differential Dynamic Programming Technique

The Discrete Differential Dynamic Programming is an iterative procedure in which the recursive equation of dynamic programming is solved within a restricted set of quantized values of the state variables. The optimal solution is obtained by gradually improving the initial solution. This technique is particularly suitable for invertible systems. A system is called invertible if for that system, the order of the state vector is equal to the order of the control vector. Thus the knowledge of stage variables enables one to compute the decision variables. The water resources systems are mostly invertible. For example, assuming that the inflows to a reservoir are known, the releases from it can be determined if the states of the reservoir at different times are known.

The DDDP computations start with either a known stage trajectory or a known policy. Because of the property of invertibility,

A computer programme has been developed for determining the optimal operating policy of a reservoir using the DDDP technique. In the beginning the computations are carried out using the DDP procedure. A rough grid is chosen for this purpose and the optimization yields the optimal trajectory which is used as the initial trajectory for the DDDP computations. The optimum trajectory is obtained after carrying out the DDDP computations. The programme is described in the following article. A listing of the programme is given in Appendix 1.

3.0 DESCRIPTION OF PROGRAMME

3.1 Main Programme

The main programme reads the complete input data from a file called BAK.INP. After initializing few variables, it calls the sub-routine DDP. Using the values returned by the subroutine DDP, the computations of discrete differential dynamic programme (DDP) are taken up. After each stage of DDP computations, the corridor width is halved Further at each stage, the computation cycle is repeated NITER times while keeping the width of the corridor as fixed. A backward computation procedure is adopted here. The results are printed out from the main programme before terminating the programme execution.

3.2 Subroutine DDP

This is the subroutine which carries out the discrete dynamic programming computations. The trasfer of data from main programme to this subroutine and back is mostly through common blocks. For the purpose of computations, the entire active storage region is divided into NDV number of divisions. The optimal state trajectory is searched from amongst the feasible states so that the objective function is maximized The optimal state trajectory so chosen becomes the initial trajectory for computations in main programme.

3.3 Subroutine BENEF

This is a user supplied subroutine which evaluates the objective function. Everytime there is a change in objective function and/or the constraints, this sub-routine has to be modified.

The constraints are incorporated by assigning a very high negative value to the function whenever a constraint is violated.

3.4 Function FINT

This function is used for linear interpolation to find the value of variable say x corresponding to the given value of y using the table of pairs of x & y values.

The organization of the program is shown in figure 4. The transfer of data to subroutines is mainly through common blocks as given in figure 5.

3.5 Programme Implementation

The programme has been developed on VAX-11/780 system with Fortran-77 compiler. While developing the programme, it was kept in mind that the programme should be sufficiently generalized in the sense that the required changes to implement it on other systems are minimum.

However, following statements might require change for susccessful running of the programme on a system other than on which it has been developed :

1. The open statement may need modification.
2. The character declaration may have to be changed.
3. Changes may also be required in Format specifiers.

3.6 Description of variables

Following is the list of important variables and their description which have been used in this programme. This list includes the variables used for input and output and subroutine calls.

Variable	Description
AINF	Inflow to reservoir (in cumecs)
AREA	Reservoir surface area (in square meters)
DEM	Demand of water from reservoir (in cumecs)
DTIM	Computation time interval (in hours)
ELEV	Reservoir elevation (in m)
NDV	Number of increments in which active reservoir storage capacity is divided
NITER	Number of iterations to be performed keeping the size of corridor fixed.
NN	No. of set of values in elevation area-storage-release capacity table.
NP	Number of computational periods
OREL	Optimal release from reservoir (in cumecs)
REL	Release from reservoir (in cumecs)
RELC	Maximum possible release at a particular elevation (in cumecs)
SMAX	Maximum reservoir storage capacity(in cubic meters)
SMIN	Minimum reservoir storage or dead storage(in cubic meters)
STIN	Initial reservoir content (in cubic meters)

3.7 Input Specification

The input details are read from a file called BAK.INF which is assigned a logical unit number 2. For ease of keying in, almost all input is through free format. The input file organisation has to be as given below:

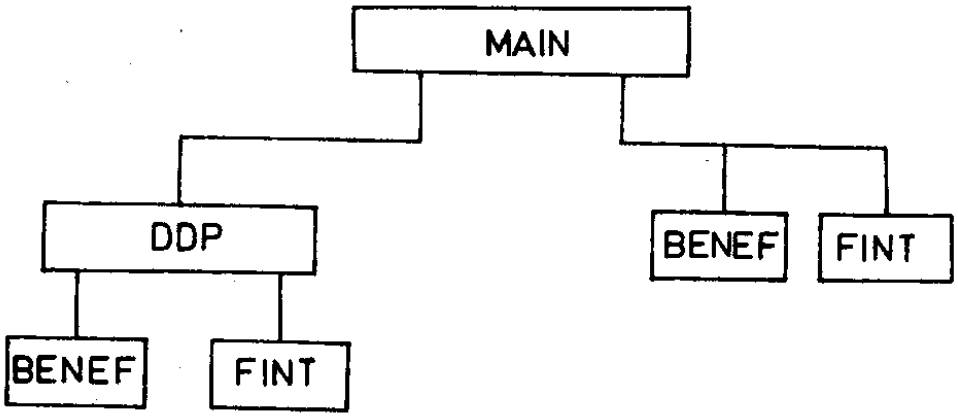

Fig. 4 - Programme Organisation

BLOCK COMMON

SUBPROGRAMMES

	C1	RES	IO	INT
MAIN	X	X	X	X
DDP	X	X	X	X
BENEF	X	X	X	X
FINT				

Fig.5 - Common Block Usage

Line No.(s)	Variable(s)	Format	Description
1	TITLE	A	Title of the problem (maximum 80 characters long)
2	NP DTIM NN SMAX SMIN NDV NITER	Free	Mainly control variables, also information about maximum and minimum storages allowed.
3 to NN+2	ELEV AREA STOR RELC	Free	Reservoir elevation area-storage release capacity table. Should be given in tabular form for ease of checking. NN values are needed.
NN+3	STIN	Free	Initial reservoir storage.
As required	AINF	Free	NP values of reservoir inflows, starting from first period.
As required	DEM	Free	NP values of water demand starting from first period.

All the input data must be in MKS system. Time is given in hours.

3.8 Programme Output

The output from the programme contains all input information and the optimal releases from the reservoirs. The output in order as typed is

a. Number of time periods.

b. Computational time interval.

c. Maximum and minimum storage capacities.

d. Elevation-area-capacity-release capacity tables.

e. Following information for each computational period-initial storage, inflow, demand,release and final storage.

The programme has been applied to a hypothetical case to demonstrate the use of programme. The objective function chosen is given is subroutine BENEF. The input and output for this hypothetical case are given in Appendix 2 and 3 respectively.

4.0 CONCLUSIONS

A computer programme has been developed which can be used to optimize operation of a single reservoir. This programme, which is based on discrete differential dynamic programming approach has been described and listed in this report. The programme has been described in detail so that a user can use it following the description. The programme listing alongwith sample input and output is also given in the report.

REFERENCES

1. Bellman,R.E., and S.E. Dreyfus, Applied Dynamic Programming, Princeton University Press, Princeton, N.J.,1962.

2. Buras,N.,Dynamic Programming in Water Resources Developement, Advances in Hydroscience, (ed)V.T.Chow, Vol. 3,Academy Press, N.Y.,1966.

3. Chow,V.T., D.R. Maidment, and G.W. Tauxe,'Computer time and memory requirement for DP and DDDP in water resources analysis' Water Resources Research, 11(6),621-628, 1975.

4. Hall, W.A., W.S.Butcher, and A.Esogbue, 'Optimization of the operation of a Multi-purpose reservoir by dynamic programming' Water Resources Research, 4(3),471-477,1968.

5. Heidari,M.,V.T. Chow, P.V. Koktovic, and D.D. Meredith, 'Discrete differential dynamic programming approach to water resources systems optimization', Water Resources Research, 7(2),273-282, 1971.

6. Larson, R.E., State Increment Dynamic Programming, American Elsevier Publishing Company Inc.,New York, 1968.

7. Taha, H.A., Operations Research, an Introduction, Macmillan Publishing Company, New York, 1976.

8. Yakowitz, S., 'Dynamic programming applications in water resources ',Water Resources Reserch, 18(4), 673-696, 1982.

9. Yeh, W.W-G.,State of the Art Review: Theories and Applications of Systems Analysis Techniques to the Optimal Management and Operation of a Reservoir System, Rep.No.UCLA-ENG-82-52, School of Engineering and Applied Science, UCLA, Los Angeles, 1982.

10. Yeh, W.W-G.,'Reservoir Management and Operation Models: A state of the art review', Water Resources Research, 21(2), 1797-1818, 1985.

