Water Resources Management **17:** 337–353, 2003.
© 2003 *Kluwer Academic Publishers. Printed in the Netherlands.*

Folded Dynamic Programming for Optimal Operation of Multireservoir System

D. NAGESH KUMAR[1]* and FALGUNI BALIARSINGH[2]
[1] *Civil Engineering Department, Indian Institute of Science, Bangalore, India;* [2] *Civil Engineering Department, OUAT, Bhubaneswar, India*
(author for correspondence, e-mail: nagesh@civil.iisc.ernet.in, Fax: 91 80 3600 404)*

(Received: 15 October 2002; in final form: 5 June 2003)

Abstract. Dynamic Programming (DP) is considered as a good technique for optimal reservoir operation due to the sequential decision making and ease in handling non-linear objective functions and constraints. But the application of DP to multireservoir system is not that encouraging due to the problem 'curse of dimensionality'. Incremental DP, discrete differential DP, DP with successive approximation, incremental DP with successive approximation are some of the algorithms evolved to tackle this curse of dimensionality for DP. But in all these cases, it is difficult to choose an initial trial trajectory, to get at an optimal solution and there is no control over the number of iterations required for convergence. In this paper, a new algorithm, Folded DP, is proposed, which overcomes these difficulties. Though it is also an iterative process, no initial trial trajectory is required to start with. So, the number of iterations is independent of any initial condition. The developed algorithm is applied to a hypothetical reservoir system, solved by earlier researchers. Operating policy obtained using the present algorithm has compared well with that of the earlier algorithm.

Key words: algorithm, dynamic programming, multireservoir operation, optimization

1. Introduction

The importance and applicability of Dynamic Programming (DP), proposed by Bellman (1957), is well known for optimal control problems in various streams of engineering and management. The application of DP in water resources area was discussed by Yakowitz (1982) and Yeh (1985). They have also discussed about a hindrance to use DP due to the 'curse of dimensionality'. In a multireservoir system, it is necessary to obtain an operating policy (release policy) for all the reservoirs simultaneously, because the optimum condition of the system cannot be investigated by considering the reservoirs in isolation. In Discrete DP (DDP), the state variables (storage of reservoirs) are normally discretized. Dense discretization is preferred over the coarse one, to obtain an operating policy close to the global optimum. These two factors, simultaneous investigation of all the reservoirs (high dimensionality) of the system and dense discretization of storage state variables, are the root causes of the curse of dimensionality.

Various methods are adopted to tackle the curse of dimensionality in the past. No method has a clear superiority over others. The trade-off between accuracy and easiness is to be considered in the selection of a method. The idea of overcoming high dimensionality problem by successive approximation was given by Bellman (1957). Using this philosophy, Larson (1968) applied Dynamic Programming with Successive Approximation (DPSA) algorithm to a hypothetical four reservoirs system. Larson and Korsak (1970) provided the proof of convergence for DPSA. Incremental DP (IDP) was proposed by Hall *et al*. (1969) and applied to water resource problem. Discrete Differential DP (DDDP) was proposed by Heidari *et al*. (1971) and was applied to the same hypothetical reservoir system, adopted by Larson (1968). Incremental DPSA (IDPSA) was applied by Giles and Wunderlich (1981) to a reservoir system, operated by Tenessee valley authority. Another method, Binary State DP algorithm, was given by Ozden (1984). All the above methods are of discrete type.

In the class of continuous type, Progressive Optimality (Turgeon, 1981), State Incremental DP (Larson, 1968), and Differential DP were already widely applied to water resources problems. Perera and Codner (1998) took the help of two factors to improve the computational efficiency of stochastic DP model during operation of an urban water supply reservoir system. They assumed strong cross correlation of stream flow among the various sites and used corridor approach.

As the algorithm, proposed in this paper, is for multidimensional discrete deterministic DP, the discussion is restricted to this type. While the authors do not claim complete superiority of the present algorithm over the earlier ones in all respects, they demonstrate a clear superiority in many aspects. All the algorithms are of iterative type. An initial feasible trial trajectory was essential for starting the iterations in each and every one of the earlier algorithms. This trial trajectory is evaluated by the objective function value. In the process of iterations, a better trajectory is continuously obtained than in the previous one, which gives better objective function value. Ultimately, an optimal trajectory is arrived at. The number of iterations required to achieve the optimal trajectory depends on how far the chosen trial trajectory is from the optimal one. The present algorithm, Folded Dynamic Programming (FDP), is different from the earlier algorithms in two aspects. There is no need to assume any initial trial trajectory and the number of iterations for achieving optimal solution is considerably less.

Earlier algorithms were discussed in different manners in the earlier works. For better clarity amongst them and to delineate their differences from the present one, the earlier algorithms are briefly explained in one frame in Section 2. The proposed algorithm, FDP is explained in Section 3. FDP is applied to the reservoir system of Larson (1968) in Section 4. Conclusions from the study are given in Section 5.

