Parallel Stochastic Dynamic Programming: Finite Element Methods

S.L. Chung, F.B. Hanson and H.H. Xu

Laboratory for Advanced Computing

Department of Mathematics, Statistics and Computer Science

University of Illinois at Chicago

P.O. Box 4348; M/C 249

Chicago, IL 60680

Internet: hanson@math.uic.edu

Running Title: Parallel Stochastic Dynamic Programming

ABSTRACT- A finite element method for stochastic dynamic programming is developed. The computational method is valid for a general class of optimal control problems that are nonlinear and perturbed by general Markov noise in continuous time, including jump Poisson noise. Stability and convergence of the method are verified and its storage utilization efficiency over the traditional finite difference method is demonstrated. This advanced numerical technique, together with parallel computation, helps to alleviate Bellman's *curse of dimensionality* by permitting the solution of larger problems.

1. INTRODUCTION

Stochastic optimal control is important in many areas such as aerospace dynamics, financial economics, resource management, robotics, medicine and power generation. The main target of this research is to develop a general computational treatment of stochastic control in continuous time. Dynamic Programming introduced by Bellman is a powerful tool to attack stochastic optimal control and problems of similar natures [1, 3]. A highly optimized computational method using the finite difference scheme for stochastic dynamic programming in small state space of moderate dimension is presented in many previous works [11, 12, 18, 4, 5]. However, dynamic programming suffers from Bellman's *Curse of Dimensionality* in which the computational demands grow exponentially with the state space dimension. The exponential growth in both computing time and memory requirements hinder the solution of large problems. Even with the most sophisticated supercomputers, the maximum number of states that can be handled, with reasonable degree of accuracy, is restricted to *four* [5, 6]. Larson [13] proposed a discretization of dynamic programming called *Incremental Dynamic Programming* which helps to save computer memory requirement for large problem. Although the method is effective, it can only be applied to deterministic optimal control problems in discrete time.

The main objective of this paper is to apply the finite element method to stochastic dynamic programming. The finite element method possesses a high degree of accuracy over the traditional finite difference scheme in effect that memory requirements are cut down significantly in this memory-bound problem, trading some computational efficiency for reduced memory requirements. Hence, Bellman's curse of dimensionality is alleviated with regard top memory requirements. In this paper, Section 2 gives a general review of the mathematical background for stochastic dynamic programming. Section 3 gives the finite element formulation. Section 4 gives the discretization in backward time, the Crank-Nicolson predictor-corrector scheme. Convergence and stability for the method is verified in Section 5. In Section 6, the computational efficiency of the finite element method is compared with the finite difference method.

2. FORMULATION OF BELLMAN'S FUNCTIONAL PDE

The mathematical foundations for stochastic differential equations are given by Gihman and Skorohod [9]. There are also many other treatments restricted to the continuous Gaussian noise case [2, 7, 16].

The Markov, multibody dynamical system is illustrated in Figure 2.1 and is governed by the stochastic differential equation (SDE):

$$d\mathbf{X}(T) = \mathbf{F}(\mathbf{X},T,\mathbf{U})dT + G(\mathbf{X},T)d\mathbf{W}(T) + H(\mathbf{X},T)d\mathbf{P}(T),$$
$$\mathbf{X}(t) = \mathbf{x}; 0 < t < T < t_f; \mathbf{X} \in \mathcal{D}_x; \mathbf{U} \in \mathcal{D}_u, \tag{2.1}$$

where $\mathbf{X}(T)$ is the $m\times 1$ state vector and the feedback control variable, $\mathbf{U}(\mathbf{X}(T),T)$, is an $n\times 1$ vector in the control space $\mathcal{D}_u$. $\mathbf{W}$ is the r-dimensional Gaussian white noise vector which represents the continuous, background component of the perturbations, such as that due to fluctuating population death rates in a biological resource environment, randomly varying winds and other background environmental noise. $\mathbf{P}$ is the q-dimensional Poisson white noise vector which models discontinuous rare event components, such as occasional mass mortality, large random weather changes or other large environmental effects. It is assumed that $\mathbf{W}$ and $\mathbf{P}$ are pairwise independent by components, i.e., their covariance is diagonal. $\mathbf{F}$ is an $m \times 1$ deterministic nonlinearity vector, G is an $m \times r$ diffusion coefficient array and H is an $m \times q$ Poisson amplitude coefficient array. (In general, H can be random and distributed, but H is treated as nonrandom here for simplicity.) The time varible t represents a family of initial times that helps to facilitate the dynamic programming analysis

