2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Optimal Management of a Two Dam System via Stochastic Control: Parallel Computing Approach

Boris Miller† and Daniel McInnes‡

Abstract— **In this paper we consider a model for the optimal management of a two dam system. Each dam is modelled via a continuous-time controlled Markov chain on a finite control period and linked to the other dam via a state and time dependent water transfer control. The consumption control for the dam system is provided by a time and state dependent price feedback control. This price feedback control takes into account the active seasonal demands of customers. We consider the case where inflow processes and evaporation for each dam are non-stationary as are the customer demands. The general approach to the solution of this problem is to consider this stochastic optimisation problem in the average case and solve it using the dynamic programming method. We show that the use of parallel computing techniques leads to substantial savings in calculation times for the solution of the optimal controls and demonstrate this via a numerical example.**

I. INTRODUCTION

For many cities, even those with abundant seasonal rainfall, dams or systems of dams are a crucial piece of water supply infrastructure. They should ensure a reliable water supply for human consumption, industry and agriculture as well as potentially mitigating the threat of major floods. The optimal control of this infrastructure, taking into account these various demands, is essential for the long term viability of the water resource.

Continuous-time Markov decision processes (MDP) are typically used to approach problems of this type (for examples see [1], [2] and [3]). Problems that have been approached this way include communications engineering, queuing systems and epidemic control [4] (see [8] in [4] for epidemic control). In this vein, attention has also been given to dams, especially single large dams, with inflow processes modelled as Weiner or compound Poisson processes and either infinite or finite capacity (for examples see [5], [6], [7], [8] and [9]).

The above examples use very simple threshold control models, that is they use long-run average criterion as the main optimality criterion. These models are problematic for a number of reasons. They generally assume an infinite time horizon for control and stationary inflow data. In reality we have to control over finite time-frames and the inflow from rain is usually highly seasonal. An infinite time horizon also means that inflows must be greater than or equal to outflows or the system will run dry in finite time. It should be clear, however, that in the short-term, outflows may well exceed inflows. It is also extremely important that water is available in the short-term as customers demand it and this should be taken into account in the optimality conditions. Finally, the long-term average criterion does not take into account the costs of transient states and the resources required for these transitions [10].

We consider the optimal control and numerical treatment of a two dam system. The optimal management of these dams is handled by state and time dependent price and water transfer controls. We approximate the level of each dam by N discrete levels and model the dam level processes as continuous-time controlled Markov chains. The solution is then found by looking at the average case of the stochastic control problem with integral and terminal optimality criteria and solving it via dynamic programming. This type of problem has been solved for server queuing systems and in general by Miller [10] and Miller et.al [4], [11]. The general framework for the solution of this type of problem is well laid out in [12].

In the following sections we will cover the theoretical basis for the two dam model and its solution, the computational approach that we have been working on thus far and finally some numerical results to give an indication of how effective parallelization of parts of our code has been.

II. THE 2-DAM MODEL

This model is covered in depth in [13], so here we give a brief account of the model. We begin by making some simplifying assumptions about the dams. First we assume that each dam has independent natural inflow and outflow processes. Secondly we assume that the consumption in each dam depends on a time and joint-state dependent price. We likewise assume that water transfers between dams depend on time and the joint-state of the dams.

We approximate the level in each dam by discretizing it into $N+1$ levels or states, $N<\infty$, and let $L_i(t)\in\{0,...,N\}$, $i=1,2$ be an integer valued random variable describing the state of dam i at time t. The martingale approach in [14] allows us to describe the $N+1$ states by the unit vectors in $\mathbb{R}^{N+1}$, giving $S_i=\left\{e_0^i,...,e_N^i\right\}$ as the set of unit vectors for

*This work was supported in part by Australian Research Council Grant DP0988685 and Russian Foundation for Basic Research Grant 10-01-00710.

†Boris Miller is with the School of Mathematical Sciences, Monash University, Clayton, VIC 3800, Australia and Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow, Russia. Boris.Miller@monash.edu

‡Daniel McInnes is with the School of Mathematical Sciences, Monash University, Clayton, VIC 3800, Australia. Corresponding author. Daniel.McInnes@monash.edu

978-1-61284-799-3/11/$26.00 ©2011 IEEE

dam i.

Define $X_i(t)$, $i = 1,2$, where $\{X_i(t) \in S_i, t \in [0,T]\}$ for $T < \infty$ on the probability space $\{\Omega, \mathscr{F}, \mathbb{P}\}$, as a controlled jump Markov process with piecewise right-continuous paths. This process is for the change in level of dam i. We make the following assumptions about the price control, $p(t) = p(t, \mathbf{X}_t)$ and the transfer control, $u_{i,j}(t) = u_{i,j}(t, \mathbf{X}_t)$, between dams i and j, $i, j = 1,2$ and $i \neq j$.