1.1. EARLIER ALGORITHMS

1.1.1. *DDP*

The well-known backward recursive equations for conventional DDP are

$$f_t(S_t) = \max_{R_t \in A_t} \left[L_t\,(S_t,\, R_t) + f_{t+1}\,(S_{t+1}) \right] \qquad t = 0, 1, \ldots, T-1\,,$$

where $f_T(ST) = 0$

Subject to $S_{i,t+1} = S_{i,t} + I_{i,t} - R_{i,t} \qquad \forall i = 1, \ldots, M\,,$

where S_t is a vector of M dimensional storage states in M reservoirs at beginning of the time period t. There are T time periods in the operating horizon, designated by $0, 1, \ldots, T-2, T-1$. $f_t(S_t)$ is the maximum total return over the remaining periods $t, t+1, \ldots, T-1$ with S_t as initial storage vector state for M reservoirs. $L_t\,(S_t, R_t)$ is the return function from the system by the operation during time period t with S_t as the initial storage state vector and R_t as the release (decision) vector during time period t. A_t is the maximum release (canal capacity) value. If any other value is taken, there will not be much change in finding out the optimal policy. The dynamic behaviour of each reservoir in the system is shown with reservoir water balance equation. $S_{i,t}$ is the storage of reservoir i at beginning of time period t. $I_{i,t}$ is the combination of independent natural inflow and release from other reservoirs to the reservoir i during time period t. $R_{i,t}$ is release from reservoir i during time period t. Various losses, like evaporation, seepage etc. are presently ignored for simpler discussion and can be easily considered later during actual application.

Largely, the algorithms used for increasing the computational capability of DP, are of two types. These are incremental and successive approximation types. In both the cases, the storage state spaces are discretized by grid points into uniform increments, called state increments. The storage is allowed to change only from one grid point to the other. In the case of incremental type, only three neighborhood storage states (corridor) for the whole operating horizon are considered in each iteration. The best operating policy (trajectory) is the line joining the storage state of each time step from initial to final time period of the operating horizon. This will be the center of corridor for the next iteration. In successive approximation type, iterations are done for the full range of storage state of one reservoir at a time.

The IDP algorithm is explained in detail in the next section. Regarding the other algorithms, only the differences from IDP and the differences amongst the algorithms are discussed.

1.1.2. *IDP*

i. P is assumed as the initial trial trajectory for all the M reservoirs during the whole operating horizon. F is the objective function value of the reservoir system, when operated with this trial trajectory.

ii. A corridor is made by considering one feasible neighbouring storage state grid point on each side of P.
iii. Conventional DP is run through this corridor and the best trajectory is found, which is denoted by P'. F' is the objective function value corresponding to P'. In each time period, $3^M * 3^M$ matrix is required to store the return value. So total size of matrix, required for this algorithm is number of time periods $*3^M * 3^M$.
iv. The trajectory and corridor for the next iteration is to be found out depending on the value of F'', which is equal to $(F' - F)/F$. The formation of corridor around the trajectory P', now termed as P, is done with the same state increment as in the previous iteration, if $F'' \geq \xi$, where ξ is a predefined factor. Otherwise, a state increment less than that in the previous iteration is to be used. The step (iii) is repeated with this new corridor.
v. This process of iterations is stopped, when $F'' < \xi$ and the state increment is below the predefined value.

1.1.3. *DDDP*

The process of iteration is same as in IDP, the only difference being in forming the corridor around the trajectory. Here, there may be more than three grid points in the corridor, depending on the choice of the model user. Nopmongcol and Askew (1976) have rightly pointed out that DDDP is a generalized version of IDP.

1.1.4. *DPSA*

The iterations are done for one state variable at a time. While fixing the corridor around trial trajectory for first iteration, the entire possible grid points for the first state variable only are considered and the trajectory of all the remaining state variables is kept fixed. Then DP is run through this corridor. The best trajectory, obtained in first iteration will be the new trajectory for second iteration. In second iteration, second state variable is considered as the first state variable. In this way, first set of iterations continues till all the state variables are considered. Then, next set of iterations is done similar to the first one, with new trajectories. This process stops when two sets of consecutive iterations yield nearly same trajectories.

1.1.5. *IDPSA*

This is a combination of IDP and DPSA. The process of iterations is done in two tiers. In first tier, iteration starts with three feasible grid points as corridor, one on each side of trial trajectory of the first state variable, for the whole operating horizon. In subsequent iterations, the trajectory for the same state variable moves towards the optimal policy with better objective function value as in IDP. When there is no more improvement in the first state variable, the second state variable is considered as the first one. In second tier, the reservoirs are selected one after the

other, as in DPSA. The first and second tier iterations are applied repeatedly till the improvement of trajectory stops.

1.1.6. *Binary State DP*

Corridor is formed with one grid point on one side of trial trajectory, the side being fixed by previous iteration. Unlike in any incremental type DP, here only two grid points are considered instead of three.

More details of the algorithms presented in this section are available in the respective references.

2. Folded Dynamic Program (Present Algorithm)

Before discussing the algorithm of FDP, it is necessary to explain the way of finding maximum and minimum possible storage at beginning of each time period, hereafter called time step, of whole operating horizon for all the reservoirs. The operating horizon, the duration of reservoir operation, is considered as stage and storage of reservoir is considered as state variable in DP formulation. In physical terms, the storage state variable can be at any point between the dead storage level and full reservoir level. In the present FDP algorithm, the entire storage state space at each time period is required to be divided into four equal state increments to form five grid points. The storage can be changed from any grid point of one time step to any grid point of adjacent time step. This procedure is illustrated in results section.

