A DIFFERENT APPROACH TO COMPLEX WATER RESOURCE SYSTEM CONTROL BY THE USE OF INPUT FORECASTS

D.A. Harwood

April 1981
WP-81-45

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

PREFACE

Water resource systems have been an important part of resources and environment related research at IIASA since its inception. As demands for water increase relative to supply, the intensity and efficiency of water resources management must be developed further. This in turn requires an increase in the degree of detail and sophistication of the analysis including economic, social and environmental evaluation of water resources development alternatives aided by application of mathematical modeling techniques, to generate inputs for planning, design, and operational decisions.

This paper is part of the comparative studies on operational decisionmaking in the multiple reservoir water resources systems initiated in 1979 by the "Regional Water Management" Research Task of the Resources and Environment Area of IIASA.

The paper presents a method that can be used for the real-time control of complex water resource systems. The method is based on the rolling control effect of forecast-decision-control. If perfectly accurate input forecasts could be obtained then a system could be controlled in an optimal fashion with respect to certain criteria. Forecasts are never perfect and hence the system is operated by the decision calculated for one time period only. The system is then updated and the process repeated.

The control decision is made over the forecasting time horizon using an iterative dynamic programming algorithm. This algorithm has been used to alleviate the problem of dimensionality with the standard dynamic programming procedure and is such that even with complex systems computer storage requirement is very small. Hence the whole control method can be con-

tained on a mini computer. The method has been incorporated in a simulation of an idealized conjunctive use pumped-storage reservoir/aquifer system in England and the Upper Vistula multi-reservoir system in Poland.

The research presented in this paper has been carried out by the Author at the University of Birmingham, England, and at IIASA.

Janusz Kindler
Chairman
Resources & Environment Area

ACKNOWLEDEGEMENTS

I am grateful to Dr. J. Kindler of IIASA for making the Vistula investigation possible and for his help and advice concerning the system. My thanks also extend to Mr. J. Eloranta for computational advice. Funding for the original work was supplied by the Natural Environment Research Council, England, and for the short Vistula investigation by IIASA and the British Government through The Royal Society. Finally, I thank Irene for her help and Figures 1 and 2.

D.A.H.

CONTENTS

A DIFFERENT APPROACH TO COMPLEX WATER RESOURCE SYSTEM CONTROL BY THE USE OF INPUT FORECASTS

D.A. Harwood

1. INTRODUCTION

The irregularity of the supply of naturally-occurring surface water is evident both temporally and spatially. The transformation of these properties into new properties with desirable values is usually carried out by a system of storages that enable the water to be developed, utilized and controlled so that specific objectives are achieved. There are two aspects of this system: the design parameters, and the operational procedure. In recent years the complexity of storage systems has increased and now a system can contain many reservoirs serving many demands and/or uses. Also, there is a need for procedures that can optimally design and control such complex systems because costs have risen sharply in recent years. The method explained in this paper is one attempt to deal with the control problem of complex storage systems.

The method is a real-time adaptive control rule that is a computer program. The program is capable of being loaded on a minicomputer and it is run once in each time period for which control is required. The program requires forecasts of future system inputs as data and calculates an optimal (with respect to certain criteria) control rule over the forecasting time horizon. The system is then controlled using this rule over one time period after which the program is updated and new forecasts obtained.

The method has been programmed into a full simulation study of a multi-purpose objective system based on a pumped-storage reservoir, Rutland Water, used in conjunction with the Lincolnshire Limestone aquifer in East Anglia, England. Also, the method has been used in a brief investigation on a system of storage reservoirs supplying competing demands on the Vistula River system in Poland.

The methodology and particular dynamic programming algorithm used are explained in the next section. This is followed by a brief explanation of the systems studied and some results from these investigations.

2. THE CONTROL METHOD

2.1. Methodology

There are many different procedures used in practice to control water resource systems. They are usually static control rules that define a control based on the present system state and the time of year. Many different techniques have been used in order to calculate such rules and they usually attempt to maximize some objective in the long-term. Two common forms used in England are the control curve and the dynamic programming rule.

An example of the design and use of control curves is given in the literature (Walsh, 1971) for the conjunctive use of reservoirs and other sources. The points on the control curve for each time period in the year are calculated from the amount of water required in store at that time, so that, if water is supplied continuously at a certain rate--the 'cut-back' rate--then the reservoir would just fail at the end of the design drought. Hence, in each time period if the current system state exceeds the calculated curve state then the system can be overdrawn. Otherwise only the 'cut-back' rate can be supplied. The other form of control is usually in a set of tables, one for each time period throughout the year. Then, given the current state of the system, the control releases can be read from the suitable table. These tables are often produced using dynamic programming procedures and some practical examples of this type of control are given in the literature (Hall, Shephard et al., 1967 and Mawer and Thorn, 1974). Both of these control procedures and all other rules that are calculated from historic data are static control rules. They specify decisions

quantities irrespective of any other information that may be available--for example input forecasts--and they need to be recalculated each time any of the cost structures used in their calculation alters. Also, when using these types of control for conjunctive use systems the reservoir is often taken as the cheapest source of water. However, with the increased use of pumped-storage reservoirs this asumption is becoming invalid and with a complex pumping cost structure it is impossible to say that one source is always the cheapest source.

The form of control presented here moves away from the methodology of control described above. The original research, carried out at Birmingham University, England (Harwood, 1980), was born out of an idea that possibly the control rule need not be fixed. Indeed if some form of forecast could be made as to future inputs to the system, then perhaps this should be taken into account at the present time in controlling the system. Of course, the forecast would be inaccurate, but if the control was only followed for a short time in relation to the forecast and then the system was updated, it is possible that this would lead to a more flexible, and in some circumstances better, form of control than that at present used.

In this adaptive real-time method, the control of a system is based upon a computer management model. The model contains no control rule for the system but is itself the system's control. The computer model is run at certain time intervals, supplied with details of past system data and the present state of the system. From these data future inputs to the system are forecast. The management part of the model then calculates the cheapest control policy for the system over the forecasting time horizon using a dynamic programming algorithm. This is passed back to the operator as a control rule for the system. The system is controlled by this rule for one time period after which the whole sequence is repeated. Hence, the output from the running of the computer program is the control rule for the system.

2.2. Algorithm

Many problems which can be divided into a number of stages or subproblems, where the decision taken at one stage affects the later stages, can be solved by the dynmaic programming technique (Bellman, 1957). The stages are described by a stage variable which is given the values 0,1,2, ... and is usually time. Two other variables are also used in dynamic programming: the state variables describe the condition of the problem at any stage, and the decision variables are chosen to obtain an optimal solution of the problem at each stage.

Dynamic programming unlike any other optimization techniques, always gives a global optimal solution. However, the computational effort increases rapidly with the number of state variables since a k-dimensional search is required to find an optimal policy for each stage with k state variables. This problem of dimensionality presents a serious obstacle in solving large problems and usually k is restricted to three to four. Other methods have been put forward in an attempt to alleviate this problem. Notable amongst these is Larson's state increment dynamic programming (Larson, 1968). It is particularly useful for problems of high dimension since the computer storage requirement is always reduced when compared with the conventional procedure. The reduction is obtained by the introduction of two new concepts:

(i) The time interval over which the control is applied, δt, is not set equal to the time interval for which optimal control is required, Δt. Instead it is chosen so that the change in any state variable, x_i, is at most one increment Δx_i. Hence, the next state is always close to the present state, in fact:

$$x_i^t - \Delta x_i \leq x_i^{t+\delta t} \leq x_i^t + \Delta x_i$$

where x_i^t is the state of variable i at time t. Therefore, when finding a path through the network, only the minimum costs at the points of an N-dimensional hypercube need to be stored.

(ii) The computations are not carried out according to the stage ordering but in units which Larson calls blocks. In the conventional procedure all state relationships at one stage are investigated before looking at the next stage. State increment investigates the state/stage space on block at a time. These blocks include a few states over many stages. All state relationships within a block are investigated before moving to another block of equal size.

The reduction in storage requirement for this procedure is dependent on both of these modifications being used. If the block format is introduced but the control time is unrestricted then the path from one state might go outside the block at the next stage. If the control time restriction is used but not the block format then all minimum costs would still need to be stored. By combining these two concepts the storage requirement is reduced from one location for every state in the state/stage space to one location for every state in a block.

For the purpose of calculating the control policy in the method described in section 2.1., the inputs, which have been forecast over a finite time horizon, are assumed accurate. Also, the current state of the system is known so that the problem can be formulated as forward, deterministic programming. However, for use in real-time control this method would normally be programmed on a mini-computer with limited storage. For this reason the standard dynamic programming algorithm has not been used as this requires a computer storage location for every state at every stage. However, the state increment algorithm was felt to be unsatisfactory as this requires the same storage on some form of background storage. Only one block is used in the computer store at any one time but all the other values need to be stored so that an optimal policy may be found. It is not always possible to obtain background storage in this way with a mini-computer. Hence, a modified algorithm is used which requires a minimum of storage in the computer and no background storage.

The algorithm used is based upon the block notion of state increment dynamic programming. This concept allows the whole state/stage space to be analysed although only one block is analysed at any one time. Large savings in storage are possible if the technique

can be made iterative with only one block being analysed at each iteration. This is achieved by redefining a block. For state increment programming it is defined as a few states over many stages: in the algorithm used here it is defined as a few states over all stages. If an initial feasible policy is found, then a block is centered on this policy and only controls that keep the state within the block are investigated. The optimal policy within a block is found and this becomes a new initial policy, a new block is defined and the process continues in an iterative manner. In this way the only storage required at any one time is for all points within a block.

The major problem with this algorithm is to restrict the movement from stage to stage such that a new state is never outside the block. State increment programming achieves this by restricting the time interval over which control is applied so that the change in any one state variable is at most one increment. However, this allows movement to states on the block boundary and movement between blocks. For the algorithm used here the range of the control variables has been restricted such that the path always stays in the defined block. The restriction is achieved by only analysing control which is within one control increment of the initial policy control. The increment need to be given and its relationship with the state increment needs to be calculated a priori. This relationship is such that if the control variable altered by one increment at each stage then the block boundary would be reached by the last stage.

For example, consider a single state variable X and a single control varialbe U. For the initial feasible policy control vector $(u_1, u_2 \ldots u_T)$ there corresponds a state vector $(x_1, x_2 \ldots x_T)$. The relationship between the state increment, Δx, and the control increment, Δu, is such that:

$$x_1 + \sum_{t=1}^{T} (u_t + \Delta u) \leq x_T + \Delta x$$

Then, having decided a state increment size, the control increment size on a can be calculated from the equational form of this relationship. However, the algorithm contains two heuristic procedure to alter the relationship if problems occur:

(i) The block, at any stage t, is defined as

$$x_t - \Delta x \leq x_t^* < x_t + \Delta x$$

where x_t^* is a new policy being analysed at time t.

This space is split into three states so that the block consists of three states over every stage. The choice of three states here is arbitrary. The use of more states might be more accurate but would increase the computer storage requirement. The states are of equal size defined by:

State 1 $$x_t - \Delta x \leq x_t^* < x_t - \frac{\Delta x}{3}$$

State 2 $$x_t - \frac{\Delta x}{3} \leq x_t^* \leq x_t + \frac{\Delta x}{3}$$

State 3 $$x_t + \frac{\Delta x}{3} < x_t^* \leq x_t + \Delta x$$

The intial policy is defined by the control vector $(u_1, u_2 \ldots u_T)$ and the state vector $(2\ 2 \ldots 2)$. This is because the intial policy always lies in state 2 since:

$$x_t - \frac{\Delta x}{3} \leq x_t \leq x_t + \frac{\Delta x}{3}$$

The intial policy control is altered at each stage by the control increment in such a way that three control levels are analysed: $u_t - \Delta u$, u_t and $u_t + \Delta u$. Of every control is investigated at every stage and x_t^* never becomes states 1 or 3, then the state increment is too large in comparison with the control increment. In this case the state increment is reduced by a factor of 0.75 and the process continues.

(ii) If the path of a new policy moves rapidly to the block boundary then the control increment is too large in comparison with the state increment. In this case the state increment is increased by a factor of 1.25 and the process continues.

The iterative process of moving through the state space, investigating one block at a time, stops when the new policy found is the same as the initial policy. For the defined block size an

optimal policy has been found. The state and control increments are then reduced by a factor of 0.75 and the process continues. The process finally halts when the control increment is reduced below a threshold value. The best policy is now the overall optimal policy. Using this modified algorithm a discretization of the state space is possible that would be unusable with the standard procedure. For a system used to test the algorithm (Harwood, 1980, p.100) the modified algorith, required 108 storage locations. With the initial state increment size the standard procedure would have required 56088 locations and with the final increment 1804032 locations. With a general system of n state variables, discretized to M levels over T stages, the modified algorithm, requires $3^n \times T$ locations compared with $M^n \times T$.