Assumption 2.1: Assume that the set of admissible controls, $\bar{P} = p(\cdot)$ and $\bar{U} = \{u_{i,j}(\cdot) : i, j = 1,2; i \neq j\}$ are sets of $\mathscr{F}_t^{\mathbf{X}}$-predictable controls taking values in $P = \{p \in [p_{min}, p_{max}]\}$ and $U = \{u \in [0,1]\}$ respectively, where $\mathbf{X} = X_1 \otimes X_2$.

Remark 2.2: If the history of the jump process from time 0 to t is denoted $\mathbf{X}_0^t = X_{1,0}^t \otimes X_{2,0}^t$, then assumption 2.1 ensures that our controls, $p(t, \mathbf{X}_0^t)$ and $u_{i,j}(t, \mathbf{X}_0^t)$ are measurable with respect to t and $\mathbf{X}_0^t$ (for detail see [13]). In fact, since we assume that the controls are $\mathscr{F}_t^{\mathbf{X}}$-predictable, which means in essence that they are measurable with respect to $X(t-)$, we can choose the control at time t based on the history of the process up to X_{t-}, the state just before time t.

A. DAM SYSTEM DYNAMICS

For this model we assume that we can approximate the inflow and outflow processes of each dam by general $\mathscr{F}_t^{\mathbf{X}}$-predictable counting processes with unit jumps, $Y_{in}^i(t)$ and $Y_{out}^i(t)$, $i, = 1,2$, respectively. The intensity of inflows comes from the deterministic intensity of natural inflows, $\lambda_i(t)$, and the intensity of inflows from the other dam, $u_{j,i}(t)$. The intensity of outflows comes from the deterministic intensity of evaporation, $\mu_i(t)$, the intensity of consumption, $C_i(t) = C_i(t, p(t), \mathbf{X}_t)$ and the intensity of transfers to the other dam, $u_{i,j}(t)$. This gives us two processes,

$$Y_{in}^i(t) = \int_0^t (\lambda_i(s) + u_{j,i}(s)) I\{L_i(s) < N\}\, ds + M_{in}^i(t),$$

and

$$Y_{out}^i(t) = \int_0^t (\mu_i(s) + C_i(s) + u_{i,j}(s)) I\{L_i(s) > 0\}\, ds + M_{out}^i(t),$$

where $M_{in}^i(t)$ and $M_{out}^i(t)$ are square integrable martingales with respective quadratic variations

$$\langle M_{in}^i \rangle_t = \int_0^t (\lambda_i(s) + u_{j,i}(s)) I\{L_i(s) < N\}\, ds$$

and

$$\langle M_{out}^i \rangle_t = \int_0^t (\mu_i(s) + C_i(s) + u_{i,j}(s)) I\{L_i(s) > 0\}\, ds.$$

The martingales $M_{in}^i(t)$ and $M_{out}^i(t)$ are zero mean random processes which provide the perturbation about the mean intensity of inflows and outflows respectively. Since these are counting processes and the deterministic parts of each are non-decreasing, Doob's decomposition theorem gives us this semi-martingale representation for each process. So now the approximate dynamics for each dam in our model are given by

$$L_i(t) = Y_{in}^i(t) - Y_{out}^i(t).$$

For a more detailed explanation of these approximations please see [13].

B. DAM SYSTEM AS A SYSTEM OF CONTROLLED MARKOV CHAINS

With the approximations in II-A we can make the following proposition regarding each dam in the system.

Proposition 2.3: Given the approximate dynamics for the i^{th} dam, as stated in II-A, in a system of 2 dams, the controlled process for this dam is represented by a controlled Markov chain with $N+1$ states and the matrix $(N+1) \times (N+1)$, describing the infinitesimal generator of the corresponding Markov chain,

$$A_i(t, p(t), u_{i,\cdot}(t), u_{\cdot,i}(t)) = A_i(t, C(t,p), u_{i,\cdot}(t), u_{\cdot,i}(t)),$$
$$A_i(t, C, u_{i,\cdot}, u_{\cdot,i}) =$$

$$\begin{pmatrix} -I_i[0,\cdot] & O_i[1,\cdot] & \dots & 0 & 0 \\ I_i[0,\cdot] & -(O_i[1,\cdot] + I_i[1,\cdot]) & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & -(O_i[N-1,\cdot] + I_i[N-1,\cdot]) & O_i[N,\cdot] \\ 0 & 0 & \dots & I_i[N-1,\cdot] & -O_i[N,\cdot] \end{pmatrix}$$

where $I_i[x,\cdot] = \lambda_i + u_{j,i}$ and $O_i[x,\cdot] = C_i + \mu_i + u_{i,j}$ when dam i is in state $X_i = x$ and the state of the other dam is represented by a dot. One can say that O_i corresponds to the outflow and I_i to the inflow. The column number corresponds to the current state of the i^{th} dam and the column entries add to zero.