Step-wise procedure of the proposed algorithm, FDP, is explained below.

i. Depending on the natural inflow, release capacity, and boundary condition of storage, the maximum and minimum possible storage values for each reservoir at every time step of operating horizon are found out.
ii. Considering the maximum and minimum possible storages as the two extreme grid points, three intermediate grid points are determined adopting a uniform state increment. This means, that the possible storage space at each time step is divided into four equal state increments to get the five grid points. Thus there will be totally $5 * M$ grid points for each time step. State increment is different for different time steps as also for different reservoirs. The mesh of these grid points for the whole operating horizon of all the reservoirs, forms the corridor.
iii. Conventional DP is run through this corridor to find the trajectory, P, which gives maximum objective function value, F.
iv. For finding the trajectory for next iteration, if this trajectory is either the minimum or maximum storage value, i.e., extreme grid points at any time step, these points are changed to the next interior grid points to form the revised trajectory. This revised trajectory will be the center of corridor for the next iteration.

v. In the next iteration, the state increment is halved at each time step. The corridor is formed by taking two state increments or grid points on each side of the trajectory. Then step (iii) is repeated to find the best trajectory, P', whose objective function value is F'.

vi. The iterations are continued with half value of state increments of those of the previous ones at each time step. There can be two stopping rules. First, the decrement of state increment at a time step stops, where state increment happens to be less than a predefined value. The iteration stops, when decrement of state increment process stops at each time step. Second, the iteration stops, when $F'' < \xi$ is satisfied, where of F'' is $(F' - F)/F$ and ξ is a predefined factor. In the present case, second stopping rule is applied.

2.1. NAME FOR THE ALGORITHM

If a flexible thread is folded twice, five points are obtained consisting of three folding points and two extremes. These five points denote the five storage states for the first iteration. Length between any two consecutive folding points is the value of state increment adopted. Taking any consecutive three points, if the thread is again folded, there will be two more folding points making up a total of five. Now the length between any two consecutive points is half of that of the first folding. By repeated foldings like this, we can reach any point of the whole of the thread. Authors visualized the importance of this folding phenomenon to reach any value of feasible storage from within the range of storage state. Hence the algorithm is called Folded Dynamic Programming (FDP).

3. Application of FDP

FDP is applied to the hypothetical reservoir system used by Larson (1968). A number of earlier methods were also applied to this system. So it is preferred to apply the present algorithm also to the same system, to facilitate direct comparison of this algorithm with the earlier ones. In this problem, the optimum operation over 24 hr of a hypothetical multipurpose four-reservoir system is to be determined. The reservoir network, which contains both series and parallel connections, is shown in Figure 1. In this optimization, use of water for power generation, irrigation, flood control and recreation are considered. Storages in the reservoirs are considered as the state variables for the problem. The volume of water in the reservoir i for time period t is denoted as $S_{i,t}, i = 1, 2, 3, 4$, where $S_{i,t}$ is expressed in normalized units. On the basis of flood control considerations, a maximum water level for each reservoir is established.

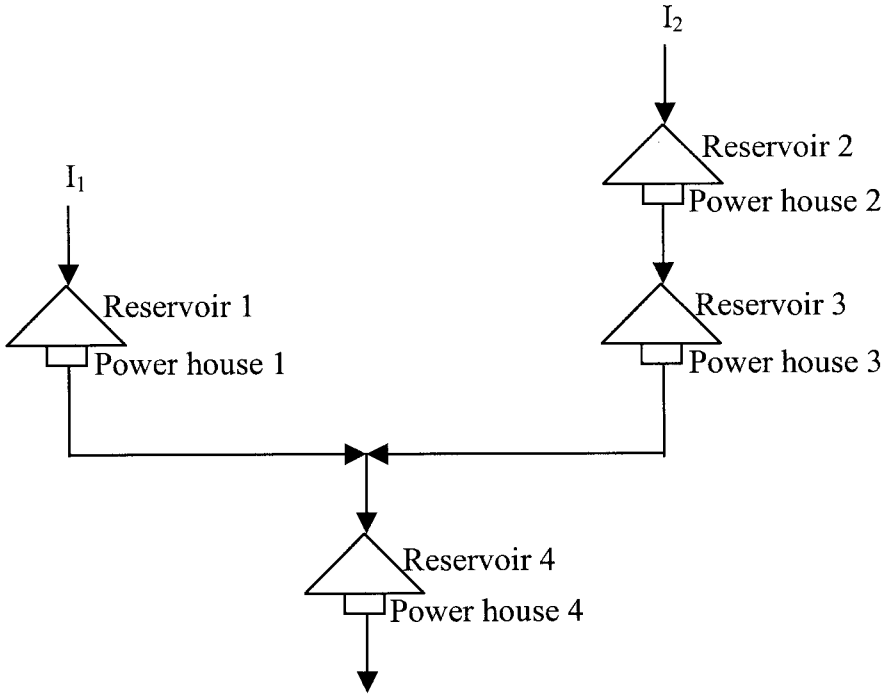


Figure 1. Multi reservoir system network (Larson, 1968).

Twelve time periods of 2 hr each are considered for operating horizon, designated by 0, 1, . . ., 11. The constraints of four dimensional storage vector are

$$0 \le S_{1,t} \le 10, \quad 0 \le S_{2,t} \le 10, \quad 0 \le S_{3,t} \le 10, \quad 0 \le S_{4,t} \le 15; \qquad \text{for } t = 0, 1, \ldots, 12 . \tag{1}$$

The control variable is taken as the release from each of the four reservoirs. These quantities are also expressed in the same normalized units. The variables $R_{i,t}$, $i = 1, 2, 3, 4$, specify the release from the reservoir i during time period t. In this study, each time period is of 2 hr duration.