3. CASE STUDIES

The testing of the methodology and algorithm described in section 2 has been carried out on two systems. Most of this work was an investigation into the operation of a pumped-storage used in conjunction with an aquifer. The work, carried out at Birmingham University, England and reported in detail elsewhere (Harwood, 1980), analysed the sensitivity of the method to certain conditions--for example errors in the forecasts and low flow situations--and compared the results with a standard method of operation of pumped-storage reservoirs used in England. A schematic representation of the simplified system, based on Rutland Water and the Lincolnshire Limestone aquifer in Eastern England, is shown in Figure 1. The second part of the work, carried out by the author at IIASA, was a brief investigation into the feasibility of using the method on a more complex multi-reservoir system and to demonstrate its use. A schematic representation of the simplified system, based on the Upper Vistula river system in Poland, is shown in Figure 2.

3.1. Rutland Water/Lincolnshire Limestone

The reservoir known as Rutland Water is the major component of a pumped-storage scheme operated by the Anglian Water Authority. It is situated in Eastern England, approximately 32 kms. East of Leicester. The reservoir is over 8 kms long, with a perimeter of 39 kms and a water surface area, when full, of 1260 hectars. It

has a volume of 124 million cubic metres and on commissioning was the largest man-made lake in Britain. A very small proportion of the reservoir intake comes from the catchment area upstream of the dam which is only 90 kms in area. The main reservoir intake comes from abstraction points on two rivers. The furthest river is 17.1 kms away from the reservoir and there is a total hydraulic lift from this river of 75.5 metres.

The primary objective of the scheme is water supply. However, the large potential for recreation and amenity at the reservoir is being developed. These dual purposes create a conflict because recreation dictates that water should be available in Summer when water for supply is at its scarcest. Hence, as well as the usual reliability constraint on the system, an amenity constraint must also be included.

The aquifer of the Lincolnshire Limestone is an area of the South Lincolnshire in Eastern England and that has been extensively developed for water supply. The limestone is generally between 20 and 30 meters thick and is confined in the West. Recharge occurs in the unconfined area (approximately 250 square kms) and is on average of the order of 31.5 million cubic meters per year. Investigations of the aquifer have taken place over many years and further details can be obtained elsewhere. (Downing and Williams, 1969).

For this work the system has been simplified (see Figure 1) so that the reservoir and aquifer are used conjunctively to supply a single demand. Certain points of interest about the simulation are:

(i) the aquifer is managed within the conjunctive use scheme by a simple rule whereby maximum abstraction within a certain time period is restricted;

(ii) the intricate cost structure of the pumping stations has been accurately modelled by including the actual three-tier electricity tariff used when estimating cost;

(iii) the reliability and amenity constraints on the reservoir have been imposed by the use of penalty functions.

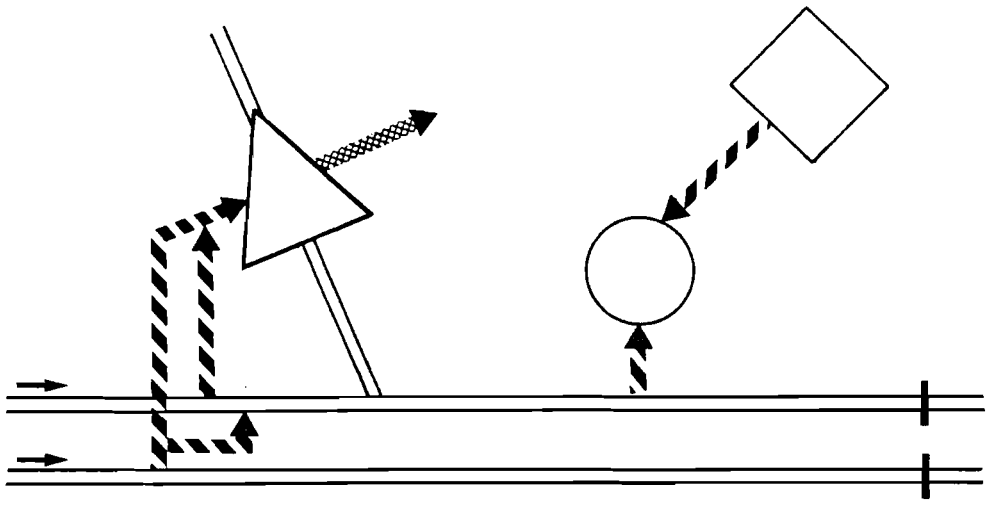

Figure 1. Rutland water.

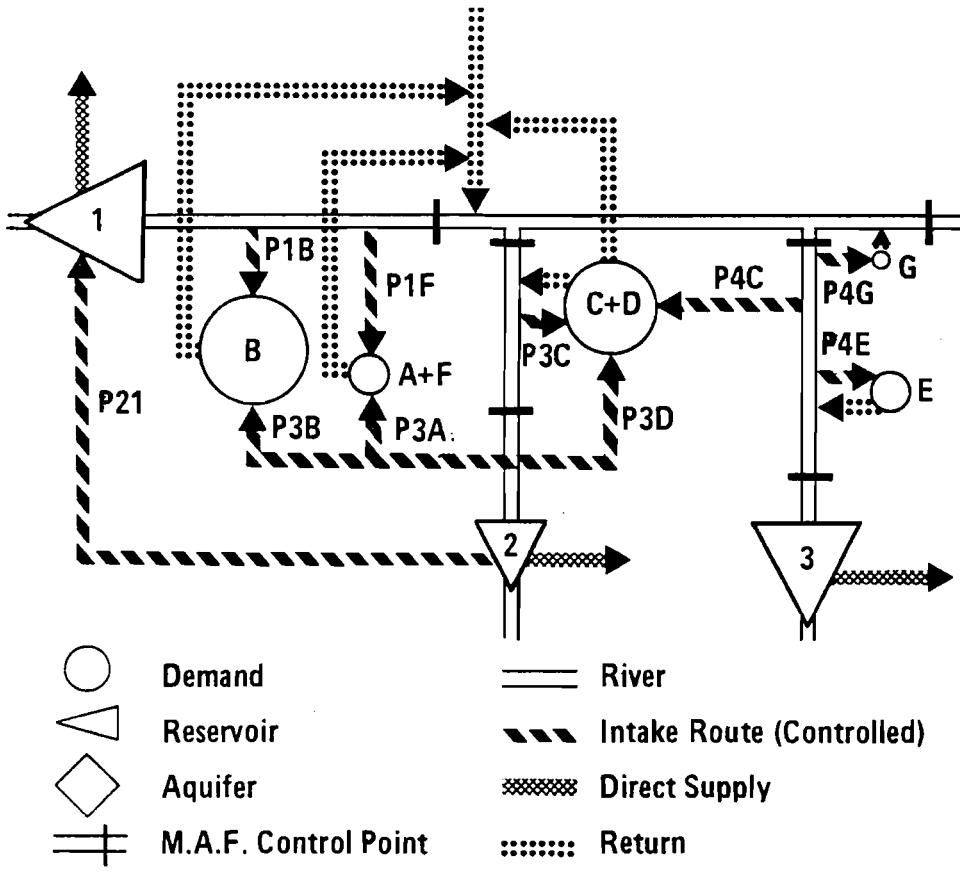

Figure 2. Vistula system.

In the simulation two methods of control have been included for comparison purposes: the adaptive real-time control, and the method of keeping the reservoir storage up to a certain level as long as possible. The latter method is a standard control rule for pumped-storage reservoirs as it maximizes the reliability of the system. However, with an electricity tariff which varies throughout the year this rule can prove to be very expensive especially after a series of low flows when the reservoir has been drawn down. An example of this is shown in Figure 3 where the cost of pumping a million gallons (approximately 4.5 Ml) is given for one of the pumping stations for each month of a simulation. The data used for this simulation were historic flows with months 21 to 24 replaced by low flows (that is, a four month drought). It can be seen from Figure 4 that after the drought the control rule refills the reservoir (up to 105 million cubic meters) as quickly as possible. This restricted maximum storage has been used for comparison because the reservoir storage rarely increases above this value using the adaptive control method. Allowing the pump whenever possible case to completely fill the reservoir would increase initial costs and hence could bias the conclusions. The costs of pumping per million gallons shown in Figure 3 are high because the reservoir is being continually 'topped-up' by the control method irrespective of the cost involved. These costs should be compared with Figure 5 which shows the costs associated with the adaptive control rule supplied with perfect foreknowledge of inputs. It can be seen that the large costs associated with pumping a small amount of water have been eliminated. In fact the total cost of the three year simulation has been reduced from £512 46[1] to £402 343. This saving has been effected by altering the pumping pattern. It can be seen from Figure 6 that the adaptive rule slowly increases the reservoir storage before, and slowly refills the reservoir after the drought. In this way overall costs are reduced.

In the investigation of the Ruland Water system many simulations were carried out using five different data sets, two demands and a number of forecasting techniques including: perfect foreknowledge, systematic errors applied to perfect foreknowledge, the use of averages as forecasts, Box-Jenkins' type forecasts (Box and Jenkins, 1976) and Kalman filter forecasts (Kalman, 1960). The adaptive

method worked well with all the data sets and there seemed to be little data dependency in the control. Many of the simulation runs, using the real-time control, produced solutions that were economically better than the strategy of keeping the reservoir as full as possible. Overall, the largest savings occurred when the system was being used the most, that is when there was a large demand on the system or at times of drought. The algorithm was programmable on a mini-computer although for this investigation all work was carried out on a CDC 7600. On this machine a weekly control rule was calculated in an average of three seconds. The benefits to be gained from the method definately depended on the accuracy of the forecasting technique used. However, the investigation showed that the method is worthy of consideration as a means of controlling a water resource system.

3.2. Upper Vistula System

The Vistula River is the largest river in Poland. It flows North to the Baltic Sea from its headwaters in the Carpathian Mountains. The total catchment area within Poland is 168 000 square kms. Of importance in this study is the reach from Goczalkowice Reservoir on the Small Vistula River to its confluence with the Skawa River. This reach and the whole Vistula basin are described in detail elsewhere (Laski and Kindler, 1976). A simplified system, equivalent to that used by Kindler (1977), based on the Upper Vistula system and shown in Figure 2 has been used.

The primary objective of the scheme is water supply for the large industrial and municipal users in the region. Also, flows must be maintained at six points in the system as the problem of water pollution is severe, and the reservoir must provide flood control as the flood hazard is high. This latter problem has not been taken into account in this work.

Work on the operation of multi-reservoir systems has been one area of study at IIASA over the past few years and the Upper Vistula system has been one of the case studies (Kindler et al., 1979). It was decided that the feasibility of using the method described in section 2 on this complex system would be tested as part of this work. A complete simulation of the simplified system has been programmed using FORTRAN IV, see Appendix 1, and this program has

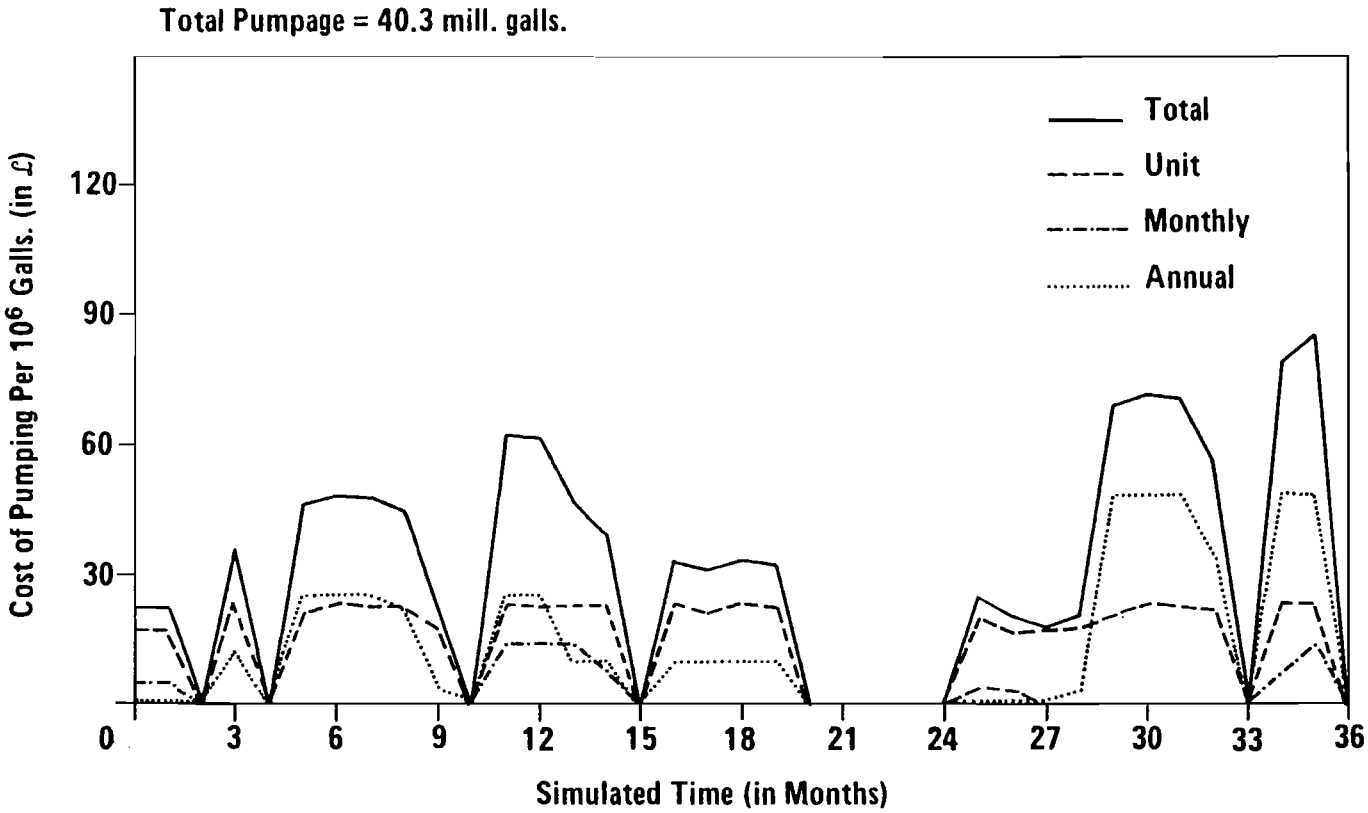

Figure 3. Pumping costs (standard rule).