Proof: The proof of proposition 2.3 follows closely the method of proof of the generator for a controlled queuing system given in [10]. The difference is that we have a link to the other dam via joint state dependency of price and transfer controls, but as these are $\mathscr{F}_t^{\mathbf{X}}$-predictable this does not affect the proof.

C. CONTROLLED DEMAND FUNCTIONS

The innovation of our method is to take into account the active demands of users on the system. We do this through a price feedback control, $p(t)$, but it is more convenient and intuitive to calculate this optimal price structure through the vehicle of optimal controlled consumption, $C(t, p(t))$. Here we derive these function. What we seek is a price structure for users of the system.

Consider the i^{th} dam and let there be n sectors or consumers in each dam. These users have their own seasonal demand intensity, $\bar{x}_{i,k}(t)$, $k = 1, \dots, n$. To control this demand intensity for each sector we need to find an optimal demand intensity, $x_{i,k}$, $k = 1, \dots, n$. For this model we define this through the utility function

$$\min_{x_{i,k}} f_{i,k}(x_{i,k})$$

where

$$f_{i,k}(x_{i,k}) = \alpha(x_{i,k} - (1-r)\bar{x}_{i,k}(t))^2 + p(t)x_{i,k}.$$

Here α is a parameter setting a limit on consumption above net natural flows and r is a minimum demand reduction target. So, differentiating $f_{i,k}(x_{i,k})$ and solving for $x_{i,k}$ gives $x_{i,k}(t,p(t)) =$

$$((1-r)\bar{x}_{i,k}(t) - \frac{p(t)}{2\alpha})I\left\{(1-r)\bar{x}_{i,k}(t) - \frac{p(t)}{2\alpha} \geq 0\right\}.$$

Now this is the optimal demand intensity for the k^{th} sector, so for the i^{th} dam the optimal intensity of demand is

$$C_i(t,p(t)) = \sum_{k=1}^{n} x_{i,k}(t,p(t)).$$

Since $p(t) = p(t,\mathbf{X}(t))$ we now have a vector of optimal demand intensities for the i^{th} dam depending on the joint state of the system. This also allows us to formulate maximum and minimum optimal demand intensities by taking the extreme values of $p(t) \in [p_{min}, p_{max}]$, giving

$$C_{i,max}(t,p_{min}) \geq C_i(t,p(t)) \geq C_{i,min}(t,p_{max})). \qquad (1)$$

Using (1) we can define piecewise solutions for $C_i(t,p(t))$ via dynamic programming.

III. DYNAMIC PROGRAMMING AND OPTIMAL CONTROL

The method of solution for this type of problem is well established. Miller has solved this problem for single server queuing systems in [10], we have solved it for a single large dam in [15] and for a system of multiple connected dams in [13]. Here we describe the solution for the two dam system modelled in II. Our solution will give us the optimal price structure and the optimal water transfer controls for our system.

A. GENERAL PERFORMANCE CRITERIA

Let $f_0(s,\mathbf{X}_s) = f_0(s,p(s),u_{1,2}(s),u_{2,1}(s),\mathbf{X}_s)$ be the running cost of our system of Markov chains in the joint state $\mathbf{X}_s$ at time $s \in [0,T]$, where $T < \infty$. Then we have the general performance criterion

$$\min_{p(\cdot),u_{1,2}(\cdot),u_{2,1}(\cdot)} J[p(\cdot),u_{1,2}(\cdot),u_{2,1}(\cdot)],$$

for

$$J[p(\cdot),u_{1,2}(\cdot),u_{2,1}(\cdot)] = \mathbb{E}\left[\phi_0(\mathbf{X}_T) + \int_0^T f_0(s,\mathbf{X}_s)ds\right],$$

where $\phi_0 \in \mathbb{R}^{(N+1)\times(N+1)}$, $\langle\cdot,\cdot\rangle$ is the standard inner product,

$$\phi_0(\mathbf{X}_T) = \langle\phi_0,\mathbf{X}_T\rangle$$

and

$$f_0(s,\mathbf{X}_s) = \langle f_0(s),\mathbf{X}_s\rangle.$$

It is readily verified that for two dams

$$\langle\phi_0,\mathbf{X}_T\rangle = \langle\phi_0,X_{1,T}\otimes X_{2,T}\rangle = X_{1,T}^*.\phi_0.X_{2,T},$$