Maximum release from each reservoir is determined by the capacity of the turbines and minimum release is determined by considering the use of the downstream flow for navigation, conservation, and municipal and industrial water supplies. The constraints on release from four reservoirs are

$$0 \le R_{1,t} \le 3, \quad 0 \le R_{2,t} \le 4, \quad 0 \le R_{3,t} \le 4, \quad 0 \le R_{4,t} \le 7; \qquad \text{for } t = 0, 1, \ldots, 11 . \tag{2}$$

The system dynamic equations for the reservoirs are given below. They are applicable for whole operating horizon, i.e., $t = 0, 1, \ldots, 11$.

$$S_{1,t+1} = S_{1,t} + I_{1,t} - R_{1,t} \qquad t = 0, 1, \ldots, 11 \tag{3}$$

$$S_{2,t+1} = S_{2,t} + I_{2,t} - R_{2,t} \qquad t = 0, 1, \ldots, 11 \tag{4}$$

$$S_{3,t+1} = S_{3,t} + R_{2,t} - R_{3,t} \qquad t = 0, 1, \ldots, 11 \tag{5}$$

$$S_{4,t+1} = S_{4,t} + R_{1,t} + R_{3,t} - R_{4,t} \qquad t = 0, 1, \ldots, 11\ , \tag{6}$$

where $I_{1,t}$ and $I_{2,t}$ are inflows into reservoirs 1 and 2, respectively.

The desired state vectors at the beginning and end of operating horizon are

$$S_0 = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \end{bmatrix} \qquad S_{12} = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 7 \end{bmatrix}\ . \tag{7}$$

The objective of this multireservoir system is to maximise the benefit from irrigation and hydropower generation. So the objective function is

$$F = \sum_{t=0}^{11} \sum_{i=1}^{4} b_{i,t} * R_{i,t} + \sum_{i=0}^{11} b_{5,t} * R_{4,t}\ , \tag{8}$$

where $b_{i,t}$ is the benefit per unit flow for reservoir i during time period t. Benefits from the flow over a given 2 hr period are assumed to be a linear function of the flow. Irrigation benefits are considered only for the release from the reservoir 4. Benefit for irrigation from the releases of reservoir 4 is denoted by $b_{5,t}$. Details of the problem are given by Larson (1968) and Heidari *et al*. (1971).

4. Results

It is necessary to find out the minimum and maximum possible storages for each reservoir at each time step. In each time step of whole operating horizon, storage corresponding to two extreme grid points, which may or may not be dead storage level or full reservoir level are found out in such a way that the chance of falling into the same storage state at the last time step as that of the first time step of operating horizon in the process of DP is avoided. The following equations are followed to find out the storage states corresponding to these two extreme grid points.

$$S_{\max,i,t+1} = S_{\max,i,t} + I_{i,t} \qquad \forall\ i \text{ and } t\ =\ 0,\ 1,\ \ldots,\ T-2 \tag{9}$$

$$S_{\min,i,t+1} = S_{\min,i,t} + I_{i,t} - R_{i,t} \qquad \forall\ i \text{ and } t\ =\ 0,\ 1,\ \ldots,\ T-2 \tag{10}$$

$$S_{\max,i,t+1} = S_{\max,i,t} + I_{i,t} - R_{i,t} \qquad \forall\ i \text{ and } t = 0,\ 1,\ \ldots,\ T-2 \tag{11}$$

$$S_{\min,i,t+1} = S_{\min,i,t} + I_{i,t} \qquad \forall\ i \text{ and } t = 0,\ 1,\ \ldots,\ T-2 \tag{12}$$

$$R_{i,t} \leq RC_{i,t} \qquad \forall\ i,\ t \tag{13}$$

$$S_{DSL,i} \leq S_{\max,i,t} \leq S_{FRL,i} \qquad \forall\ i,\ t \tag{14}$$

$$S_{DSL,i} \leq S_{\min,i,t} \leq S_{FRL,i} \qquad \forall\ i,\ t\ , \tag{15}$$

where

$S_{\max,i,t}$ = maximum possible storage of reservoir i at beginning of time period t;
$S_{\min,i,t}$ = minimum possible storage of reservoir i at beginning of time period t;
$I_{i,t}$ = the inflow volume to the reservoir i during time period t;
$R_{i,t}$ = the release from reservoir i during time period t;
$RC_{i,t}$ = the maximum release capacity of reservoir i during time period t;
$S_{DSL,i}$ = the reservoir storage of reservoir i at dead storage level;
$S_{FRL,i}$ = the reservoir storage of reservoir i at full reservoir level.

Equations (9) and (10) are used during forward pass and Equations (11) and (12) are used during backward pass. The aim of these four equations is to get the maximum possible storage as high as feasible and the minimum possible storage as low as feasible. Equation (13) shows the limit of release of each reservoir at each time step. In all these four equations, the storage should also satisfy the physical limits, generally dead storage level (DSL) and full reservoir level (FRL) by Equations (14) and (15).

For illustration, storage limits obtained for reservoir 1 for each time step are shown in Figure 2. Here the operating horizon is twelve time periods of 2 hr each, giving 13 time steps from 0 to 12. The desired storage states are 5 units at both time steps 0 and 12. So both minimum and maximum possible storages at the time steps of both 0 and 12 are fixed as 5 units. During forward pass, maximum possible storage for successive time steps was obtained using Equation (9) and shown as curve A in Figure 2. It varies from 5 to 10 units during time step 0–3 and then continues to be 10 up to 12th time step. Similarly, the minimum possible storage was obtained by applying Equation (10) and shown as curve B in Figure 2. It varies from 5 to 0 units during time step 0–5 and then continues to be at 0 up to 12th time step. The storage state is restricted at 0 and 10 to satisfy the condition laid by storage constraint for reservoir 1 (Equation (1)). The desired storage state at time step 12 is 5 as per the Equation (7), i.e., the minimum and maximum possible