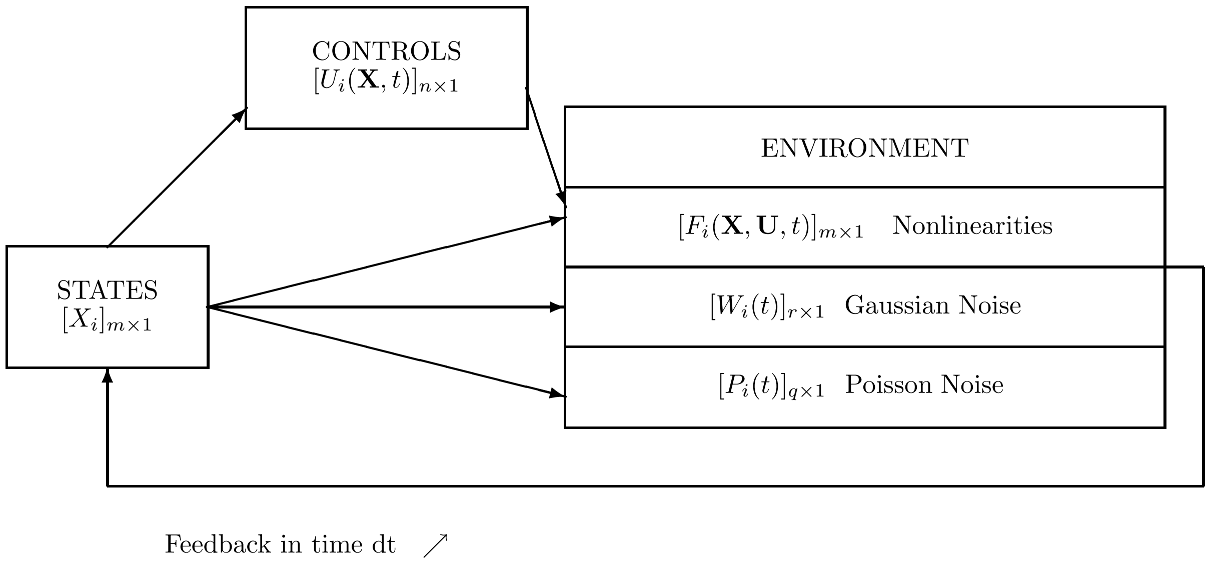
CONTROLS
$[U_i(\mathbf{X},t)]_{n\times 1}$
ENVIRONMENT
$[F_i(\mathbf{X},\mathbf{U},t)]_{m\times 1}$ Nonlinearities
$[W_i(t)]_{r\times 1}$ Gaussian Noise
$[P_i(t)]_{q\times 1}$ Poisson Noise
STATES
$[X_i]_{m\times 1}$
Feedback in time dt

Figure 2.1 Multibody Dynamical System

and becomes the fundamental backward time variable of the equation of dynamic programming. Typically, the target initial time is zero, but is not appropriate for variation.

The fairly general objective of the problem is to optimize the expected value of the performance criterion or total costs

$$V(\mathbf{X},\mathbf{U},\mathbf{P},\mathbf{W},t) = \int_t^{t_f} dT\ C(\mathbf{X}(T),T,\mathbf{U}(\mathbf{X}(T),T)) + Z(\mathbf{X}(t_f)), \tag{2.2}$$

where $C(\mathbf{x},t,\mathbf{u})$ is the instantaneous cost function and Z is the terminal cost function. The optimal expected performance on the time horizon (t,t_f) is given by

$$V^*(\mathbf{x},t) = \min_{\mathbf{u}}[\underset{\{\mathbf{P},\mathbf{W}\}}{\mathrm{MEAN}}[V(\mathbf{X},\mathbf{U},\mathbf{P},\mathbf{W},t)|\mathbf{X}(t)=\mathbf{x},\mathbf{U}(\mathbf{X}(t),t)=\mathbf{u}], \tag{2.3}$$

such as to minimize costs of production, costs of extraction, fuel consumption, or lateral/longitudinal perturbation of motion. Instead of a difficult direct path search over all the state and control spaces, a dynamic programming formulation is used. The minimization in (2.3) is reduced to the simpler minimization of a switching term over the control. The reduction of the minimization is accomplished by Bellman's *Principle of Optimization*, followed by generalized *Itô* Chain Rule (see [10], for instance), leading to the *Bellman functional PDE* of dynamic programming