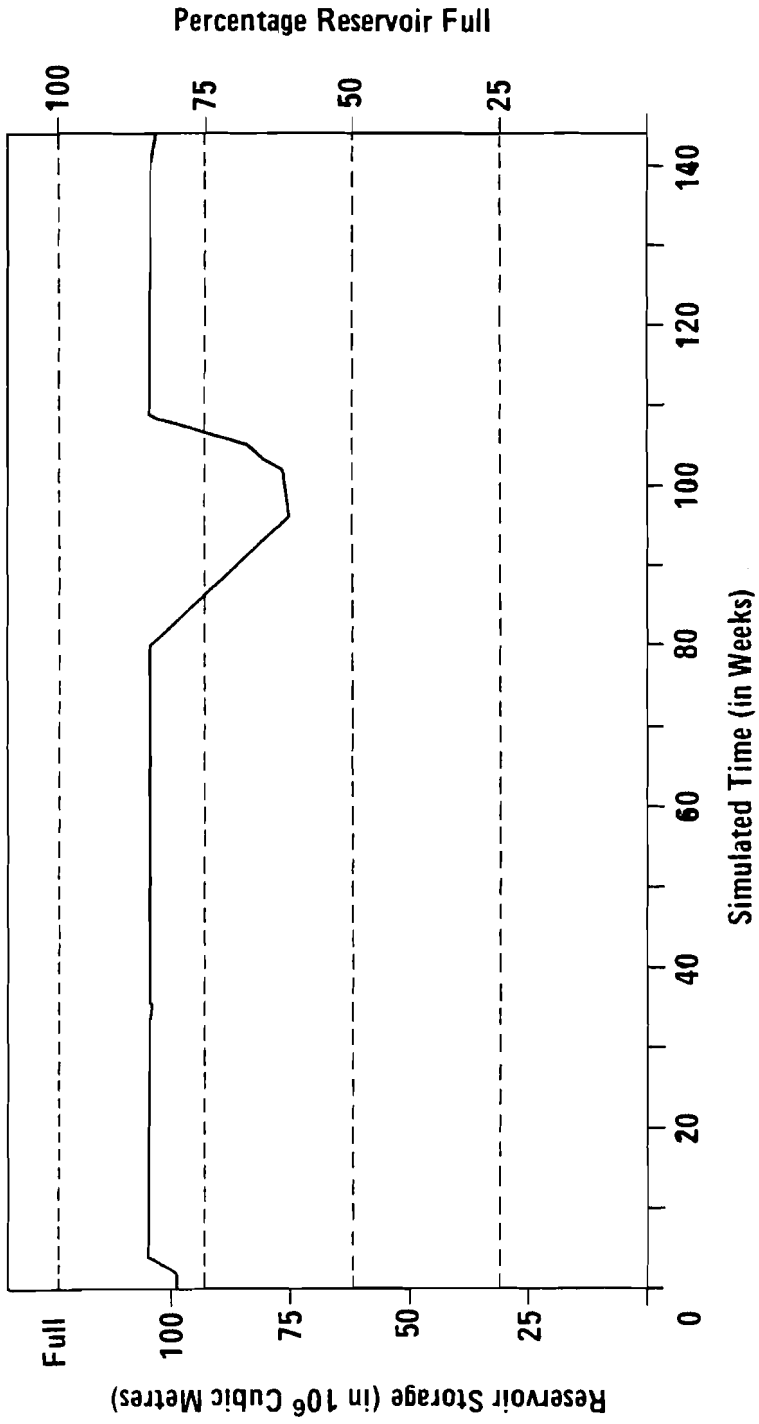

Figure 4. Reservoir storage (standard rule).

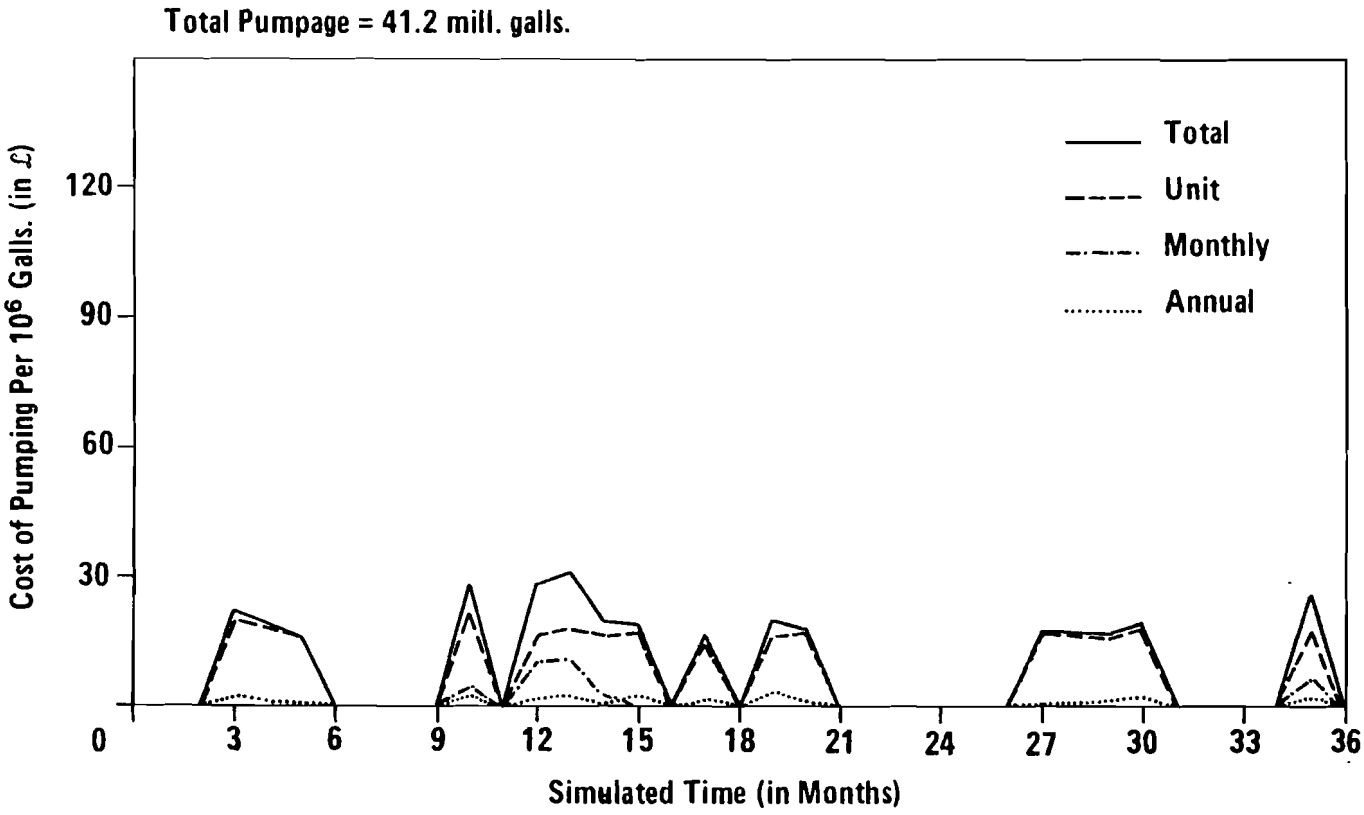

Figure 5. Pumping costs (adaptive rule).

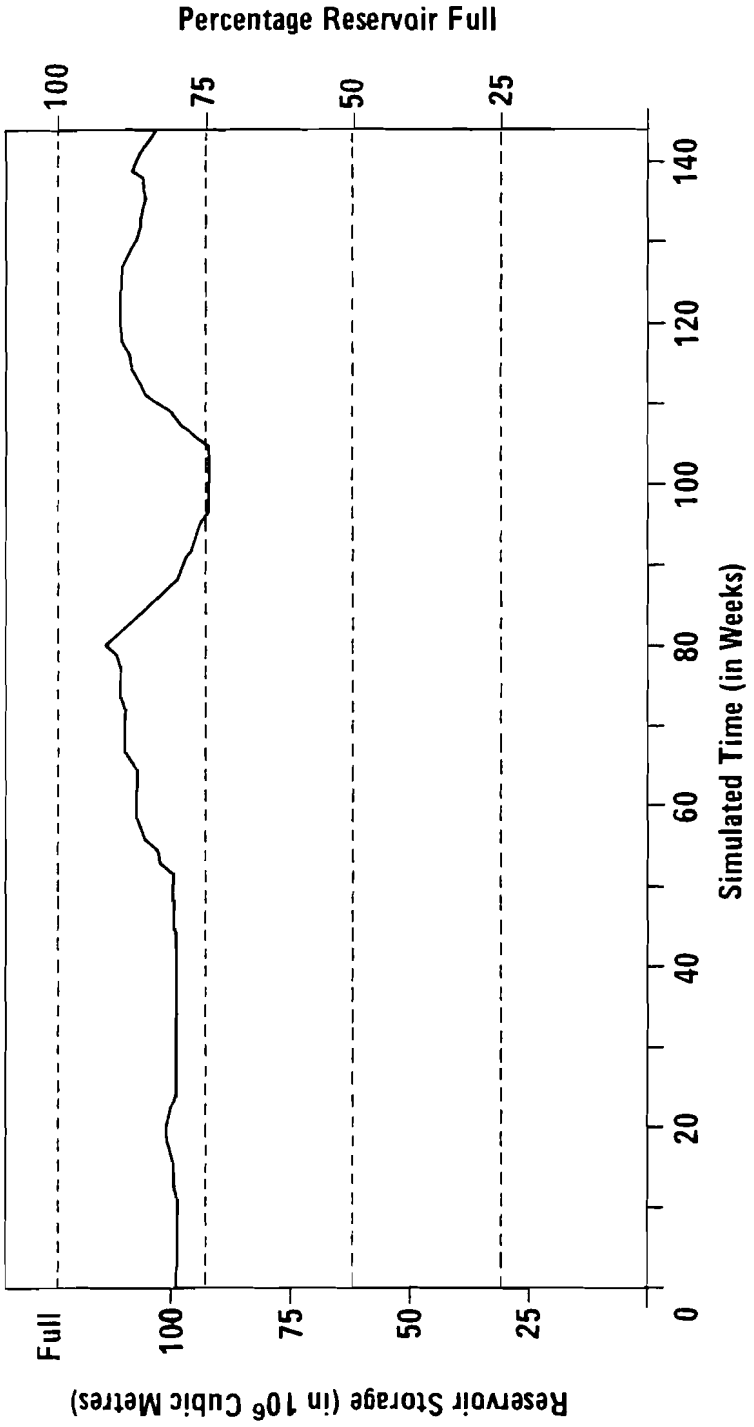

Figure 6. Reservoir storage (adaptive rule).

been run on a mini-computer (a PDP 11/70 running under the UNIX operating system). However, although the use of the iterative algorithm allows the method to be loaded on a mini-computer the complexity of the control variables means that run-time increases. It is evident at present that the calculation of a monthly control rule for the reservoirs is feasible but the running of a complete simulation of many months takes too long to the PDP 11/70. Appendix 2 shows the iterative steps the method goes through when calculating the control required at the next time step in order to move from the initial policy to an 'optimum' policy. It should be noted that to reach the point shown required more than ten hours computer time on a PDP 11/70 and fourteen minutes computer time on a CDC 7600.