Table I. Maximum/minimum possible storage for each reservoir in each time step

Reser-	Time steps												
voir	0	1	2	3	4	5	6	7	8	9	10	11	12
1	5/5	7/4	9/3	10/2	10/1	10/0	10/0	10/0	9/0	8/0	7/1	6/3	5/5
2	5/5	8/4	10/3	10/2	10/1	10/0	10/0	10/0	9/0	8/0	7/0	6/2	5/5
3	5/5	9/1	10/0	10/0	10/0	10/0	10/0	10/0	10/0	10/0	10/0	9/1	5/5
4	5/5	12/0	15/0	15/0	15/0	15/0	15/0	15/0	15/0	15/0	15/0	14/0	7/7

storages at this time step should be 5. At the end of forward pass, as the minimum and maximum possible storages are obtained as 10 and 0 respectively, the backward pass is to be carried out. During backward pass, curves C and D of Figure 2 are obtained for maximum and minimum possible storage by using Equations (11) and (12), respectively. Then the final minimum possible storage is found out by taking the higher value of minimum possible storage states obtained by forward and backward pass (comparing curve B and D of Figure 2), which is shown as curve A in Figure 3. The final maximum possible storage is obtained by taking the lower value of maximum possible storage state, obtained by forward and backward pass (comparing curves A and C of Figure 2), which is shown as curve E of Figure 3. At each time step, the storage space between minimum and maximum possible storages, AE, is divided into four equal state increments, there by creating three intermediate grid points. By joining the corresponding points of all the time steps from 0 to 12, curves B, C, and D are formed. In this way, the mesh of five grid points at each time step for all twelve periods for the first reservoir are found. The mesh of all the four reservoirs, obtained similarly, forms the corridor for first iteration of FDP. Additional details on practical problems in choosing the maximum and minimum storages for the grid generation are discussed in Baliarsingh (2000).

Depending on full range of possible inflows and possible releases from the reservoirs, maximum and minimum possible storage values are found and shown in Table I for all the reservoirs.

The corridor of five grid points with equal storage state increments at each time step is formed. The iteration continues as per the algorithm presented in the previous section. The state increments are halved in each consecutive iteration. For example, the state increments during first five iterations of reservoir 2 at time step 1 are 4, 2, 1, 0.5, respectively, and at time step 3 are 8, 4, 2, 1, respectively. If ξ is taken as 0.002, the process converges at 5th iteration and 398.0 units is obtained as the objective function value. If ξ is taken as 0.0004, the process converges at 7th iteration with the objective function value as 398.7. This is only 0.64% less than global optimum value of 401.3, obtained by earlier methods. The objective function value and rate of convergence are shown in Figure 4. Although, development of trajectory in each iteration is obtained from the solution, trajectories

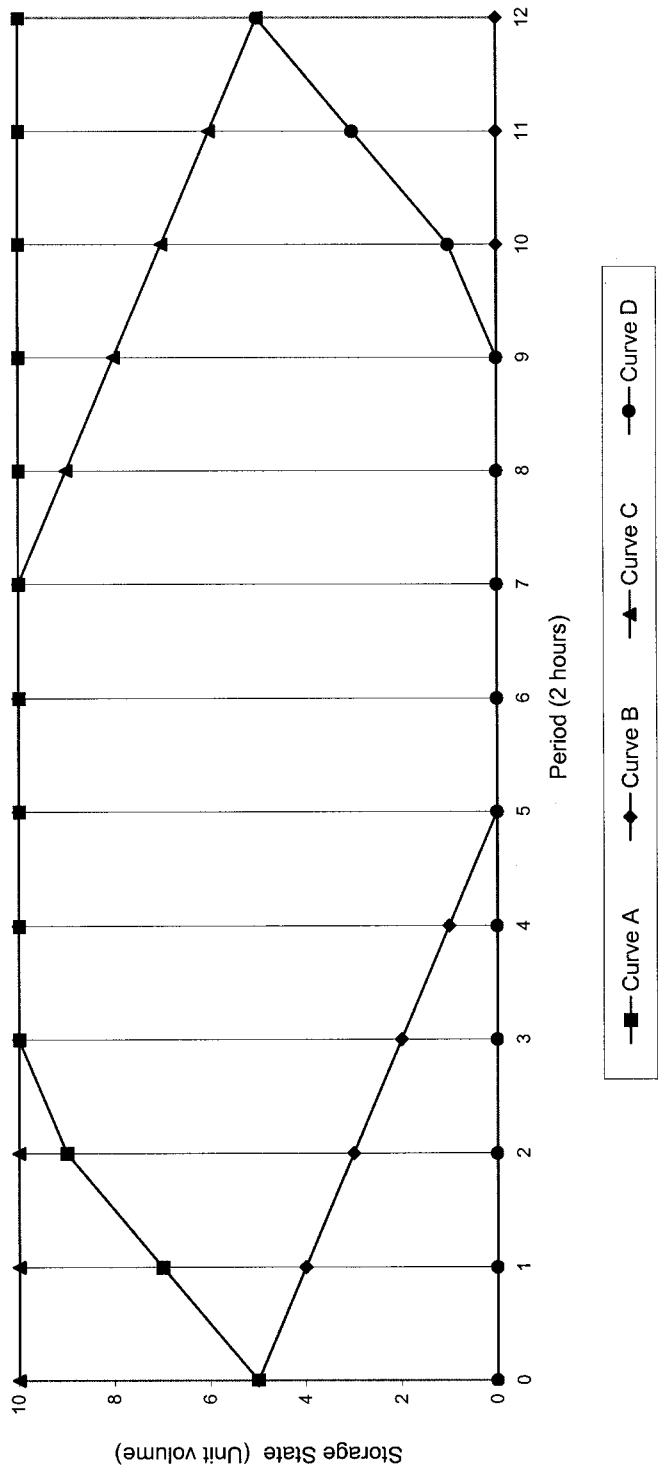


Figure 2. Maximum and minimum possible storage for reservoir 1 in forward and backward passes.