where * designates the transpose operator. A similar representation is true for $f_0(s,\mathbf{X}_s)$. Now, if we specify $\mathbf{X}_s = e_i^1 \otimes e_j^2$ for some specific $i,j \in \{0,...,N\}$ then we can form a matrix of the running costs of the chain,

$$f_0^*(s) = (f_0(s,e_i^1\otimes e_j^2))_{i,j=0,...,N}.$$

Assumption 3.1: For each $e_i^1 \otimes e_j^2$, $i,j = 0,...,N$, the elements of $f_0^*(s)$ are bounded below and continuous on $[0,T]\times[p_{min},p_{max}]\times[0,1]$.

B. VALUE FUNCTION

The value function gives us the minimum cost of control to go from state $\mathbf{X}_t = \mathbf{x}$ starting at time $t \in [0,T]$ to a given terminal state, $\mathbf{X}_T$. It has the form

$$V(t,\mathbf{x}) = \inf_{p(\cdot),u_{1,2}(\cdot),u_{2,1}(\cdot)} J[p(\cdot),u_{1,2}(\cdot),u_{2,1}(\cdot)|\mathbf{X}_t = \mathbf{x}],$$

for

$$J[\cdot|\mathbf{X}_t = \mathbf{x}] = \mathbb{E}\left[\phi_0(\mathbf{X}_T) + \int_t^T f_0(s,\mathbf{X}_s)ds|\mathbf{X}_t = \mathbf{x}\right].$$

From assumption 3.1 we know that this infimum exists.

We now let $V(t,\mathbf{x}) = \langle\phi(t),\mathbf{x}\rangle = x_1^*.\phi(t).x_2$, where $\phi(t) = (\phi_{i,j}(t))_{i,j=0,...,N} \in \mathbb{R}^{(N+1)\times(N+1)}$ is a matrix of measurable functions to be solved for.

C. DYNAMIC PROGRAMMING

At this point one usually applies the traditional continuous-time dynamic programming equation to solve this problem but the dynamic programming equation requires us to use the generator of the system. This is the Kronecker product of the generator matrices of the two separate dams and is large and cumbersome to work with. Instead we use an equivalent equation,

$$\begin{aligned}\langle\phi'(t),\mathbf{x}\rangle \quad = \quad & -\min_{p\in\bar{P},\cdot}[x_1^*.[A_1^*(t,p(t),\cdot).\phi(t) \\ & +\phi(t).A_2(t,p(t),\cdot) + f_0(t)].x_2,\end{aligned} \qquad (2)$$

where we use unit vectors $e_i, i = 0,...N$ for x_1 and $f_j, j = 0,...N$ for x_2. By theorem 4.4 of [13], we can choose optimal Markovian controls that will satisfy this equation and the dynamic programming equation. This gives us a system of $(N+1)\times(N+1)$ ordinary differential equations to solve for the $\phi_{i,j}(t)$, which in turn give us the set of optimal Markovian controls.

D. PERFORMANCE CRITERIA

For this example with two dams we have four types of performance criteria. The first type seeks to minimise the difference between the customer's seasonal demand intensity and the optimal demand intensity:

$$J_1(t,p(t),\mathbf{X}_t) = \left(C_1(t) - \sum_{k=1}^{n} x_{1,k}(t)\right)^2,$$

and

$$J_2(t,p(t),\mathbf{X}_t) = \left(C_2(t) - \sum_{k=1}^{n} x_{2,k}(t)\right)^2.$$

The second type considers the difference squared of the natural inflows and transfers into each dam and the customer demand and evaporation in each dam:

$$J_3(t,u_{2,1}(t),\mathbf{X}_t)=\left(\lambda_1(t)+u_{2,1}(t)-\sum_{k=1}^{n}x_{1,k}(t)-\mu_1(t)\right)^2,$$

and

$$J_4(t,u_{1,2}(t),\mathbf{X}_t)=\left(\lambda_2(t)+u_{1,2}(t)-\sum_{k=1}^{n}x_{2,k}(t)-\mu_2(t)\right)^2.$$

The third type is to minimise the probability that on average either dam falls below some critical level $M<N$:

$$J_5(t,p(t),u_{1,2}(t),u_{2,1}(t),\mathbf{X}_t)=\sum_{l=1}^{2}\left(\mathbb{E}\left[\int_0^T\sum_{k=1}^{M}X_{l,k}(s)ds\right]\right).$$

The fourth type is to minimise the probability the either dam is below level M at the terminal time T:

$$J_6(t,p(t),u_{1,2}(t),u_{2,1}(t)\mathbf{X}_t)=\sum_{l=1}^{2}\left(\mathbb{E}\left[\sum_{k=1}^{M}X_{l,k}(T)\right]\right).$$

Here, for the J_5 and J_6 criteria, the expectation is taken under the probability measure induced by the set of optimal controls.