It can be seen from Appendix 2 that the forecasting time horizon used for the method was six time steps. Reference should be made to Figure 2 in order to understand the results. The initial policy was calculated as one solution from the family of feasible solutions. Taking the forecasts to be correct the storages and releases from the three reservoirs are shown over the forecasting time horizon. The demands at centres A, B, ..., F and G and the flows into the three reservoirs are given. The initial policy is shown as the quantity of water, in cumecs to be pumped through the ten pipelines and the total cost of such a policy. The control increment used in this work was 0.25 cumecs and it can be seen that after one iteration a new policy has been found which decreases the overall cost and alters some of the pumpings by this increment. The method continued finding cheaper policies until a minimum cost policy was found. The results shown in Appendix 2 do not extend to this minimum cost policy. The different policies found are highly dependent on the penalties introduced for drawing down the reservoir and not meeting demand. It can be seen in the final iteration that certain demands are not met and hence different penalties would need to be used in order to overcome this deficit. Work is continuing on testing the simulation of the system.

A number of points of interest have emerged from demonstrating the use of the adaptive method on a more complex system:

(i) as the complexity of the system analysed increases it is the run-time of the program and not the com-

puter storage requirement that is the limiting factor in its applicability;

(ii) minimizing a penalty for not meeting demand is not enough on its own for an objective function. The feasible initial policy required by the algorithm has, for the data tested, always met demand and hence the penalty is zero. Realistic pumping costs are required so that the method can calculate the cheapest policy, expecially when more than one abstraction satisfies one demand. Also, reservoir penalty functions are required to inhibit reservoir drawdown;

(iii) the relationship between the costs and penalties in (ii) needs careful analysis as this relationship is important to the final policy chosen;

(iv) the relationship of the three reservoir state increments relative to each other is very important. It is necessary to find a relationship whereby feasible policy routes spread evenly throughout the state space. The heuristic procedures incorporated in the algorithm alter all increments by the same amount. Hence, the relationship needs analysis beforehand so that it can be set correctly. It may be necessary to incorporate in the algorithm a procedure to alter the relationship but this has not been attempted here.

Further study into the possibility of the use of the method on the Upper Vistula system would need to analyse these points carefully. However, work is continuing to investigate certain aspects of the method's application to this system.

REFERENCES

Bellman, R.E. (1957) Dynamic Programming (Princeton, U.S.A.: Princeton University Press), 340 pp.

Box, G.E.P., and G.M. Jenkins (1976) Time Series Analysis: Forecasting and Control (Revised edition; Holden-Day series in time series analysis; San Francisco, U.S.A.: Holden-Day Inc.), 575 pp.

Downing, R.A., and B.P.J. Williams (1969) *The ground-water hydrology of the Lincolnshire Limestone*, Water Resource Board Publication No. 9, Reading, England, 160 pp.

Hall, W.A., Shephard, R.W., and W.S. Butcher, et al., (1967) *Optimum Operations for Planning of a Complex Water Resource System*, Contribution No. 122, Water Resource Center, University of California, U.S.A., 75 pp.

Harwood, D.A. (1980) *The Real-time Control of Water Resource Systems*, Unpublished Ph.D. Thesis, University of Birmingham, England, 302 pp.

Kalman, R.E. (1960) *A New Approach to Linear Filtering and Prediction Problems*, Transactions of the American Society of Mechanical Engineers, Series D - Journal of Basic Engineering, Vol. 83, pp. 95-108.

Kindler, J. (1977) *The Monte Carlo Approach to Optimization of the Operation Rules for a System of Storage Reservoirs*, Hydrological Sciences Bulletin, Vol. 12, No. 1, pp. 203-214.

Kindler, J., Salewicz, K.A., Slota, H. and T. Terlikowski (1979) *Operation of multiple Reservoir Systems: A case Study of the Upper Vistula System (An Introduction)*, Collaborative Paper No. CP-79-17, IIASA, Laxenburg, Austria, 35 pp.

Larson, R.E. (1968) State Increment Dynamic Programming (Modern, analytic and computational methods in science and mathematics science; New York, U.S.A.: American Elsevier Publishing Company Inc.), 256 pp.

Laski, A. and J. Kindler (1976) *The Vistula River Project*, in Szöllösi-Nagy, A. (ed.) (1976) Workshop on the Vistula and Tisza River Basins, 11-13 February 1975, Collaborative Paper No. CP.76-5, IIASA, Laxenburg, Austria, 136 pp.

Mawer, P.A. and D. Thorn (1974) *Improved Dynamic Programming Procedures and their Practical Application to Water Resource Systems*, Water Resources Research, Vol. 10, No. 2, pp. 183-190.

Walsh, P.D. (1971) *Designing Control Rules for the Conjunctive Use of Impounding Reservoirs*, Journal of the Institution of Water Engineers and Scientists, Vol. 25, No. 7, pp. 371-380.