11. Young, G.K.,'Finding reservoir operating rules', Journal of the Hydraulics Division, ASCE,93(HY6),297-321,1967.

APPENDIX - 1
COMPUTER PROGRAMME

```
C*********************************************************************
C       INCREMENTAL DYNAMIC PROGGRAMMING FOR ONE RESVOIR OPTIMIZATION
C*********************************************************************
        COMMON/C1/ BEN(20,20),STBO(30,20),TPRO(30,20),OPS(20,20),
       1IP(30,20),EPS(30,20),OBEN(30,20)
        COMMON/RES/ SMAX,SMIN,STIN(50),ELEV(25),AREA(25),STOR(25),RELC(25)
        COMMON/IO/  AINF(50),REL(30,20,20),OREL(50),DEM(50)
        COMMON/INT/ NP,NN,NDV,DTIM
        CHARACTER*80 TITLE
C*********************************************************************
        OPEN(UNIT=2,FILE='BAK.INP')
        READ(2,1) TITLE
 1      FORMAT(A)
        READ(2,*) NP,DTIM,NN,SMAX,SMIN,NDV,NITER
        READ(2,*) (ELEV(I),AREA(I),STOR(I),RELC(I),I=1,NN)
        READ(2,*) STIN(1)
        READ(2,*) (AINF(I),I=1,NP)
        READ(2,*) (DEM(I),I=1,NP)
C*********************************************************************
        IDELS=(SMAX-SMIN)/NDV
        DELS=IDELS
        DTIM=DTIM*3600
C*******************************                                  D D P
        CALL DDP(DELS)
C*******************************
        DO 200 IT=1,NITER
         DELS=DELS*0.5
         DO 20 I=1,NP
         DO 20 J=1,3
          STBO(I,J)=STIN(I+1)+DELS*(2-J)
          IF(STBO(I,J).GT.SMAX) STBO(I,J)=SMAX
```

```
          IF(STBO(I,J).LT.SMIN) STBO(I,J)=SMIN
20        CONTINUE
         DO 100 I=NP,2,-1
          DO 40 K=1,3
          DO 40 J=1,3
           REL(I,J,K)=(STBO(I-1,J)-STBO(I,K))/DTIM+AINF(I)
           AVST=(STBO(I-1,J)+STBO(I,K))/2
           RMAX=FINT(STOR,RELC,AVST,NN)
           CALL BENEF(I,J,K,AVST)                                    B E N E F
           IF(REL(I,J,K).GT.RMAX) BEN(J,K)=-1.E+20
           IF(REL(I,J,K).LT.0) BEN(J,K)=-1.E+20
40        CONTINUE
          DO 55 K=1,3
           KO=1
           IF(BEN(K,2).GT.BEN(K,1)) KO=2
           IF(BEN(K,3).GT.BEN(K,KO)) KO=3
           IP(I,K)=KO
           OBEN(I,K)=BEN(K,KO)
55        CONTINUE
100      CONTINUE
         DO 30 K=1,3
          REL(1,2,K)=(STIN(1)-STBO(1,K))/DTIM+AINF(1)
          AVST=(STIN(1)+STBO(1,K))/2
          RMAX=FINT(STOR,RELC,AVST,NN)
          CALL BENEF(1,2,K,AVST)                                     B E N E F
          IF(REL(1,2,K).GT.RMAX) BEN(2,K)=-1.E+20
          IF(REL(1,2,K).LT.0) BEN(2,K)=-1.E20
30        CONTINUE
         KO=1
         IF(BEN(2,1).LT.BEN(2,2)) KO=2
         IF(BEN(2,3).GT.BEN(2,KO)) KO=3
         OREL(1)=REL(1,2,KO)
         STIN(2)=STBO(1,KO)
         DO 150 I=2,NP
          KE=IP(I,KO)
          STIN(I+1)=STBO(I,KE)
          OREL(I)=REL(I,KO,KE)
          KO=KE
150      CONTINUE
200      CONTINUE
```

```
C*********************************************************************
      TYPE 900,TITLE
900   FORMAT(///A//20X,'*** I N P U T   D A T A  ***'////5X
     1' All data are in MKS units')
      TYPE 901,NP,dtim/3600
901   FORMAT(/5x'The number of time periods for analysis are :'i3/
     15x,'The computational time interval is : 'f4.1' hours')
      TYPE 902,SMAX,SMIN
902   FORMAT(/5X'Maximum storage capacity is : 'f14.3' cum'/
     15X'Minimum storage capacity is : 'f14.3' cum')
      TYPE 903,(I,ELEV(I),AREA(I),STOR(i),RELC(i),I=1,NN)
903   FORMAT(/10x,'Elevation - Area - Capacity - Release Capacity Table'/
     15x,' S N     Elevation      Area       Capacity      Rel Capacity'