In the current example, the control resources are unconstrained so $f_0(s,\mathbf{X}_t)$ is simply the sum of the first four criteria. The fifth criteria is included as a running cost on each state of the chain below level M on the control period and the sixth as a cost on the terminal states below level M. In the constrained case one needs to find a weighting for each criteria using the Lagrangian approach [11], but this is not pursued here.

IV. COMPUTATIONAL ASPECTS

Given the system of differential equations (2), we must write efficient code to solve the system and obtain the optimal controls. In the first instance we wrote the code to solve the system in a serial fashion and the results of this have been included as a numerical example in [13]. With a serial processor, however, the time taken to solve the system and calculate the controls can be considerable for a two dam system with a large number of states in each. For example, it took approximately 45 minutes to solve for a two dam system with 20 states in each dam on the office desktop computer. It was desirable to reduce this substantially. Parallel computing was the obvious way forward because there were parts of the code that could clearly be parallelized. In this section we discuss the main points of how our code was parallelized.

The ideas presented here are taken from a tutorial on parallel computing given at Supercomputing 2010 (SC10) [16]. Since this is not intended as a technical exposition of parallel computing in general, we will limit discussion to the parallelization of our dam system code. The first step was to identify parts of the code that could be easily run in parallel. This was comprised of two different types of code in general. The first type was where code simply constructed a definition. For example, the following code defines the $J_1(t,p(t),\mathbf{X_t})$ performance criteria. It seeks to minimise the difference between the demand for water in dam one, the sum of the $x_k(t)$, and the optimally supplied water, taking into account the level of both dams, $C_1(i,j)(t)$. The Mathematica code for the serial definition is

```
J_1=Table[(c_1[i][j][t]-Sum[x[k][t],
    {k,1,nsector[1]}])^2,{i,1,L},{j,1,L}].
```

The key point to note is that since we are simply defining this criterion and no calculation is taking place, we can safely define this criterion for each joint state in parallel. The following code is for the same definition but parallelized:

```
J_1=WaitAll[Table[ParallelSubmit[{i,j},
    (c_1[i][j][t]-Sum[x[k][t],
    {k,1,nsector[1]}])^2],{i,1,L},{j,1,L}]].
```

The command `ParallelSubmit` submits each element of the definition to the next available Mathematica kernel and `WaitAll` ensures that no code after this command is executed until all the definitions have been made. This is not so important here but becomes important when the value of a definition is changed as a result of this calculation. If code is executed before all new values are assigned, errors result. So this command is for safety and reliability of execution.

The second type of code that could be parallelized involved calculations which included variables that would not be evaluated until a later point. For example, in our problem we must minimise over each control individually. The solution is a function that involves some combinations of $\Phi(i,j)(t)$, which are the running costs of the joint states. However, these $\Phi(i,j)(t)$ are to be solved for as the solution to a system of ODE's at a later time and so we can do these minimisations in parallel without affecting the values of $\Phi(i,j)(t)$. The following code finds the minimising function for consumption in the first dam, $c_1(i,j)(t)$, in serial:

```
S_1=Table[s_1=Solve[D[EQ[[i,j]],
    c_1[i][j][t]]==0,c_1[i][j][t]];
    c_1[i][j][t]/.s_1,{i,1,L},{j,1,L}].
```

Compare this with the parallelized code:

```
S_1=WaitAll[Table[ParallelSubmit[{i,j},
    s_1=Solve[D[EQ[[i,j]],c_1[i][j][t]]==0,
    c_1[i][j][t]];c_1[i][j][t]/.s_1],
    {i,1,L},{j,1,L}]]
```

You will notice that the commands are the same and used in the same way as for simple parallel definitions. Approximately 27% of the operations (or individual execution lines of Mathematica code) were able to be altered to take advantage of parallel computing. If we use Amdahl's law (see [16]) to obtain an estimate for what kind of speed-up we could expect with $P=0.27$ and $N=32$ cores, then

$$\frac{1}{(1-P)+\frac{P}{N}}=1.35.$$

The next section shows that our results are much better than predicted. This is probably due to the optimised load bearing and parallelizing algorithms used by Mathematica.