Appendix 1

Vistula Simulation Computer Program

```
      DIMENSION D(7), C(26), P(10), F(3), DS(3,12), QI(3,42), RM(3),
     *AF(6), R(3)
      REAL MAF(6,12)
      LOGICAL EQUAL, ZERO
      COMMON /BLOCK1/ F9
      COMMON /BLOCK2/ MAF, RM, RES1, RES2, RES3, L9, DS, D, DLS1, DLS2,
     *DLS3, DLT, QI
      COMMON /BLOCK3/ AF, P, C, R
      EQUAL(A,B) = ABS(A-B).LT.0.00001
      ZERO(A) = ABS(A).LT.0.00001
C
C          MAIN PROGRAM SIMULATES BEHAVIOUR OF UPPER VISTULA SYSTEM.
C
C
C          READ IN DATA :
C
C            L = 0, 1 OR 2 DEPENDING ON THE CONTROL RULE TIME STEP
C               REQUIRED. 0 = DAY, 1 = WEEK AND 2 = MONTH.
C            L9 = FORECASTING TIME HORIZON.
C            IT = NUMBER OF SIMULATION TIME STEPS.
C            MAF(I,J) = MINIMUM ACCEPTABLE FLOW AT POINT I IN MONTH J.
C            DS(I,J) = DIRECT SUPPLY REQUIRED FROM RESERVOIR I IN
C                      MONTH J.
C            D(I) = DEMAND I.
C            RM(I) = MAXIMUM STORAGE FOR RESERVOIR I.
C            RESN = STARTING STORAGE FOR RESERVOIR N.
C            DLSN = STATE INCREMENT FOR RESERVOIR N.
C            DLT = CONTROL INCREMENT.
C            QI(I,J) = FLOW IN RIVER I IN TIME PERIOD J.
C
      READ (1,9999) L
 9999 FORMAT (I5)
      F9 = 86400.0
      IF (L.EQ.1) F9 = 604800.0
      IF (L.EQ.2) F9 = 2419200.0
      READ (1,9999) L9
      READ (1,9999) IT
      DO 1 I=1,6
       READ (1,9998) (MAF(I,J),J=1,12)
 9998  FORMAT (12F6.2)
    1 CONTINUE
      DO 2 J=1,3
       READ (1,9998) (DS(J,I),I=1,12)
    2 CONTINUE
      READ (1,9998) (D(I),I=1,7)
      READ (1,9997) (RM(I),I=1,3)
 9997 FORMAT (3F13.2)
      READ (1,9997) RES1, RES2, RES3
      READ (1,9997) DLS1, DLS2, DLS3
      READ (1,9998) DLT
```

```
      DO 3 I=1,42
       READ (1,9996) (QI(J,I),J=1,3)
 9996  FORMAT (3F7.2)
    3 CONTINUE
C
C          VARIABLES OF THE FORM APMN AND PMN (E.G. AP4E) AND ALWAYS
C         PMNI (E.G. P4GI(J)) RELATE TO THE QUANTITY OF WATER PUMPED
C         FROM M TO N.  THE CODES ARE :
C
C            3A = FROM RIVER 3 TO DEMAND A
C            1B = FROM RIVER 1 TO DEMAND B
C            3B = FROM RIVER 3 TO DEMAND B
C            3C = FROM RIVER 3 TO DEMAND C
C            4C = FROM RIVER 4 TO DEMAND C
C            3D = FROM RIVER 3 TO DEMAND D
C            4E = FROM RIVER 4 TO DEMAND E
C            1F = FROM RIVER 1 TO DEMAND F
C            4G = FROM RIVER 4 TO DEMAND G
C            21 = FROM RESERVOIR 2 TO RESERVOIR 1
C
C          DEMAND A = D(1), ... , DEMAND G = D(7).
C
C          DEMAND A AND F RELATE TO THE SAME DEMAND CENTRE AS DO C AND
C         D.  THIS IS BECAUSE ALTHOUGH THEY RELATE TO THE SAME DEMAND
C       . THEY HAVE DIFFERENT PENALTIES ATTACHED.
C
      AP1F = 0.0
      AP3A = 0.0
      AP1B = 0.0
      AP3B = 0.0
      AP3C = 0.0
      AP4C = 0.0
      AP3D = 0.0
      AP4E = 0.0
      AP21 = 0.0
      AP4G = 0.0
      WRITE (4,9995)
 9995 FORMAT (20X, 31HRESERVOIR STORAGES AND RELEASES/8X, 2HS1, 10X,
     *2HR1, 10X, 2HS2, 10X, 2HR2, 10X, 2HS3, 10X, 2HR3//)
      WRITE (5,9994)
 9994 FORMAT (24X, 28HFLOWS AND MAFS AT SIX POINTS//)
      WRITE (3,9993)
 9993 FORMAT (23X, 30HTEN CONTROL PUMPINGS AND COSTS/15H   P21    P1B   ,
     *62H  P3B    P1F    P3A    P4G    P3D    P3C    P4C    P4E    COST/
     */)
      DO 6 I=1,IT
C
C          CALL MANAGEMENT SUBROUTINE.
C
       CALL VSTPOL(I, AP1F, AP3A, AP1B, AP3B, AP3C, AP4C, AP3D, AP4E,
```

```
     * AP21, AP4G)
       DO 4 J=1,3
        F(J) = QI(J,I)
     4 CONTINUE
       M9 = MOD(I,12)
       IF (M9.EQ.0) M9 = 12
C
C          MAKE DECISION ON PUMPING USING CONTROL RULE CALCULATED.
C
       A = ((MM(1)-RES1)/F9) + DS(1,M9)
       IF (A.GT.(RES2/F9)-DS(2,M9)) A = (RES2/F9) - DS(2,M9)
       IF (A.GT.AP21) A = AP21
       IF (A.LT.0.0) A = 0.0
       P(9) = A
       A = RES1/F9
       IF (A.GT.AP1B) A = AP1B
       IF (A.LT.0.0) A = 0.0
       P(3) = A
       A = (RES2/F9) - P(9)
       IF (A.GT.AP3B) A = AP3B
       IF (A+P(3).GT.D(2)) A = D(2) - P(3)
       IF (A.LT.0.0) A = 0.0
       P(4) = A
       A = (RES2/F9) - P(9) - P(4)
       IF (A.GT.AP3A) A = AP3A
       IF (A.LT.0.0) A = 0.0
       P(2) = A
       A = (RES2/F9) - P(9) - P(4) - P(2)
       IF (A.GT.AP3D) A = AP3D
       IF (A.LT.0.0) A = 0.0
       P(7) = A
       A = (RES2/F9) - P(9) - P(4) - P(2) - P(7)
       IF (A.GT.AP3C) A = AP3C
       IF (A.LT.0.0) A = 0.0
       P(5) = A
       A = (RES1/F9) - P(3)
       IF (A.GT.AP1F) A = AP1F
       IF (A.LT.0.0) A = 0.0
       P(1) = A
       A = RES3/F9
       IF (A.GT.AP4C) A = AP4C
       IF (A+P(5).GT.D(3)) A = D(3) - P(5)
       IF (A.LT.0.0) A = 0.0
       P(6) = A
       A = (RES3/F9) - P(6)
       IF (A.GT.AP4E) A = AP4E
       IF (A.LT.0.0) A = 0.0
       P(8) = A
       A = (RES3/F9) - P(6) - P(8)
       IF (A.GT.AP4G) A = P4G
```

```
      IF (A.LT.0.0) A = 0.0
      P(10) = A
      R(1) = P(1) + P(3) + MAF(1,M9)
      IF (R(1).GT.RES1/F9) R(1) = RES1/F9
      AF(1) = R(1) - P(1) - P(3)
      R(2) = P(2) + P(4) + P(7) + MAF(2,M9)
      IF (R(2).LT.P(2)+P(4)+P(7)+P(5)+MAF(3,M9)-0.16) R(2) = P(2) +
     * P(4) + P(7) + P(5) + MAF(3,M9) - 0.16
      IF (P(2).GT.(RES2/F9)-P(9)) R(2) = (RES2/F9) - P(9)
      AF(2) = R(2) - R(2) - P(4) - P(7)
      AF(3) = AF(2) + 0.16 - P(5)
      R(3) = MAF(4,M9)
      IF (R(3).LT.MAF(5,M9)+P(6)+P(8)+P(10)-0.25) R(3) = MAF(5,M9) +
     * P(6) + P(8) + P(10) - 0.25
      IF (R(3).GT.RES3/F9) R(3) = RES3/F9
      AF(4) = R(3)
      AF(5) = AF(4) + 0.25 - P(6) - P(8) - P(10)
      AF(6) = AF(1) + AF(3) + AF(5) + P(10) + 16.08
      RES1 = RES1 + ((P(9)+F(1)-R(1)-DS(1,M9))*F9)
      RES2 = RES2 + ((F(2)-P(9)-R(2)-DS(2,M9))*F9)
      RES3 = RES3 + ((F(3)-R(3)-DS(3,M9))*F9)
      CST = 0.0
      DO 5 J=1,10
       CALL COST(J, P(J), C(J))
       CST = CST + C(J)
    5 CONTINUE
      WRITE (4,9992) RES1, R(1), RES2, R(2), RES3, R(3)
 9992 FORMAT (1X, 3(F16.2, F8.3))
      WRITE (5,9991) (AF(J),MAF(J,M9),J=1,6)
 9991 FORMAT (1X, 12F6.2)
      WRITE (3,9990) (P(J),J=1,10), CST
 9990 FORMAT (1X, 10F7.3, F7.2)
    6 CONTINUE
      STOP 0
      END
```

```
      SUBROUTINE VSTPOL(N9, AP1F, AP3A, AP1B, AP3B, AP3C, AP4C, AP3D,
     *AP4E, AP21, AP4G)
      DIMENSION TC(27,7), P1FI(6), P3AI(6), P1BI(6), R(3), RM(3),
     *TCI(7), P3BI(6), P3CI(6), P4CI(6), P3DI(6), P4EI(6), P21I(6),
     *P4GI(6), QI(3,42), Q(3,6), D(7), C(26), SH(27,3,7), DEC(27,13,7),
     *P(10), AF(6), DS(3,12), S(3), RI(6,3)
      REAL MAF(6,12)
      INTEGER T, S1(27,7), S2(27,7), S3(27,7)
      LOGICAL EQUAL, ZERO
      COMMON /BLOCK1/ F9
      COMMON /BLOCK2/ MAF, RM, RES1, RES2, RES3, L9, DS, D, DLS1, DLS2,
     *DLS3, DLT, QI
      COMMON /BLOCK3/ AF, P, C, R
      EQUAL(A,B) = ABS(A-B).LT.0.00001
      ZERO(A) = ABS(A).LT.0.00001
      K9(I,J,K) = ((K-1)*9) + ((J-1)*3) + I
C
C          SUBROUTINE CALCULATES CONTROL RULE.
C
      DO 2 J=1,3
       DO 1 I=1,6
        Q(J,I) = QI(J,N9+I-1)
    1  CONTINUE
    2 CONTINUE
      WRITE (2,9999)
 9999 FORMAT (1X/////32H RESERVOIR STORAGES AND RELEASES/)
C
C          INITIALIZE ARRAYS.
C
      DO 4 I=1,27
        DO 45 J=1,L9+1
        TC(I,J) = -1.0
    3  CONTINUE
    4 CONTINUE
      TC(14,1) = 0.0
      TCI(1) = 0.0
C
C          CALCULATE INITIAL POLICY.
C
      S(1) = RES1
      S(2) = RES2
      S(3) = RES3
      DO 3 T=1,L9
       N8 = N9 + T
       M9 = MOD(N8,12)
       IF (M9.EQ.0) M9 = 12
       P21I(T) = 0.0
       A = S(2)/F9
       IF (A.GT.D(2)) A = D(2)
       IF (A.GT.5.5) A = 5.5
```

```
      AF(3) = AF(2) - P3CI(T) + 0.16
      AF(4) = R(3)
      AF(5) = R(3) - P4CI(T) - P4EI(T) - P4GI(T) + 0.25
      AF(6) = AF(1) + AF(3) + AF(5) + P4GI(T) + 16.08
      DO 6 I=1,3
       SH(14,I,T+1) = S(I)
       RI(T,I) = R(I)
    7 CONTINUE
      WRITE (2,9998) (S(I),R(I),I=1,3)
 9998 FORMAT (1X, 3(F15.3, F7.3))
      DO 7 I=1,6
       J = I + 10
       CALL MPEN(MAF(I,M9), AF(I), C(J))
    8 CONTINUE
      DO 8 I=1,3
       J = I + 16
       CALL SPEN(RM(I), S(I), C(J))
    9 CONTINUE
      P(1) = P1FI(T)
      P(2) = P3AI(T)
      P(3) = P1BI(T)
      P(4) = P3BI(T)
      P(5) = P3CI(T)
      P(6) = P4CI(T)
      P(7) = P3DI(T)
      P(8) = P4EI(T)
      P(9) = P21I(T)
      P(10) = P4GI(T)
      DO 9 I=1,10
       CALL COST(I, P(I), C(I))
   10 CONTINUE
      P(1) = D(1) - P(2)
      P(2) = D(2) - P(3) - P(4)
      P(3) = D(3) - P(5) - P(6)
      P(4) = D(4) - P(7)
      P(5) = D(5) - P(8)
      P(6) = D(6) - P1FI(T)
      P(7) = D(7) - P(10)
      DO 10 I=1,7
       J = 19 + I
       CALL DPEN(I, P(I), C(J))
   11 CONTINUE
      TCI(T+1) = 0.0
      DO 11 I=1,26
       TCI(T+1) = TCI(T+1) + C(I)
   12 CONTINUE
      TCI(T+1) = TCI(T+1) + TCI(T)
      S1(14,T+1) = 2
      S2(14,T+1) = 2
      S3(14,T+1) = 2
```

```
   13 CONTINUE
C
C          WRITE OUT INITIAL POLICY.
C
      WRITE (2,9997)
 9997 FORMAT(1X///22X'DEMANDS',34X,'OUTFLOWS'/)
      DO 12 I=1,L9
       WRITE (2,9996) (D(J),J=1,7), (Q(K,I),K=1,3)
 9996  FORMAT (1X, 7F7.3, 5X, 3F7.3)
   14 CONTINUE
      WRITE (2,9995)
 9995 FORMAT (1X///14HINITIAL POLICY//30H   P21    P1B    P3B    P1F    P3A,
     *39H    P3D    P3C    P4C    P4E    P4G      COST/)
      DO 14 T=1,L9
       TEMP = TCI(T+1) - TCI(T)
       WRITE (2,9994) P21I(T), P1BI(T), P3BI(T), P1FI(T), P3AI(T),
     * P3DI(T), P3CI(T), P4CI(T), P4EI(T), P4GI(T), TEMP
 9994  FORMAT (1X, 10F6.3, F12.3)
   15 CONTINUE
      WRITE (2,9993) TCI(L9+1)
 9993 FORMAT (1X///27H COST OF INITIAL POLICY IS , F12.3)
      TOTC = TCI(L9+1)
   16 SH(14,1,1) = RES1
      SH(14,2,1) = RES2
      SH(14,3,1) = RES3
      MPNTR1 = 0
C
C          TAKE INITIAL POLICY AND INVESTIGATE ALL CONTROLS +/- DLT
C         FROM THE INITIAL CONTROL THAT KEEPS PATH WITHIN BLOCK.
C
   17 T = 1
   18 N8 = T + N9
      M9 = MOD(N8,12)
      IF (M9.EQ.0) M9 = 12
      I1 = 1
   19 I2 = 1
   20 I3 = 1
   21 IF (ZERO(TC(K9(I1,I2,I3),T))) GO TO 22
      IF (TC(K9(I1,I2,I3),T).LT.0.0) GO TO 42
   22 DO 15 I4=1,3
       A1 = SH(K9(I1,I2,I3),1,T)
       A2 = SH(K9(I1,I2,I3),2,T)
       IF (ZERO(P21I(T)) .AND. I4.EQ.1) GO TO 41
       IF (EQUAL(P21I(T),10.0) .AND. I4.EQ.3) GO TO 41
       IF (EQUAL(P21I(T),A2/F9) .AND. I4.EQ.3) GO TO 41
       IF (EQUAL(P21I(T),((RM(1)-A1)/F9)-Q(1,T)) .AND. I4.EQ.3) GO TO 41
       P21 = P21I(T) + (I4-2)*DLT
       IF (P21.GT.10.0) P21 = 10.0
       IF (P21.GT.A2/F9) P21 = A2/F9
       IF (P21.GT.((RM(1)-A1)/F9)-Q(1,T)) P21 = ((RM(1)-A1)/F9) - Q(1,T)
```

```
      IF (P21.LT.0.0) P21 = 0.0
      DO 41 I5=1,3
       A1 = SH(K9(I1,I2,I3),1,T)
       IF (ZERO(P1BI(T)) .AND. I5.EQ.1) GO TO 40
       IF (EQUAL(P1BI(T),10.0) .AND. I5.EQ.3) GO TO 40
       IF (EQUAL(P1BI(T),A1/F9) .AND. I5.EQ.3) GO TO 40
       P1B = P1BI(T) + (I5-2)*DLT
       IF (P1B.GT.10.0) P1B = 10.0
       IF (P1B.GT.A1/F9) P1B = A1/F9
       IF (P1B.GT.D(2)) P1B = D(2)
       IF (P1B.LT.0.0) P1B = 0.0
       DO 40 I6=1,3
        A2 = SH(K9(I1,I2,I3),2,T)
        IF (ZERO(P3BI(T)) .AND. I6.EQ.1) GO TO 39
        IF (EQUAL(P3BI(T),5.5) .AND. I6.EQ.3) GO TO 39
        P3B = P3BI(T) + (I6-2)*DLT
        IF (P3B.GT.5.5) P3B = 5.5
        IF (P3B.GT.(A2/F9)-P21) P3B = (A2/F9) - P21
        IF (P3B.GT.D(2)-P1B) P3B = D(2) - P1B
        IF (P3B.LT.0.0) P3B = 0.0
        DO 39 I7=1,3
         A2 = SH(K9(I1,I2,I3),2,T)
         IF (ZERO(P3AI(T)) .AND. I7.EQ.1) GO TO 38
         IF (EQUAL(P3AI(T),D(1)) .AND. I7.EQ.3) GO TO 38
         P3A = P3AI(T) + (I7-2)*DLT
         IF (P3A.GT.(A2/F9)-P21-P3B) P3A = (A2/F9) - P21 - P3B
         IF (P3A.GT.D(1)) P3A = D(1)
         IF (P3A.LT.0.0) P3A = 0.0
         DO 38 I8=1,3
          A2 = SH(K9(I1,I2,I3),2,T)
          IF (ZERO(P3DI(T)) .AND. I8.EQ.1) GO TO 37
          IF (EQUAL(P3DI(T),D(4)) .AND. I8.EQ.3) GO TO 37
          P3D = P3DI(T) + (I8-2)*DLT
          IF (P3D.GT.(A2/F9)-P21-P3B-P3A) P3D = (A2/F9) - P21 - P3B -
     *    P3A
          IF (P3D.GT.D(4)) P3D = D(4)
          IF (P3D.LT.0.0) P3D = 0.0
          DO 37 I9=1,3
           A2 = SH(K9(I1,I2,I3),2,T)
           IF (ZERO(P3CI(T)) .AND. I9.EQ.1) GO TO 36
           IF (EQUAL(P3CI(T),D(3)) .AND. I9.EQ.3) GO TO 36
           P3C = P3CI(T) + (I9-2)*DLT
           IF (P3C.GT.(A2/F9)-P21-P3B-P3A-P3D) P3C = (A2/F9) - P21 -
     *     P3B - P3A - P3D
           IF (P3C.GT.D(3)) P3C = D(3)
           IF (P3C.LT.0.0) P3C = 0.0
           DO 36 I11=1,3