     25x,'           ( m )       (sqm)       ( cum )       ( cumecs)'//
     2,(5x,i4,4f14.2))
1000  TYPE 904
904   FORMAT(///20x' ****   Results of Calculations   ****'//
     1,'  Period     Init Storage       Inflow        Demand
     2       Release    Final Storage'/
     3                ( cum )          (cumecs)      (cumecs)
     4       (cumecs)      ( cum )'/)
      DO 160 I=1,NP
160   TYPE 910, I,STIN(I),AINF(I),DEM(I),OREL(I),STIN(I+1)
910   FORMAT(I3,4X,4F14.3,3X,F14.3)
      STOP
      END

C*********************************************************************
      SUBROUTINE DDP(DES)
C*********************************************************************
      COMMON/C1/ BEN(20,20),STBO(30,20),TPRO(30,20),OPS(20,20),
     1IP(30,20),EPS(30,20),OBEN(30,20)
      COMMON/RES/ SMAX,SMIN,STIN(50),ELEV(25),AREA(25),STOR(25),RELC(25)
      COMMON/IO/  AINF(50),REL(30,20,20),OREL(50),DEM(50)
      COMMON/INT/ NP,NN,NDV,DTIM
C*********************************************************************
      DO 20 I=1,NP
       STBO(I,1)=SMIN
       DO 20 J=2,NDV
       STBO(I,J)=STBO(I,J-1)+DES
20    CONTINUE
```

```
      DO 100 I=NP,2,-1
       DO 40 J=1,NDV
       DO 40 K=1,NDV
        REL(I,J,K)=(STBO(I-1,J)-STBO(I,K))/DTIM+AINF(I)
        RMAX=FINT(STOR,RELC,AVST,NN)
        AVST=(STBO(I-1,J)+STBO(I,K))/2
        CALL BENEF(I,J,K,AVST)                                  B E N E F
        IF(REL(I,J,K).GT.RMAX) BEN(J,K)=-1.E+20
        IF(REL(I,J,K).LT.0) BEN(J,K)=-1.E20
40     CONTINUE
       DO 55 J=1,NDV
       KO=1
       DO 54 K=2,NDV
54     IF(BEN(J,K).GT.BEN(J,KO)) KO=K
       IP(I,J)=KO
       OBEN(I,J)=BEN(J,KO)
55     CONTINUE
100   CONTINUE
      DO 30 K=1,NDV
       REL(1,2,K)=(STIN(1)-STBO(1,K))/DTIM+AINF(1)
       AVST=(STIN(1)+STBO(1,K))/2
       RMAX=FINT(STOR,RELC,AVST,NN)
       CALL BENEF(1,2,K,AVST)                                   B E N E F
       IF(REL(1,2,K).GT.RMAX) BEN(2,K)=-1.E+20
       IF(REL(1,2,K).LT.0) BEN(2,K)=-1.E20
30    CONTINUE
      KO=1
      DO 44 K=2,NDV
44    IF(BEN(2,K).GT.BEN(2,KO)) KO=K
      STIN(2)=STBO(1,KO)
      OREL(1)=REL(1,2,KO)
      DO 150 I=2,NP
       KE=IP(I,KO)
       STIN(I+1)=STBO(I,KE)
       OREL(I)=REL(I,KO,KE)
       KO=KE
150   CONTINUE
      RETURN
      END
```

```
C*********************************************************************
      SUBROUTINE BENEF(IT,IJ,IK,AVST)
C*********************************************************************
      COMMON/C1/ BEN(20,20),STBO(30,20),TPRO(30,20),OPS(20,20),
     1IP(30,20),EPS(30,20),OBEN(30,20)
      COMMON/RES/ SMAX,SMIN,STIN(50),ELEV(25),AREA(25),STOR(25),RELC(25)
      COMMON/IO/  AINF(50),REL(30,20,20),OREL(50),DEM(50)
      FF=100000.0
      BEN(IJ,IK)=REL(IT,IJ,IK)+FF*AVST
      BEN(IJ,IK)=BEN(IJ,IK)+OBEN(IT+1,IK)
      IF(REL(IT,IJ,IK).LT.DEM(IT)) BEN(IJ,IK)=-1.E+20
      RETURN
      END


C*********************************************************************
      FUNCTION FINT(A,B,AVAL,NN)
C*********************************************************************
      DIMENSION A(1),B(1)
      IF(AVAL.LT.A(1)) THEN
       FINT=B(1)
       RETURN
      ENDIF
      IF(AVAL.GT.A(NN)) THEN
       FINT=B(NN)
       RETURN
      ENDIF
      DO 10 I=1,NN
      IF(AVAL.EQ.A(I)) THEN
       FINT=B(I)
       RETURN
      ENDIF
      IF(A(I-1).LT.AVAL.AND.A(I).GT.AVAL) THEN
       FINT=B(I-1)+(B(I)-B(I-1))/(A(I)-A(I-1))*(AVAL-A(I-1))
       RETURN
      ENDIF
10    CONTINUE
      END
```