The last element which must be considered is the distribution of variable and function definitions to all available processors, in our case Mathematica kernels. This is accomplished by executing the command `DistributeDefinitions[...,...,...]`, where the arguments are all of the variable and function definitions which must be available to each kernel in the subsequent calculation, separated by commas. At least in Mathematica, the process of parallelizing code is relatively simple, although experimentation is required. One aspect that needs to be considered is the computational overhead of parallelization. Every call to a different kernel and transfer of data between kernels takes time and some parts of the code may not be worth parallelizing if they already execute very quickly. In our example there were a number of very simple definitions that could be parallelized but the resulting performance was either a very minimal increase in speed or it was slightly slower. We found that it pays to focus on the areas of code that seem to take the most time when executing in serial.

All of this can be accomplished in various other programming languages, but the ease of implementation may differ considerably. We have used Mathematica 7.0 due to familiarity with this package and its relative ease of use. Implementing this in C, for example, would be more difficult but the execution would very likely be faster. For the purpose of preliminary experimentation Mathematica has been sufficient, however, we may need to write our problem in a lower level language like C at a later point depending on the size of the dam system we want to model.

V. PARALLEL COMPUTING RESULTS

The results of this section were obtained on a desktop computer with an Intel®Core™2 Duo CPU E8600 3.33Ghz processor and 3.49GB of RAM. The operating system was Microsoft Windows XP Professional version 2002, Service Pack 3 and the numerical software was Wolfram Mathematica version 7.0.0, Microsoft Windows 32-bit. Having a dual-core processor results in Mathematica being able to run two computational kernels in parallel. This was an ideal environment to perform some numerical experiments with parallel computing to see what the performance gains are like by parallelizing sections of appropriate code. After this, the code was run on the Monash Sun Grid (MSG), a high performance computer.

The basic idea was to see the difference in CPU seconds used between the Mathematica code executed in serial and in parallel. The code used solved for the optimal controls in a two dam system, found the probability of the dam system being in a particular state at any time $t \in [0,1]$ and compared the original demand with an average optimal consumption weighted by these probabilities. From these results one can obtain all of the pertinent performance characteristics of the optimal system. This code was executed in series and parallel three times each for dams with $3,4,...,9,10$ states in each, and the mean time calculated. The results of this are shown in Table I.

TABLE I
MEAN CPU SECONDS FOR SERIAL AND PARALLEL EXECUTION OF MATHEMATICA CODE.

States per dam	Serial	Parallel	Speed-up
3	1.74	1.97	0.88
4	3.47	3.46	1.00
5	6.97	5.93	1.18
6	12.34	8.83	1.40
7	20.08	11.00	1.83
8	33.85	15.29	2.21
9	54.05	19.13	2.83
10	84.36	23.92	3.53

From Table I we can see that when the dams have three levels each, the performance is marginally worse with the parallel code since there is a lot of computational overhead for a very small number of equations to solve. However, it is clear from the column showing 'Speed-up' that there is a rapid increase in performance with the increase in states. For interest we also ran the two versions of code once at $15,20$ and 25 levels. For 15 levels the approximate speed-up was 8.69 times, for 20 levels it was approximately 16.24 times, and at 25 levels Mathematica ran out of memory.

For a very small amount of change in the original serial code there is clearly a significant gain in execution speed. As stated, we have also executed these two versions of our code on the head node of the MSG, the university's grid computer. These results for one run are for dams with 10, 20,...,50 levels in each and shown in Table II, although serial calculations were aborted once the calculation time exceeded that of the largest parallel calculation. This node consists of a Sun X4600 chassis with 8 Opteron quad-core CPUs for a total of 32 cores. Each core has 2GB of RAM [17]. These calculations were carried out on one CPU (4 cores with 8GB RAM). You will note that for 10 levels the parallel calculations on the desktop desktop are faster. This is apparently due to hardware differences, especially processor speed. While not shown, the calculations for 20 levels were also approximately twice as fast on the desktop, so for a number of levels that a desktop can handle, desktop performance is comparatively good.

Of course, the MSG can handle much larger systems. In this application we want to control water consumption in a two dam system with an annual storage fluctuation of up to 20%. Since dam levels are usually quoted in 0.1% increments, this would require up to 200 levels per dam (a system of 40000 ODEs), an extremely demanding task computationally. Probably 0.5% changes in level are fine

TABLE II

CPU SECONDS FOR SERIAL AND PARALLEL EXECUTION OF MATHEMATICA CODE ON THE MSG.

States per dam	Serial	Parallel	Speed-up
10	308.19	48.79	6.32
20	4639.55	316.76	14.65
30	>21636.5	1882.44	-
40	-	6235.81	-
50	-	21636.5	-

enough for our purposes and this would give up to 40 levels in each dam (a system of 1600 ODEs), which took under two hours for a single CPU on the MSG.