            A1 = SH(K9(I1,I2,I3),1,T)
            A2 = SH(K9(I1,I2,I3),2,T)
            IF (ZERO(P1FI(T)) .AND. I11.EQ.1) GO TO 35
```

```
      IF (EQUAL(P1FI(T),D(6)) .AND. I11.EQ.3) GO TO 33
      P1F = P1FI(T) + (I11-2)*DLT
      IF (P1F.GT.(A1/F9)-P1B) P1F = (A1/F9) - P1B
      IF (P1F.GT.D(6)) P1F = D(6)
      IF (P1F.LT.0.0) P1F = 0.0
      R(1) = P1F + P1B + MAF(1,M9)
      IF (R(1).GT.A1/F9) R(1) = A1/F9
      S(1) = SH(K9(I1,I2,I3),1,T) + ((Q(1,T)+P21-R(1)-DS(1,M9))*
     *F9)
      IF (EQUAL(S(1),RM(1))) GO TO 23
      IF (S(1).LT.RM(1)) GO TO 23
      R(1) = R(1) + ((S(1)-RM(1))/F9)
      S(1) = RM(1)
      IF (ZERO(S(1)) .OR. S(1).GT.0.0) GO TO 23
      R(1) = R(1) + S(1)
      S(1) = 0.0
23    IF (S(1).GT.SH(K9(I1,I2,I3),1,T+1)+DLS1) GO TO 35
      IF (S(1).LT.SH(K9(I1,I2,I3),1,T+1)-DLS1) GO TO 35
      AF(1) = R(1) - P1B - P1F
      R(2) = P3A + P3B + P3D + MAF(2,M9)
      IF (R(2).LT.P3A+P3B+P3C+P3D+MAF(3,M9)-0.16) R(2) = P3A +
     *P3B + P3C + P3D + MAF(3,M9) - 0.16
      IF (R(2).GT.(A2/F9)-P21) R(2) = (A2/F9) - P21
      S(2) = SH(K9(I1,I2,I3),2,T) + ((Q(2,T)-P21-R(2)-DS(2,M9))*
     *F9)
      IF (EQUAL(S(2),RM(2))) GO TO 24
      IF (S(2).LT.RM(2)) GO TO 24
      R(2) = R(2) + ((S(2)-RM(2))/F9)
      S(2) = RM(2)
      IF (ZERO(S(2)) .OR. S(2).GT.0.0) GO TO 24
      R(2) = R(2) + S(2)
      S(2) = 0.0
24    IF (S(2).GT.SH(K9(I1,I2,I3),2,T+1)+DLS2) GO TO 35
      IF (S(2).LT.SH(K9(I1,I2,I3),2,T+1)-DLS2) GO TO 35
      AF(2) = R(2) - P3A - P3B - P3D
      AF(3) = AF(2) - P3C + 0.16
      DO 35 I14=1,3
       A3 = SH(K9(I1,I2,I3),3,T)
       IF (ZERO(P4CI(T)) .AND. I14.EQ.1) GO TO 34
       IF (EQUAL(P4CI(T),A3/F9) .AND. I14.EQ.3) GO TO 34
       P4C = P4CI(T) + (I14-2)*DLT
       IF (P4C.GT.A3/F9) P4C = A3/F9
       IF (P4C.GT.D(3)-P3C) P4C = D(3) - P3C
       IF (P4C.LT.0.0) P4C = 0.0
       DO 34 I15=1,3
        A3 = SH(K9(I1,I2,I3),3,T)
        IF (ZERO(P4EI(T)) .AND. I15.EQ.1) GO TO 33
        IF (EQUAL(P4EI(T),D(5)) .AND. I15.EQ.3) GO TO 33
        P4E = P4EI(T) + (I15-2)*DLT
        IF (P4E.GT.D(5)) P4E = D(5)
```

```
      IF (P4E.GT.(A3/F9)-P4C) P4E = (A3/F9) - P4C
      IF (P4E.LT.0.0) P4E = 0.0
      DO 33 I16=1,3
       A3 = SH(K9(I1,I2,I3),3,T)
       IF (ZERO(P4GI(T)) .AND. I16.EQ.1) GO TO 32
       IF (EQUAL(P4GI(T),D(7)) .AND. I16.EQ.3) GO TO 32
       P4G = P4GI(T) + (I16-2)*DLT
       IF (P4G.GT.D(7)) P4G = D(7)
       IF (P4G.GT.(A3/F9)-P4E-P4C) P4G = (A3/F9) - P4E - P4C
       IF (P4G.LT.0.0) P4G = 0.0
       R(3) = MAF(4,M9)
       IF (MAF(4,M9).LT.MAF(5,M9)+P4C+P4E+P4G-0.25) R(3) = P4C
     *  + P4E + P4G + MAF(5,M9) - 0.25
       IF (R(3).GT.A3/F9) R(3) = A3/F9
       S(3) = SH(K9(I1,I2,I3),3,T) + ((Q(3,T)-R(3)-DS(3,49))*
     *  F9)
       IF (EQUAL(S(3),RM(3))) GO TO 25
       IF (S(3).LT.RM(3)) GO TO 25
       R(3) = R(3) + ((S(3)-RM(3))/F9)
       S(3) = RM(3)
       IF (ZERO(S(3)) .OR. S(3).GT.0.0) GO TO 25
       R(3) = R(3) + S(3)
       S(3) = 0.0
25     IF (S(3).GT.SH(K9(I1,I2,I3),3,T+1)+DLS3) GO TO 32
       IF (S(3).LT.SH(K9(I1,I2,I3),3,T+1)-DLS3) GO TO 32
       AF(4) = R(4)
       AF(5) = AF(4) - P4C - P4E - P4G + 0.25
       AF(6) = AF(1) + AF(3) + AF(5) + P4G + 16.08
C
C      CALCULATE COST OF ANY FEASIBLE PATHS.
C
       P(1) = P1F
       P(2) = P3A
       P(3) = P1B
       P(4) = P3B
       P(5) = P3C
       P(6) = P4C
       P(7) = P3D
       P(8) = P4E
       P(9) = P21
       P(10) = P4G
       DO 32 I=1,10
        CALL COST(I, P(I), C(I))
26     CONTINUE
       DO 26 I=1,6
        J = I + 10
        CALL MPEN(MAF(I,M9), AF(I), C(J))
27     CONTINUE
       DO 27 I=1,3
        J = I + 16
```

```
                CALL SPEN(RM(I), S(I), C(J))
   28           CONTINUE
                P(1) = D(1) - P(2)
                P(2) = D(2) - P(3) - P(4)
                P(3) = D(3) - P(5) - P(6)
                P(4) = D(4) - P(7)
                P(5) = D(5) - P(8)
                P(6) = D(6) - P1F
                P(7) = D(7) - P(10)
                DO 28 I=1,7
                 J = I + 19
                 CALL DPEN(I, P(I), C(J))
   29           CONTINUE
                A = S(1) - SH(K9(I1,I2,I3),1,T+1)
                IZ1 = 2
                IF (A.GT.DLS1/3.0) IZ1 = 3
                IF (A.LT.(0.0-DLS1)/3.0) IZ1 = 1
                A = S(2) - SH(K9(I1,I2,I3),2,T+1)
                IZ2 = 2
                IF (A.GT.DLS2/3.0) IZ2 = 3
                IF (A.LT.(0.0-DLS2)/3.0) IZ2 = 1
                A = S(3) - SH(K9(I1,I2,I3),3,T+1)
                IZ3 = 2
                IF (A.GT.DLS3/3.0) IZ3 = 3
                IF (A.LT.(0.0-DLS3)/3.0) IZ3 = 1
                TMCST = 0.0
                DO 29 I=1,26
                 TMCST = TMCST + C(I)
   30           CONTINUE
                TMCST = TMCST + TC(K9(I1,I2,I3),T)
C
C         IF NEW POLICY CHEAPER THAN ANY OTHER TO THE SAME STATE OR
C       NO OTHER ROUTE TO THE SAME STATE EXISTS THEN RECORD ROUTE.
C
                I91 = K9(IZ1,IZ2,IZ3)
                IF (ZERO(TC(I91,T+1))) GO TO 32
                IF (TC(I91,T+1).LT.0.0) GO TO 31
                IF (EQUAL(TC(I91,T+1),TMCST)) GO TO 32
                IF (TMCST.GT.TC(I91,T+1)) GO TO 32
   31           TC(I91,T+1) = TMCST
                S1(I91,T+1) = I1
                S2(I91,T+1) = I2
                S3(I91,T+1) = I3
                DEC(I91,1,T+1) = P21
                DEC(I91,2,T+1) = P1B
                DEC(I91,3,T+1) = P3B
                DEC(I91,4,T+1) = P3A
                DEC(I91,5,T+1) = P3D
                DEC(I91,6,T+1) = P3C
                DEC(I91,7,T+1) = P4G
```

```
                DEC(I91,8,T+1) = P1E
                DEC(I91,9,T+1) = P4C
                DEC(I91,10,T+1) = P4E
                DEC(I91,11,T+1) = R(1)
                DEC(I91,12,T+1) = R(2)
                DEC(I91,13,T+1) = R(3)
                SH(I91,1,T+1) = S(1)
                SH(I91,2,T+1) = S(2)
                SH(I91,3,T+1) = S(3)
   32          CONTINUE
   33         CONTINUE
   34        CONTINUE
   35       CONTINUE
   36      CONTINUE
   37     CONTINUE
   38    CONTINUE
   39   CONTINUE
   40  CONTINUE
   41 CONTINUE
   42 I3 = I3 + 1
      IF (I3.LE.3) GO TO 21
      I2 = I2 + 1
      IF (I2.LE.3) GO TO 20
      I1 = I1 + 1
      IF (I1.LE.3) GO TO 19
      WRITE (6,9992) T, (TC(I1,T+1),I1=1,27)
 9992 FORMAT (1X, 5H TIME, I3, 10H COSTS ARE/(1X, 6F12.3))
      T = T + 1
      IF (T.LE.L9) GO TO 18
C
C          FIND CHEAPEST POLICY.
C
      DO 30 I1=1,3
       DO 46 I2=1,3
        DO 45 I3=1,3
         I91 = K9(I1,I2,I3)
         IF (I1.EQ.2 .AND. I2.EQ.2 .AND. I3.EQ.2) GO TO 44
         IF (ZERO(TC(I91,L9+1))) GO TO 43
         IF (TC(I91,L9+1).LT.0.0) GO TO 44
   43    TEMP = TC(I91,L9+1)
         IZ1 = I1
         IZ2 = I2
         IZ3 = I3
         GO TO 54
   44    CONTINUE
   45   CONTINUE
   46 CONTINUE
      IF (TC(14,L9+1).LT.0.0) GO TO 48
      IF (EQUAL(TC(14,L9+1),TOTC)) GO TO 68
      IF (TC(14,L9+1).GT.TOTC) GO TO 47
```

```
      IZ1 = 2
      IZ2 = 2
      IZ3 = 2
      TEMP = TC(14,L9+1)
      GO TO 59
   47 DLS1 = DLS1*0.75
      DLS2 = DLS2*0.75
      DLS3 = DLS3*0.75
      WRITE (6,9991) DLS1, DLS2, DLS3
 9991 FORMAT (1X, 19H 1 DLS DECREASED TO, 3F13.3)
      GO TO 49
   48 DLS1 = DLS1*1.25
      DLS2 = DLS2*1.25
      DLS3 = DLS3*1.25
      WRITE (6,9990) DLS1, DLS2, DLS3
 9990 FORMAT (1X, 17H DLS INCREASED TO, 3F13.3)
   49  DO 395 T=1,L9+1
       DO 44 I1=1,3
        DO 53 I2=1,3
         DO 52 I3=1,3
          TC(K9(I1,I2,I3),T) = -1.0
   50     CONTINUE
   51    CONTINUE
   52   CONTINUE
   53 CONTINUE
      TC(14,1) = 0.0
      GO TO 17
   54 DO 51 I1=1,3
       DO 50 I2=1,3
        DO 58 I3=1,3
         I91 = K9(I1,I2,I3)
         IF (ZERO(TC(I91,L9+1))) GO TO 55
         IF (TC(I91,L9+1).LT.0.0) GO TO 56
         IF (EQUAL(TEMP,TC(I91,L9+1))) GO TO 56
         IF (TEMP.LT.TC(I91,L9+1)) GO TO 56
   55    IZ1 = I1
         IZ2 = I2
         IZ3 = I3
         TEMP = TC(I91,L9+1)
   56    CONTINUE
   57   CONTINUE
   58 CONTINUE
      IF (EQUAL(TEMP,TOTC) .AND. IZ1.EQ.2 .AND. IZ2.EQ.2 .AND.
     *IZ3.EQ.2) GO TO 69
      IF (EQUAL(TEMP,TOTC) .AND. MPNTR1.EQ.2) GO TO 69
      IF (EQUAL(TEMP,TOTC)) MPNTR1 = MPNTR1 + 1
      IF (TEMP.GT.TOTC) GO TO 47
      GO TO 60
   59 DLS1 = DLS1*0.75
      DLS2 = DLS2*0.75
```

```
      DLS3 = DLS3*0.75
      WRITE (6,9989) DLS1, DLS2, DLS3
 9989 FORMAT (1X, 19H 2 DLS DECREASED TO, 3F13.3)
   60 I91 = K9(IZ1,IZ2,IZ3)
C
C          IF NEW POLICY CHEAPER THAN INITIAL POLICY THEN IT BECOMES
C         THE INITIAL POLICY AND THE PROCESS CONTINUES.
C
      DO 57 I=1,L9
       J = L9 - I + 1
       TC(14,J+1) = TC(I91,J+1)
       P21I(J) = DEC(I91,1,J+1)
       P1BI(J) = DEC(I91,2,J+1)
       P3BI(J) = DEC(I91,3,J+1)
       P3AI(J) = DEC(I91,4,J+1)
       P3DI(J) = DEC(I91,5,J+1)
       P3CI(J) = DEC(I91,6,J+1)
       P4GI(J) = DEC(I91,7,J+1)
       P1FI(J) = DEC(I91,8,J+1)
       P4CI(J) = DEC(I91,9,J+1)
       P4EI(J) = DEC(I91,10,J+1)
       SH(14,1,J+1) = SH(I91,1,J+1)
       SH(14,2,J+1) = SH(I91,2,J+1)
       SH(14,3,J+1) = SH(I91,3,J+1)
       RI(J,1) = DEC(I91,11,J+1)
       RI(J,2) = DEC(I91,12,J+1)
       RI(J,3) = DEC(I91,13,J+1)
       I8 = I91
       I91 = K9(S1(I8,J+1),S2(I8,J+1),S3(I8,J+1))
   61 CONTINUE
      WRITE (6,9988) TOTC, TEMP
 9988 FORMAT (1X, 7H OLD C=, F12.3, 7H NEW C=, F12.3)
      WRITE (2,9999)
      DO 56 I=1,L9
       WRITE (2,9998) (SH(14,J,I+1),RI(I,J),J=1,3)
   62 CONTINUE
      WRITE (2,9987)
 9987 FORMAT (1X///11H NEW POLICY//33H   P21   P1B   P3B   P1F   P3A
     *36HP3D   P3C   P4C   P4E   P4G      COST/)
      DO 61 I=1,L9
       TMP = TC(14,I+1) - TC(14,I)
       WRITE (2,9994) P21I(I), P1BI(I), P3BI(I), P1FI(I), P3AI(I),
     * P3DI(I), P3CI(I), P4CI(I), P4EI(I), P4GI(I), TMP
   63 CONTINUE
      WRITE (2,9986) TC(14,L9+1)
 9986 FORMAT (1X///22H COST OF NEW POLICY IS, F12.3)
      TOTC = TEMP
       DO 425 T=1,L9+1
       TCI(T) = TC(14,T)
       DO 62 I1=1,3
```

```
          DO 63 I2=1,3
           DO 67 I3=1,3
            TC(K9(I1,I2,I3),T) = -1.0
   64      CONTINUE
   65     CONTINUE
   66    CONTINUE
   67   CONTINUE
        TC(14,1) = 0.0
        GO TO 16
   68   DLS1 = DLS1*0.75
        DLS2 = DLS2*0.75
        DLS3 = DLS3*0.75
        WRITE (6,9985) DLS1, DLS2, DLS3
 9985   FORMAT (1X, 19H 3 DLS DECREASED TO, 3F13.3)
C
C          IF THE NEW POLICY IS THE SAME AS THE INITIAL POLICY THEN
C         REDUCE ALL THE DELTAS I.E. DLT, DLS1, DLS2 AND DLS3.
C
   69   IF (DLT.LT.0.05) GO TO 70
        DLT = DLT*0.75
        DLS1 = DLS1*0.75
        DLS2 = DLS2*0.75
        DLS3 = DLS3*0.75
        WRITE (6,9984) DLT
 9984   FORMAT (1X, 15H DLT REDUCED TO, F7.3)
        GO TO 16
   70   AP1F = P1FI(1)
        AP3A = P3AI(1)
        AP1B = P1BI(1)
        AP3B = P3BI(1)
        AP3C = P3CI(1)
        AP4C = P4CI(1)
        AP3D = P3DI(1)
        AP4E = P4EI(1)
        AP21 = P21I(1)
        AP4G = P4GI(1)
        RETURN
        END
```

```
      SUBROUTINE MPEN(A, Q, C)
      LOGICAL EQUAL
      EQUAL(A,B) = ABS(A-B).LT.0.00001
C
C          SUBROUTINE CALCULATES THE PENALTY ASSOCIATED WITH NOT
C         CONTROLLING THE M.A.F.'S IN THE RIVERS.
C
      IF (EQUAL(A,Q)) GO TO 1
      IF (Q.GT.A) GO TO 1
      C = (A-Q)*10.0
      RETURN
    1 C = 0.0
      RETURN
      END
```

```
      SUBROUTINE SPEN(R, S, C)
      COMMON /BLOCK1/ F9
C
C          SUBROUTINE CALCULATES THE PENALTY ASSOCIATED WITH THE
C        RESERVOIR STORAGE.
C
      C = ((R-S)/F9)*400.0
      RETURN
      END
```

```
      SUBROUTINE DPEN(I, D, C)
C
C          SUBROUTINE CALCULATES THE PENALTY ASSOCIATED WITH NOT
C        MEETING DEMAND.
C
      T = 1500.0
      IF (I.EQ.1 .OR. I.EQ.3) T = 1000.0
      IF (I.EQ.5) T = 500.0
      IF (I.EQ.2) T = 5000.0
      C = D*T
      RETURN
      END
```

```
      SUBROUTINE COST(I, P, C)
      COMMON /BLOCK1/ F9
C
C          SUBROUTINE CALCULATES THE COST OF PUMPING.
C
      GO TO (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), I
    1 T = 0.06
      GO TO 11
    2 T = 0.06
      GO TO 11
    3 T = 0.06
      GO TO 11
    4 T = 0.10
      GO TO 11
    5 T = 0.0
      GO TO 11
    6 T = 0.10
      GO TO 11
    7 T = 0.06
      GO TO 11
    8 T = 0.0
      GO TO 11
    9 T = 0.10
      GO TO 11
   10 T = 0.06
   11 C = (((P*F9)/4.5)*T)/100.0
      C = C*0.75
      RETURN
      END
```