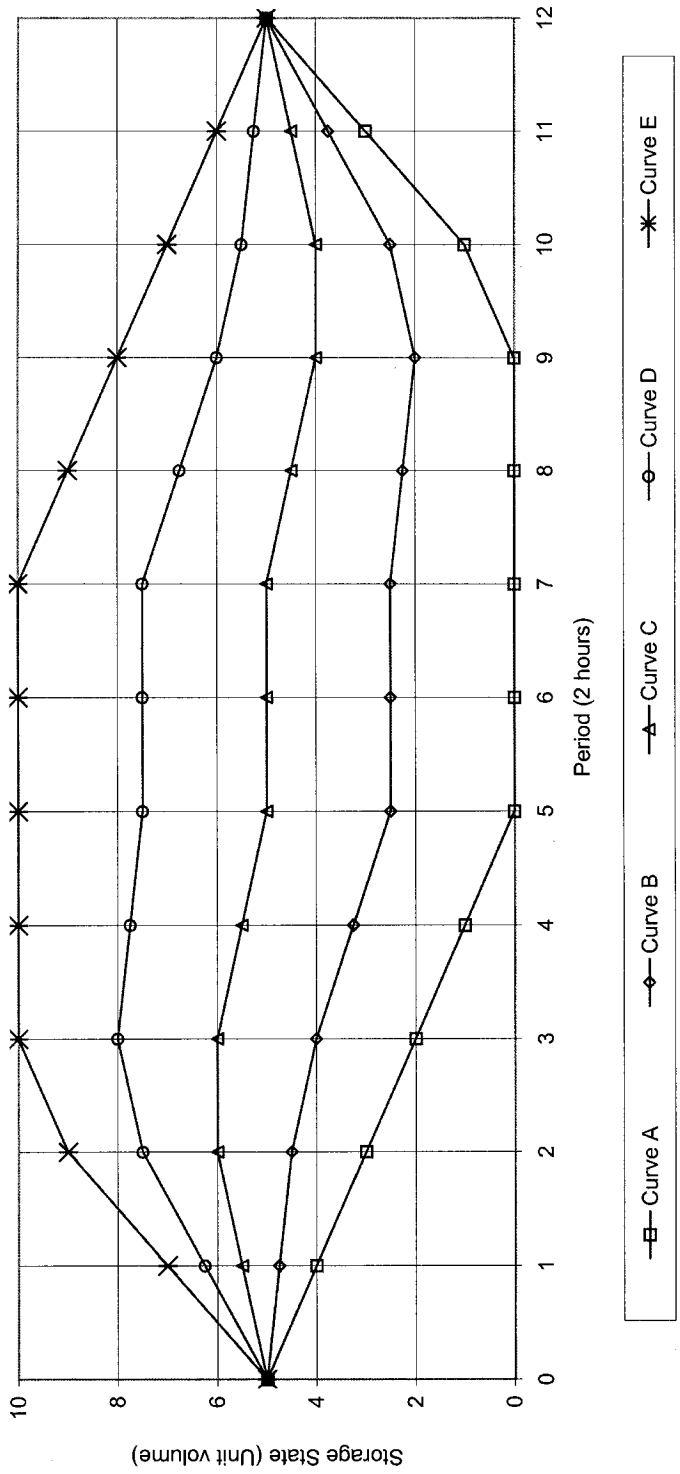


Figure 3. Grid point mesh of reservoir 1 for first iteration of folded dynamic programming.

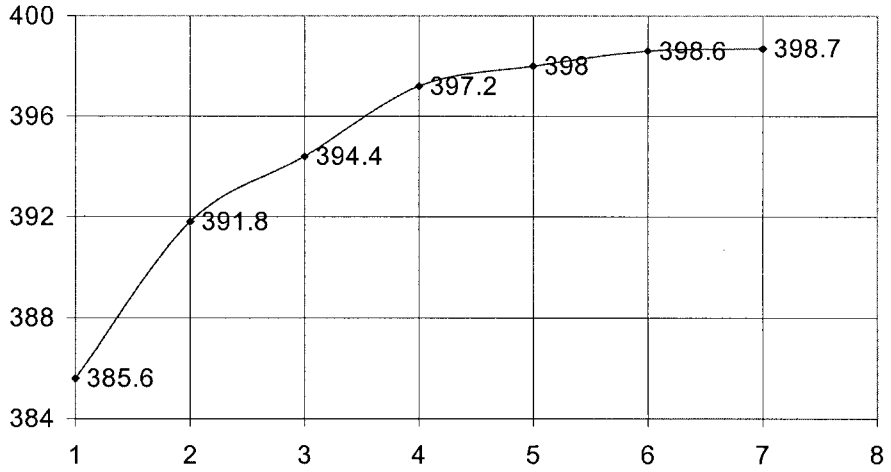


Figure 4. Objective function value at different iterations.

for the alternative iterations, i.e., 1, 3, 5 of each individual reservoir are shown in Figures 5a–d.