In general, if we have $N+1$ states in each dam, then for a two dam system the maximum number of cores that could be used in execution of parallel code would be $(N+1)^2$, one for each joint state. This is unlikely to be optimal considering the overhead involved but we don't have the computing capacity to test what number would be optimal for a reasonable number of states in each dam (say 20 or more). This is an interesting question that we will have to consider in future work.

VI. NUMERICAL RESULTS

Figures 1 to 3 give the optimal price controls found using our model as a function of the fixed times $t = 0, 0.25, 0.5, 0.75, 1$ and as a percentage of each dams capacity. In general you can observe that the prices are reasonable in the sense that they are higher when the dams are at low capacity and low when at higher capacity. There is a bias toward higher prices based on the capacity of dam two. In our model there is a higher demand in dam one and so there is a tendency to try and take water from dam two, hence the uniformly high prices when the capacity of dam one is low. Figures 4 and 5 give selected flows between the dams when one of them is about 2/3 full. A positive flow is from dam one to dam two and a negative flow from dam two to dam one. There is a clear bias of transfers to dam one regardless of the state of dam two. This further supports the conclusion about higher demand in dam one. Further analysis of the numerical solutions of this model can be found in [13].

VII. FURTHER REMARKS ON NUMERICAL COMPLEXITY

In this model we have considered a two dam system where the jumps in levels that occur due to inflows are unit jumps. This gives rise to some nice structure in the system of ODE's in that each joint state depends not on all the joint states but only those immediately neighbouring. This makes their solution far more tractable, however, this assumption of unit jumps is not very realistic since in reality a dam could overflow given a sufficiently large rain event. This is being taken into account in a more advanced model which we are currently working on.

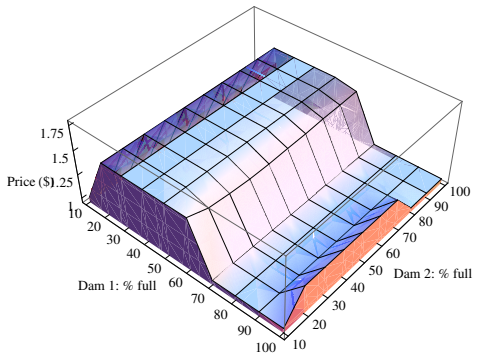

Fig. 1. Prices at t=0.

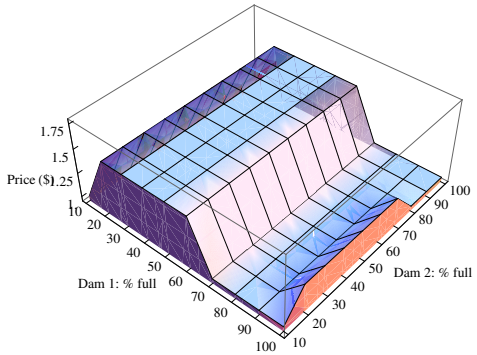

Fig. 2. Prices at t=0.5.

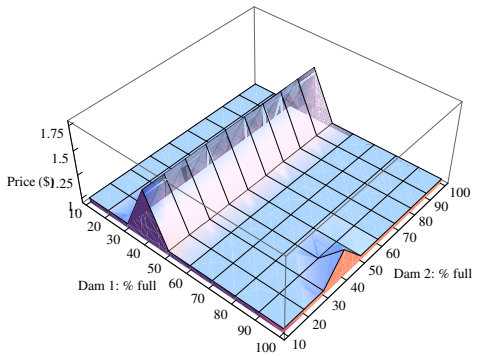

Fig. 3. Prices at t=1.

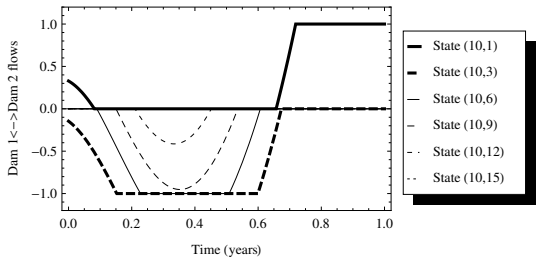

Fig. 4. Selected dam flows with dam 1 at about 2/3 full.

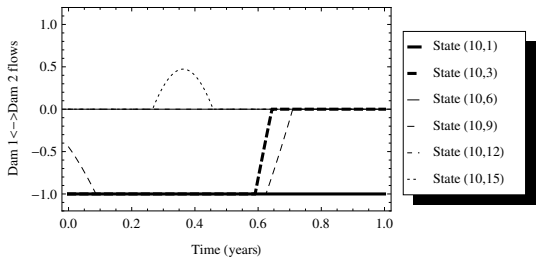

Fig. 5. Selected dam flows with dam 2 at about 2/3 full.