APPENDIX - 2
SAMPLE INPUT

```
   Test problem for dynamic programming - reservoir operation
 20   24   11   1375332480   55506692   10    3
       178.3     27654620.0    185022304.0          0.0
       179.8     34568276.0    234361584.0     280000.0
       181.4     42701988.0    296035680.0     896000.0
       182.9     51649072.0    370044608.0    2408000.0
       184.4     62629580.0    450220960.0    4060000.0
       185.9     74830152.0    555066944.0    5880000.0
       187.5     87844088.0    678415104.0    8064000.0
       189.0    100044656.0    844935232.0   10444000.0
       190.5    113871968.0   1023790080.0   13356000.0
       192.0    126885904.0   1208812416.0   16380000.0
       193.5    138273104.0   1375332480.0   19600000.0
1275332480
  1120.00   3120.00   1420.00   9120.00   8400.00   3360.00   2800.00
  1800.00   1560.00   2960.00   6160.00   4840.00   1550.00   9120.00
  5120.00   3420.00   2430.00   1780.00   9800.00   6440.00
  1540.0    9822.0    2988.0    2600.0    7200.0    4250.0    2200.0
  6712.0    1380.0    5600.0    5104.0    2034.0    2000.0    1750.0
  4500.0    3200.0    6380.0    2660.0    3504.0    3550.0
```

APPENDIX - 3
SAMPLE OUTPUT

Test problem for dynamic programming - reservoir operation

*** I N P U T D A T A ***

All data are in MKS units

The number of time periods for analysis are : 20
The computational time interval is : 24.0 hours

Maximum storage capacity is : 1375332480.000 cum
Minimum storage capacity is : 55506692.000 cum

Elevation - Area - Capacity - Release Capacity Table

S N	Elevation (m)	Area (sqm)	Capacity (cum)	Rel Capacity (cumecs)
1	178.30	27654620.00	185022304.00	0.00
2	179.80	34568276.00	234361584.00	280000.00
3	181.40	42701988.00	296035680.00	896000.00
4	182.90	51649072.00	370044608.00	2408000.00
5	184.40	62629580.00	450220960.00	4060000.00
6	185.90	74830152.00	555066944.00	5880000.00
7	187.50	87844088.00	678415104.00	8064000.00
8	189.00	100044656.00	844935232.00	10444000.00
9	190.50	113871968.00	1023790080.00	13356000.00
10	192.00	126885904.00	1209812416.00	16380000.00
11	193.50	[illegible]	1375332480.00	19600000.00

**** Results of Calculations ****

Period	Init Storage (cum)	Inflow (cumecs)	Demand (cumecs)	Release (cumecs)	Final Storage (cum)
1	1275332480.000	1120.000	6440.000	6454.791	814406528.000
2	814406528.000	3120.000	2988.000	3120.000	814406528.000
3	814406528.000	1420.000	1540.000	2756.630	698921728.000
4	698921728.000	9120.000	9822.000	10647.576	566939136.000
5	566939136.000	8400.000	2988.000	3817.271	962886912.000
6	962886912.000	3360.000	2600.000	3360.000	962886912.000
7	962886912.000	2800.000	7200.000	7382.729	566939136.000
8	566939136.000	1800.000	4250.000	4855.152	302974016.000
9	302974016.000	1560.000	2200.000	3087.576	170991456.000
10	170991456.000	2960.000	6712.000	2960.000	170991456.000
11	170991456.000	6160.000	1380.000	1577.272	566939136.000
12	566939136.000	4840.000	5600.000	7895.152	302974016.000
13	302974016.000	1550.000	5104.000	3077.576	170991456.000
14	170991456.000	9120.000	2034.000	3009.696	698921728.000
15	698921728.000	5120.000	2000.000	2064.847	962886912.000
16	962886912.000	3420.000	1750.000	1892.424	1094869504.000
17	1094869504.000	2430.000	4500.000	5485.153	830904320.000
18	830904320.000	1780.000	3200.000	3307.576	698921728.000
19	698921728.000	9800.000	6380.000	6744.848	962886912.000
20	962886912.000	6440.000	2660.000	3384.849	1226851968.000