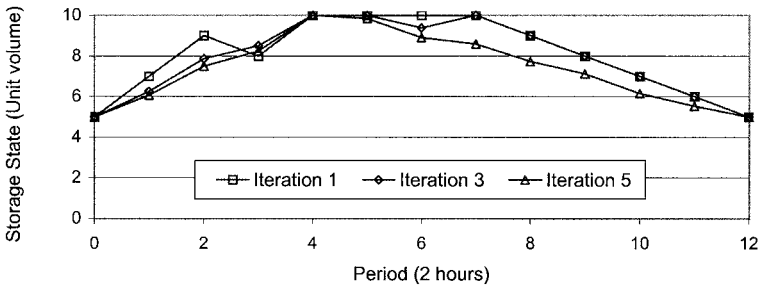
Optimal operating policy obtained using FDP is presented in Figure 6 for the four reservoirs. For comparison, final recommended trajectory, obtained by Heidari *et al.* (1971) using DDDP is also presented in Figures 6a–d for the four reservoirs and they compared well.

5. Discussions

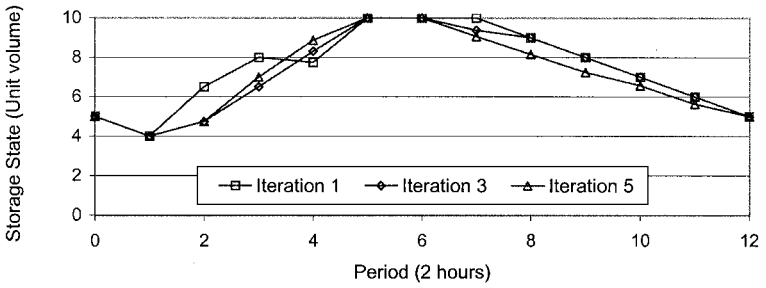
In earlier algorithms, the convergence depends on the proximity of the initial trial trajectory to the optimal trajectory. But in the present method, there is no such depending factor for convergence. From the algorithm, it can be seen that the present method takes remarkably less number of iterations than in the earlier methods to converge. However, rate of improvement slows down at higher number of iterations, as can be seen from Figure 4.

When DP is run with the mesh of grid points (corridor) generated by FDP, the chance of sticking to the storage state value at all the time steps of operating horizon is ruled out, as the storage value corresponding to minimum storage at each time step is not the same. It can also be noticed that by this process the state increment at each time step is not same and the process ensures arriving at the desired storage values at first and last time periods of operating horizon.

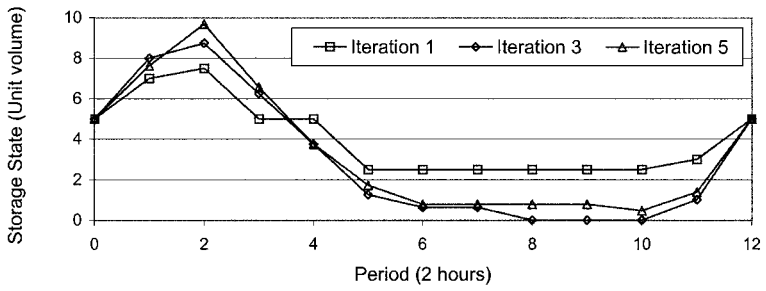
The only disadvantage in the present algorithm is that there is no guarantee of reaching the global optimum, which is also not possible with most of the earlier methods. Heidari *et al.* (1971) have agreed in their paper that the global optimum was found by luck, as the optimum trajectory happens to fall only on integer value and they have chosen state increment as 1.0. While solving with a state increment



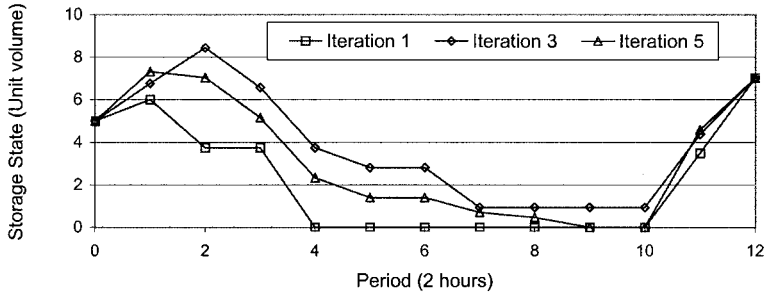
a. Reservoir 1



b. Reservoir 2

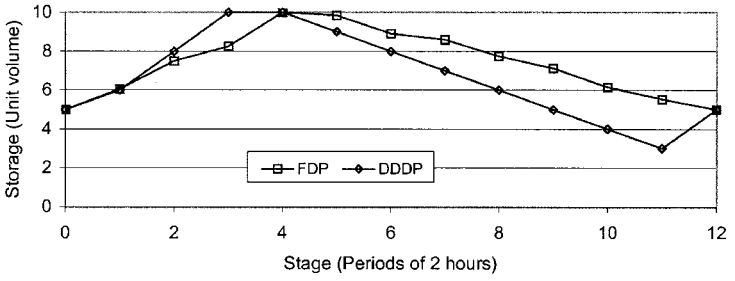


c. Reservoir 3

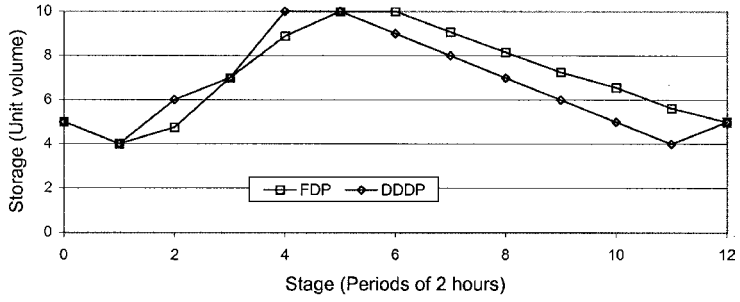


d. Reservoir 4

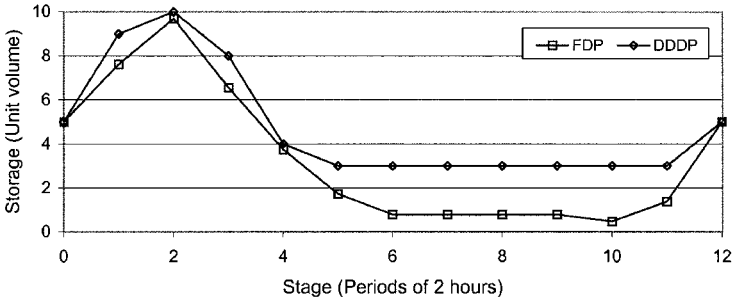
Figure 5. Trajectory of the system at various iterations for each reservoir.