We need also consider the possibility of more dams in the system. In [13] we detail how this model can be extended to any number of dams connected in quite an arbitrary way. Essentially we look at the joint states of the dam system as a tensor of the depth of the number of dams and then consider the derivative of the generalised inner product of this tensor with the tensor $\phi(t)$ of the same depth (for details please see [13]). This results in a system of $(N+1)^d$ ODE's where d is the number of linked dams in the system.

Both of these aforementioned changes will make the numerical solution of this problem more difficult. Both will increase the calculation time required for the numerical solution of the ODE system and will also increase the time taken to define the equations and carry out intermediate calculations. We have not done this work yet but parallelization and HPC will be the focus of our efforts.

VIII. CONCLUSION

In this paper we have explained a model for the optimal control of a system of two dams through the agency of state and time dependent price and water transfer controls. The innovation of this model is to take into account the seasonal demand intensity for each customer of each dam when defining our optimal price control. This allows us to manage the system taking into account traditional criteria of optimal performance as well as attempting to satisfy the dam users in some optimal way. We have also explained the use of parallel computing in our work as a means for reducing calculation time and making larger systems more tractable. Parallel computing presents real opportunities to attempt to solve problems with a larger number of states in each dam and more dams. Our future work will focus on solving systems which include a more random inflow, impulsive controls under control resource constraints, the possibility of overflow and the numerical procedures necessary to make the solutions of these problems tractable.

REFERENCES

[1] E. Altman, *Constrained Markov Decision Processes*. Boca Raton, FL: Chapman & Hall/CRC, 1999.

[2] D. Bertsekas and S. Shreve, *Stochastic Optimal Control: The Discrete-Time Case*. Belmont, MA: Athena Scientific, 1996.

[3] M. Kitaev and V. Rykov, *Controlled Queueing Systems*. Boca Raton, FL: CRC, 1995.

[4] B. Miller, G. Miller, and K. Siemenikhin, "Control of markov chains with constraints," in *Proceedings of the VIII International Conference "System Identification and Control Problems" SICPRO '09*, Moscow, 26-30 January 2009.

[5] M. Faddy, "Optimal control of finite dams: Discrete (2-stage) output procedure," *Journal of Applied Probability*, vol. 11, no. 1, pp. 111–121, 1974.

[6] L. Yeh and L. Hua, "Optimal control of a finite dam: Wiener process input," *Journal of Applied Probability*, vol. 24, no. 1, pp. 186–199, 1987.

[7] M. Abdel-Hameed, "Optimal control of a dam using $p^M_{\lambda,\tau}$ policies and penalty cost when the input process is a compound poisson process with positive drift," *Journal of Applied Probability*, vol. 37, no. 2, pp. 408–416, 2000.

[8] J. Bae, S. Kim, and E. Lee, "Average cost under the $p^M_{\lambda,\tau}$ policy in a finite dam with compound poisson inputs," *Journal of Applied Probability*, vol. 40, no. 2, pp. 519–526, 2003.

[9] V. Abramov, "Optimal control of a large dam," *Journal of Applied Probability*, vol. 44, no. 1, pp. 249–258, 2007.

[10] B. Miller, "Optimization of queuing system via stochastic control," *Automatica*, vol. 45, pp. 1423–1430, 2009.

[11] B. Miller, G. Miller, and K. Siemenikhin, "Torwards the optimal control of markov chains with constraints," *Automatica*, vol. 46, pp. 1495–1502, 2010.

[12] A. Miller and B. Miller, "Control of connected markov chains. application to congestion avoidance in the internet," accepted to IEEE-CDC 2011.

[13] B. Miller and D. McInnes, "Management of dam systems via optimal price control," Procedia Computer Science, 2011, to be published as conference proceedings of ICCS 2011, June 1-3, 2011.

[14] L. Aggoun, R. Elliot, and J. Moore, *Hidden Markov Models: Estimation and Control*. New York: Springer-Verlag, 1995.

[15] B. Miller and D. McInnes, "Management of a large dam via optimal price control," IFAC-PapersOnLine, 2011, to be published as coference proceeding of IFAC 2011, August 28 -September 2, 2011.

[16] Q. Stout and C. Jablonowski, "Parallel computing 101," 14 November 2010, tutorial notes from Supercomputing 2010 (SC10) - electronic copy given to participants.

[17] (2010, September) Monash sun grid overview. Monash University e-Research Centre. [Online]. Available: https://confluence-vre.its.monash.edu.au/display/mcgwiki/Monash+Sun+Grid+Overview