Appendix 2

Vistula Simulation Specimen Results

RESERVOIR STORAGES AND RELEASES

141184512.000	9.800	64799744.000	14.540	126354816.000	11.800
125508096.000	9.800	87443456.000	14.540	140966784.000	11.800
109154304.000	9.800	95596160.000	14.540	137386368.000	11.800
130443264.000	9.800	126841600.000	23.664	209478528.000	11.800
120331008.000	9.800	126841600.000	31.690	272958336.000	11.800
98969472.000	9.800	126841600.000	24.120	273611520.000	11.970

DEMANDS							FLOWS		
2.500	13.900	5.900	0.750	6.000	0.900	1.000	3.340	15.500	8.060
2.500	13.900	5.900	0.750	6.000	0.900	1.000	4.480	24.280	18.400
2.500	13.900	5.900	0.750	6.000	0.900	1.000	4.220	18.310	10.900
2.500	13.900	5.900	0.750	6.000	0.900	1.000	19.800	37.000	42.200
2.500	13.900	5.900	0.750	6.000	0.900	1.000	6.920	32.160	38.700
2.500	13.900	5.900	0.750	6.000	0.900	1.000	2.130	24.500	12.800

INITIAL POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	8.400	5.500	0.900	2.500	0.750	2.950	2.950	6.000	1.000	70747.29
0.000	8.400	5.500	0.900	2.500	0.750	2.950	2.950	6.000	1.000	57179.31
0.000	8.400	5.500	0.900	2.500	0.750	2.950	2.950	6.000	1.000	69127.29
0.000	8.400	5.500	0.900	2.500	0.750	2.950	2.950	6.000	1.000	43521.06
0.000	8.400	5.500	0.900	2.500	0.750	2.950	2.950	6.000	1.000	39697.06
0.000	8.400	5.500	0.900	2.500	0.750	2.950	2.950	6.000	1.000	43121.06

COST OF INITIAL POLICY IS 338393.094

RESERVOIR STORAGES AND RELEASES

141154312.000	9.800	65404544.000	14.290	127564416.000	11.300
124903296.000	10.050	88048256.000	14.540	142781184.000	11.550
108549504.000	9.800	95596160.000	14.790	139805568.000	11.550
129838464.000	9.800	126841600.000	23.664	212502528.000	11.550
119726208.000	9.800	126841600.000	31.690	273611520.000	12.780
98364672.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	8.400	5.500	0.900	2.250	0.750	2.950	2.700	5.750	1.000	71119.21
0.000	8.650	5.250	0.900	2.500	0.750	3.200	2.700	6.000	1.000	66951.39
0.000	8.400	5.500	0.900	2.500	0.750	3.200	2.700	6.000	1.000	63939.70
0.000	8.400	5.500	0.900	2.500	0.750	3.200	2.700	6.000	1.000	43154.76
0.000	8.400	5.500	0.900	2.500	0.750	3.200	2.700	6.000	1.000	39722.76
0.000	8.400	5.500	0.900	2.500	0.750	3.200	2.700	6.000	1.000	43254.75

COST OF NEW POLICY IS 338142.594

RESERVOIR STORAGES AND RELEASES

140579712.000	10.050	66614144.000	13.790	128774016.000	10.800
124903296.000	9.800	88048256.000	15.040	145200384.000	11.050
108549504.000	9.800	94991360.000	15.040	142829568.000	11.300
129838464.000	9.800	126841600.000	23.414	216131328.000	11.300
120331008.000	9.800	126841600.000	31.440	273611520.000	14.280
98969472.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	8.650	5.250	0.900	2.000	0.750	2.950	2.450	5.500	1.000	71232.61
0.000	8.400	5.500	0.900	2.500	0.750	3.450	2.450	5.750	1.000	66605.91
0.000	8.400	5.500	0.900	2.500	0.750	3.450	2.450	6.000	1.000	63433.90
0.000	8.400	5.500	0.900	2.500	0.750	3.450	2.450	6.000	1.000	47451.45
0.250	8.400	5.500	0.900	2.500	0.750	3.450	2.450	6.000	1.000	39620.26
0.000	8.400	5.500	0.900	2.500	0.750	3.450	2.450	6.000	1.000	43051.46

COST OF NEW POLICY IS 336395.625

RESERVOIR STORAGES AND RELEASES

139974912.000	10.300	67823744.000	13.290	129378816.000	10.550
124903296.000	9.800	88048256.000	15.290	146409984.000	10.800
109759104.000	9.550	93781760.000	15.290	145853568.000	10.550
131048064.000	9.800	126841600.000	22.914	219760128.000	11.000
120935808.000	9.800	126841600.000	31.690	273611520.000	15.780
99574272.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	8.900	5.000	0.900	1.750	0.750	2.950	2.450	5.250	1.000	71301.82
0.250	8.400	5.500	0.900	2.500	0.750	3.700	2.200	5.750	1.000	66400.91
0.250	8.400	5.500	0.650	2.500	0.750	3.700	2.200	5.750	0.750	68675.43
0.000	8.400	5.500	0.900	2.500	0.750	3.700	2.200	6.000	1.000	46548.15
0.000	8.400	5.500	0.900	2.500	0.750	3.700	2.200	6.000	1.000	39316.17
0.000	8.400	5.500	0.900	2.500	0.750	3.700	2.200	6.000	1.000	42843.15

COST OF NEW POLICY IS 335090.656

RESERVOIR STORAGES AND RELEASES

139370112.000	10.550	69033344.000	12.790	129983616.000	10.300
124298496.000	9.800	88653056.000	15.540	147619584.000	10.550
109154304.000	9.800	94386560.000	15.040	145248768.000	11.300
131048064.000	9.800	126841600.000	22.914	219760128.000	10.800
120935808.000	9.800	126841600.000	31.690	273611520.000	15.780
100179072.000	9.800	126841600.000	23.870	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	9.150	4.750	0.900	1.500	0.750	2.950	2.450	5.000	1.000	71371.02
0.250	8.400	5.500	0.900	2.500	0.750	3.950	1.950	5.750	1.000	66095.10
0.500	8.400	5.500	0.900	2.500	0.750	3.450	2.450	6.000	1.000	68235.50
0.250	8.400	5.500	0.900	2.500	0.750	3.950	1.950	6.000	1.000	46545.65
0.000	8.400	5.500	0.900	2.500	0.750	3.950	1.950	6.000	1.000	39212.84
0.250	8.400	5.500	0.900	2.500	0.750	3.950	1.950	6.000	1.000	42745.65

COST OF NEW POLICY IS 334205.781

RESERVOIR STORAGES AND RELEASES

133765312.000	10.800	70242944.000	12.290	130583416.000	10.000
122484096.000	10.050	90467456.000	15.540	143829184.000	10.300
107339904.000	9.800	95596160.000	15.290	147063168.000	11.050
129838464.000	9.800	126841600.000	23.164	222784128.000	10.300
119726208.000	9.800	126841600.000	31.690	273611520.000	17.030
98364672.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	9.600	4.500	0.900	1.250	0.750	2.950	2.450	4.750	1.000	71440.21
0.000	8.650	5.250	0.900	2.500	0.750	4.200	1.700	5.750	1.000	65543.18
0.500	8.400	5.500	0.900	2.500	0.750	3.700	2.200	6.000	1.000	67929.70
0.500	8.400	5.500	0.900	2.500	0.750	4.200	1.700	5.750	1.000	46365.65
0.000	8.400	5.500	0.900	2.500	0.750	4.200	1.700	6.000	1.000	39309.54
0.000	8.400	5.500	0.900	2.500	0.750	4.200	1.700	6.000	1.000	42841.56