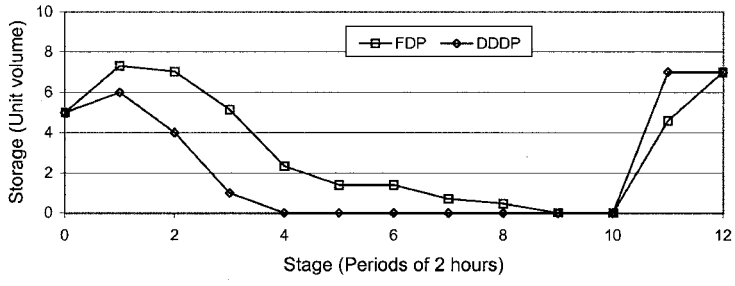
a. Reservoir 1



b. Reservoir 2



c. Reservoir 3



d. Reservoir 4

Figure 6. Comparison of recommended final policy from FDP and DDDP for 4 reservoir system.

of 1.3, optimum objective function value obtained using FDP was 399.06, which is very close to the global optimum of 401.3.

The required matrix size in the present algorithm is the number of time periods $*5^M * 5^M$ instead of number of time periods $*3^M * 3^M$ for IDP. So in the application of FDP, the required matrix size is $12 \times 625 \times 625$. In case of multireservoir system, the matrix size increases exponentially. In this situation of large number of reservoirs in the system, FDP can be applied with successive approximation, i.e. considering only one dimension at a time. So the matrix size in each iteration is limited to number of time periods $*5 * 5$.

6. Conclusions

A new algorithm, Folded Dynamic Programming, is presented to overcome the curse of dimensionality inherent in dynamic programming. In the earlier methods, an initial trial trajectory is essential to start. The trial trajectory moves towards the global optima in the grid of state increments. Trajectory may fall into local optima, if it is in between initial trial trajectory and global one. Bellman has suggested solving the problem with different trial trajectories. But if the dimension, i.e., number of reservoirs in the system, is large, there is no clue as to the number of trial trajectories required to ensure the global optimum condition. The above inconvenience can be avoided in the present algorithm, as initial trajectory is not at all necessary. Proposed algorithm is applied to a hypothetical multi reservoir system from the literature. Earlier algorithms require 7 to 18 iterations to reach optimal solution, depending on the initial trial trajectory and state increment. The present algorithm requires only 5 iterations for the same solution without requiring any initial trajectory. It is concluded that FDP can be used as an improvement to IDP. In the case of higher dimensional problems, FDP can be solved by successive approximation.

References

Baliarsingh, F.: 2000, 'Long-term and Short-term Optimal Reservoir Operation for Flood Control', *Doctoral Thesis*, Indian Institute of Technology, Kharagpur, India.

Bellman, R.: 1957, *Dynamic Programming*, Princeton University Press, Princeton, N.J.

Giles, J. E. and Wunderlich, W. O.: 1981, 'Weekly multipurpose planning model for TVA reservoir system', *J. Water Resour. Plng. Mgmt. ASCE* **107**(WR2), 495–511.

Hall, W. A., Tauxe, G. W. and Yeh, W. W.-G.: 1969, 'An alternate procedure for the optimization of operations for planning with multiple river, multiple purpose systems', *Water Resour. Res.* **5**(6), 1367–1372.

Heidari, M., Chow, V. T., Kokotovic, P. V. and Meredith, D. D. 1971, 'Discrete differential dynamic programming approach to water resources systems optimization', *Water Resour. Res.* **7**(2), 273–282.

Larson, R. E.: 1968, *State Increment Dynamic Programming*, Elsevier, New York, U.S.A.

Larson, R. E. and Korsak, A. J.: 1970, 'A dynamic programming successive approximations technique with convergence proofs', *Automatica* **6**, 245–252.

Nopmongcol, P. and Askew, A. J.: 1976, 'Multilevel incremental dynamic programming', *Water Resour. Res.* **12**(6), 1291–1297.

Ozden, M.: 1984, 'A Binary State DP algorithm for operation problems of multireservoir systems', *Water Resour. Res.* **20**(1), 9–14.

Perera, B. J. C. and Codner, G. P.: 1998, 'Computational improvement for stochastic dynamic programming models of urban water supply reservoirs', *J. Amer. Water Resour. Assoc.* **34**(2), 267–278.

Turgeon, A.: 1981, 'Optimal short-term hydro scheduling from the principle of progressive optimality', *Water Resour. Res.* **17**(3), 481–486.

Yakowitz, S.: 1982, 'Dynamic programming applications in water resources', *Water Resour. Res.* **18**(4), 673–696.

Yeh, W. W.-G.: 1985, 'Reservoir management and operations models a state-of-the-art review', *Water Resour. Res.* **21**(12), 1797–1818.