COST OF NEW POLICY IS 333534.875

RESERVOIR STORAGES AND RELEASES

138160912.000	11.050	71452544.000	11.790	131193216.000	9.800
121274496.000	10.300	92281856.000	15.290	150038784.000	10.050
105525504.000	9.800	97410560.000	15.540	148877568.000	10.800
128628864.000	9.800	126841600.000	23.664	224598528.000	10.300
118516608.000	9.800	126841600.000	31.690	273611520.000	17.780
97155072.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	9.650	4.250	0.900	1.000	0.750	2.950	2.450	4.500	1.000	71509.42
0.000	8.900	5.000	0.900	2.250	0.750	4.450	1.450	5.750	1.000	68391.57
0.250	8.400	5.500	0.900	2.500	0.750	3.950	1.950	6.000	1.000	67423.10
0.750	8.400	5.500	0.900	2.500	0.750	4.450	1.450	6.000	1.000	46140.65
0.000	8.400	5.500	0.900	2.500	0.750	4.450	1.450	6.000	1.000	39406.26
0.000	8.400	5.500	0.900	2.500	0.750	4.450	1.450	6.000	1.000	42938.25

COST OF NEW POLICY IS 332809.281

RESERVOIR STORAGES AND RELEASES

137550712.000	11.300	72662144.000	11.290	131798016.000	9.590
120669696.000	10.300	92281856.000	15.790	152457984.000	9.300
105525504.000	9.800	96200960.000	15.790	151901568.000	10.300
123628864.000	9.800	126841600.000	23.164	228832128.000	9.800
119121408.000	9.800	126841600.000	31.440	273611520.000	19.530
97759872.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	9.900	4.000	0.900	0.750	0.750	2.950	2.450	4.250	1.000	71578.61
0.000	8.900	5.000	0.900	2.500	0.750	4.700	1.200	5.500	0.750	65228.27
0.500	8.400	5.500	0.900	2.500	0.750	4.200	1.700	6.000	1.000	67118.10
0.750	8.400	5.500	0.900	2.500	0.750	4.700	1.200	5.750	1.000	45459.85
0.250	8.400	5.500	0.900	2.500	0.750	4.700	1.200	6.000	1.000	39303.76
0.000	8.400	5.500	0.900	2.500	0.750	4.700	1.200	6.000	1.000	42734.96

COST OF NEW POLICY IS 331423.594

RESERVOIR STORAGES AND RELEASES

137313792.000	11.400	73508864.000	10.940	132402816.000	9.300
121032576.000	10.050	91918976.000	16.290	153062784.000	9.300
105283384.000	9.800	95838080.000	16.040	153111168.000	10.300
127782144.000	9.800	126841600.000	23.264	230041728.000	9.800
118879488.000	9.800	126841600.000	31.190	273611520.000	20.030
97517952.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	10.000	3.900	0.900	0.500	0.750	2.950	2.450	4.000	1.000	71672.01
0.000	8.650	5.250	0.900	2.500	0.750	4.950	0.950	5.500	1.000	64730.78
0.250	8.400	5.500	0.900	2.500	0.750	4.450	1.450	6.000	1.000	66811.50
0.500	8.400	5.500	0.900	2.500	0.750	4.950	0.950	6.000	1.000	45073.26
0.500	8.400	5.500	0.900	2.500	0.750	4.950	0.950	6.000	1.000	39341.25
0.000	8.400	5.500	0.900	2.500	0.750	4.950	0.950	6.000	1.000	42671.65

COST OF NEW POLICY IS 330320.469

RESERVOIR STORAGES AND RELEASES

137313792.000	11.400	74113664.000	10.690	133007616.000	9.000
121637376.000	9.800	91314176.000	16.790	154272384.000	8.040
105888384.000	9.800	94628480.000	16.290	154923968.000	10.000
128991744.000	9.800	126841600.000	22.514	232460944.000	9.550
120693888.000	9.800	126841600.000	30.960	273611520.000	21.030
99332352.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	10.000	3.900	0.900	0.250	0.750	2.950	2.450	3.750	1.000	71781.53
0.000	8.400	5.500	0.900	2.500	0.750	5.200	0.700	5.500	1.000	64485.29
0.250	8.400	5.500	0.900	2.500	0.750	4.700	1.200	6.000	1.000	66505.70
0.750	8.400	5.500	0.900	2.500	0.750	5.200	0.700	6.000	1.000	44472.20
0.750	8.400	5.500	0.900	2.500	0.750	4.950	0.950	6.000	1.000	39142.04
0.000	8.400	5.500	0.900	2.500	0.750	5.200	0.700	6.000	1.000	42268.34

COST OF NEW POLICY IS 328655.125

RESERVOIR STORAGES AND RELEASES

137313792.000	11.400	74716464.000	10.440	133612416.000	8.300
121032576.000	10.050	92523776.000	16.540	155481984.000	8.800
105283584.000	9.800	95233280.000	16.540	156739968.000	9.800
128991744.000	9.800	126841600.000	22.514	233484944.000	9.050
121298688.000	9.800	126841600.000	30.690	273611520.000	22.230
99937152.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	10.000	3.900	0.900	0.000	0.750	2.950	2.450	3.500	1.000	71391.03
0.000	8.650	5.250	0.900	2.250	0.750	5.450	0.450	5.300	1.000	64228.69
0.250	8.400	5.500	0.900	2.500	0.750	4.950	0.950	6.000	1.000	66099.90
1.000	8.400	5.500	0.900	2.500	0.750	5.450	0.450	5.750	1.000	44090.75
1.000	8.400	5.500	0.900	2.500	0.750	4.950	0.950	6.000	1.000	39142.84
0.000	8.400	5.500	0.900	2.500	0.750	5.450	0.450	6.000	1.000	42065.06

COST OF NEW POLICY IS 327518.312

RESERVOIR STORAGES AND RELEASES

137313792.000	11.400	75323264.000	10.190	134217216.000	8.570
120627776.000	10.300	92523776.000	16.790	157901184.000	9.050
104678784.000	9.800	94628480.000	16.790	159763968.000	9.050
128991744.000	9.800	126841600.000	22.314	233508944.000	19.080
121903488.000	9.800	126841600.000	30.440	273611520.000	25.530
100541952.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	10.000	3.900	0.900	0.000	0.750	2.700	2.450	3.250	1.000	72061.05
0.000	8.900	5.000	0.900	2.500	0.750	5.700	0.200	5.250	0.750	64025.08
0.250	8.400	5.500	0.900	2.500	0.750	5.200	0.700	6.000	1.000	65694.10
1.250	8.400	5.500	0.900	2.500	0.750	5.700	0.200	6.000	1.000	43472.20
1.250	8.400	5.500	0.900	2.250	0.750	5.200	0.700	6.000	1.000	39229.89
0.000	8.400	5.500	0.900	2.500	0.750	5.700	0.200	6.000	1.000	41861.75

COST OF NEW POLICY IS 326344.094

RESERVOIR STORAGES AND RELEASES

137313792.000	11.400	74113664.000	10.690	134822016.000	8.300
120427776.000	10.300	90830336.000	16.990	157780224.000	8.350
104073984.000	9.800	92935040.000	17.040	160247808.000	9.300
128991744.000	9.800	126841600.000	21.064	240081424.000	8.600
122508288.000	9.800	126341600.000	30.190	273611520.000	24.120
101146752.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	10.000	3.900	0.900	0.250	0.750	2.950	2.450	3.000	1.000	71341.53
0.000	8.900	5.000	0.900	2.500	0.750	5.900	0.000	5.500	1.000	63303.42
0.000	8.400	5.500	0.900	2.500	0.750	5.450	0.450	6.000	1.000	65787.50
1.500	8.400	5.500	0.900	2.500	0.750	5.900	0.000	5.750	1.000	43359.87
1.500	8.400	5.500	0.900	2.500	0.750	5.200	0.700	6.000	1.000	39041.17
0.000	8.400	5.500	0.900	2.500	0.750	5.900	0.000	6.000	1.000	41679.12

COST OF NEW POLICY IS 325517.625

RESERVOIR STORAGES AND RELEASES

138523392.000	11.150	75323264.000	9.940	135426816.000	8.050
121032576.000	10.550	92644736.000	16.740	158389824.000	8.350
104678784.000	9.800	94144640.000	17.290	161457408.000	9.050
129596844.000	9.800	126841600.000	21.564	240686224.000	8.850
123717888.000	9.800	126841600.000	29.940	273611520.000	24.430
102356352.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.250	10.000	3.900	0.650	0.000	0.500	2.700	2.450	2.750	1.000	72630.89
0.000	9.150	4.750	0.900	2.500	0.750	5.900	0.000	5.500	1.000	63268.09
0.000	8.400	5.500	0.900	2.500	0.750	5.700	0.200	6.000	1.000	65181.70
1.500	8.400	5.500	0.900	2.500	0.750	5.900	0.000	6.000	1.000	43034.87
1.750	8.400	5.500	0.900	2.500	0.750	5.450	0.450	6.000	1.000	39838.65
0.000	8.400	5.500	0.900	2.500	0.750	5.900	0.000	6.000	1.000	41479.12

COST OF NEW POLICY IS 324433.344

RESERVOIR STORAGES AND RELEASES

137313792.000	11.400	75323264.000	10.190	135426816.000	8.050
121032576.000	10.300	91435136.000	16.990	158989824.000	8.100
104678784.000	9.800	92451192.000	17.490	163150848.000	8.600
128991744.000	9.800	126841600.000	21.114	242379664.000	8.850
122508288.000	9.800	126841600.000	30.190	273611520.000	25.130
101146752.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	10.000	3.900	0.900	0.000	0.750	2.700	2.700	2.500	1.000	72076.85
0.250	8.900	5.000	0.900	2.500	0.750	5.900	0.000	5.250	1.000	63629.22
0.000	8.400	5.500	0.900	2.500	0.750	5.900	0.000	5.750	1.000	66217.06
1.250	8.400	5.500	0.900	2.500	0.750	5.900	0.000	6.000	1.000	42753.57
1.500	8.400	5.500	0.900	2.500	0.750	5.700	0.200	6.000	1.000	39834.56
0.000	8.400	5.500	0.900	2.500	0.750	5.900	0.000	6.000	1.000	41679.12

COST OF NEW POLICY IS 324195.406

RESERVOIR STORAGES AND RELEASES

137313792.000	11.400	75323264.000	10.190	136426816.000	8.050
121637376.000	10.050	90830336.000	17.240	159594624.000	7.850
105283584.000	9.800	91846392.000	17.490	163755648.000	8.600
123991744.000	9.800	126841600.000	21.114	243589264.000	8.600
121903488.000	9.800	126841600.000	30.440	273611520.000	25.630
100541952.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	10.000	3.900	0.900	0.000	0.750	2.700	2.950	2.250	1.000	72052.65
0.250	8.650	5.250	0.900	2.500	0.750	5.900	0.000	5.000	1.000	63689.54
0.000	8.400	5.500	0.900	2.500	0.750	5.900	0.000	5.750	1.000	65117.06
1.000	8.400	5.500	0.900	2.500	0.750	5.900	0.000	5.750	1.000	42577.76
1.250	8.400	5.500	0.900	2.500	0.750	5.900	0.000	6.000	1.000	38751.12
0.000	8.400	5.500	0.900	2.500	0.750	5.900	0.000	6.000	1.000	41779.12

COST OF NEW POLICY IS 323967.281

RESERVOIR STORAGES AND RELEASES

137313792.000	11.400	75323264.000	10.190	136031616.000	7.800
122242176.000	9.800	90225536.000	17.490	160804224.000	7.600
105283584.000	10.050	91846392.000	17.240	165570048.000	8.350
129596544.000	9.800	126841600.000	20.864	245403664.000	8.600
123113088.000	9.800	126841600.000	30.190	273611520.000	26.380
101751552.000	9.800	126841600.000	24.120	273611520.000	12.240

NEW POLICY

P21	P1B	P3B	P1F	P3A	P3D	P3C	P4C	P4E	P4G	COST
0.000	10.000	3.900	0.900	0.000	0.750	2.700	2.950	2.000	1.000	72072.65
0.250	8.400	5.500	0.900	2.500	0.750	5.900	0.000	4.750	1.000	63549.85
0.000	8.650	5.250	0.900	2.500	0.750	5.900	0.000	5.500	1.000	64896.75
1.250	8.400	5.500	0.900	2.500	0.750	5.900	0.000	5.750	1.000	42281.07
1.500	8.400	5.500	0.900	2.500	0.750	5.650	0.250	6.000	1.000	39753.21
0.000	8.400	5.500	0.900	2.500	0.750	5.900	0.000	6.000	1.000	41579.12

COST OF NEW POLICY IS 323234.687