$$\begin{aligned} 0 &= \frac{\partial V^*}{\partial t} + \frac{1}{2}\left(GG^T\right)(\mathbf{x},t) : \nabla_x {\nabla_x}^T V^*(\mathbf{x},t) \\ &+ \sum_{l=1}^{q} \lambda_l \cdot [V^*(\mathbf{x}+\mathbf{H}_l(\mathbf{x},t),t) - V^*(\mathbf{x},t)] + S(x,t), \end{aligned} \tag{2.4}$$

where the minimized functional control switching term is

$$\begin{aligned} S(x,t) &= \min_{\mathbf{u}}[\mathcal{S}(\mathbf{x},t,\mathbf{u})] \\ &= \min_{\mathbf{u}}[C(\mathbf{x},t,\mathbf{u}) + \mathbf{F}^T(\mathbf{x},t,\mathbf{u})\nabla_x V^*(\mathbf{x},t)], \end{aligned} \tag{2.5}$$

if it is assumed that only the scalar cost function C and the $m \times 1$ nonlinearity function $\mathbf{F}$ are control dependent. In (2.4), the scalar matrix product

$$A : B = \sum_i \sum_j A_{ij} B_{ij} = Trace[AB^T].$$

In general, Eq. (2.4) will be nonlinear when the argument of the minimum in (2.5) is nonlinear in the control vector $\mathbf{U}$. Also note that the discrete Poisson noise leads to a PDE with a non-local delay functional term through the $m \times 1$ argument $\mathbf{x}+\mathbf{H}_l$, whose components are $x_i + H_{il}$. The use of general Markov noise does not lead to simple boundary conditions for (2.4-2.5), but fortunately the boundary conditions are embedded inthe Bellman equation, unlike the forward Kolmogorov equation.

To simplify the solution of the Bellman functional PDE (2.4), the costs and dynamics are assumed to be functions of the control so that formal solution of the minimization in (2.5) is permitted, while still retaining some generality. Costs that are a quadratic function of the control are a reasonable choice, taking the minimum energy form

$$C(\mathbf{x},t,\mathbf{u}) = C_0(\mathbf{x},t) + \mathbf{C}_1^T(\mathbf{x},t)\mathbf{u} + \frac{1}{2}\mathbf{u}^T C_2(\mathbf{x},t)\mathbf{u}, \tag{2.6}$$

so that the unit cost of the controls becomes more expensive at large values, provided C_2 is positive definite. In (2.6), C_0 is a scalar function as is C, $\mathbf{C}_1$ is an $n \times 1$ vector, and C_2 is an $n \times n$ array.

In addition, the dynamics in (2.1) are assumed to be linear in the controls,

$$\mathbf{F}(\mathbf{x},t,\mathbf{u}) = \mathbf{F}_0(\mathbf{x},t) + F_1(\mathbf{x},t)\mathbf{u}, \tag{2.7}$$

while remaining nonlinear in the state vector. In (2.7), $\mathbf{F}_0$ is an $m \times 1$ nonlinearity vector, and F_1 is an $m \times n$ array. The linear-quadratic control form does permit the formal solution of the minimization problem in (2.5). The regular or unconstrained control, $\mathbf{U}_R$, is determined from the critical points of the argument of the minimum in (2.5), that is

$$0 = \nabla_u \mathcal{S} = \mathbf{C}_1 + F_1^T \nabla_x V^* + \frac{1}{2}(C_2 + C_2^T)\mathbf{u}. \tag{2.8}$$

Assuming that the $n \times n$ coefficient C_2 in (2.8) is nonsingular and symmetric, the explicit form of the regular control is obtained as

$$\mathbf{U}_R(\mathbf{x},t) = -C_2^{-1} \cdot (\mathbf{C}_1 + F_1^T \nabla_x V^*). \tag{2.9}$$

For costs more general than quadratic, Eq.(2.9) would be an initial approximation. The optimal control for the constrained problem will be a regular control only when the regular control is in the control domain $\mathcal{D}_u$. In the case of a hypercube constraints, $U_{min,i} \leq U_i(\mathbf{x},t) \leq U_{max,i}$ for $i = 1$ to n, the optimal control $\mathbf{U}^*$ will take the form

$$U_i^*(\mathbf{x},t) = \begin{cases} U_{R,i}, & U_{min,i} \leq U_{R,i} \leq U_{max,i} \\ U_{min,i}, & U_{R,i} \leq U_{min,i} \\ U_{max,i}, & U_{R,i} \geq U_{max,i} \end{cases} \tag{2.10}$$

or

$$U_i^*(\mathbf{x},t) = min[U_{max,i}, max[U_{min,i}, U_{R,i}(\mathbf{x},t)]], \tag{2.11}$$

where U_i^* is the i^{th} component of the optimal control vector $\mathbf{U}^*$; $U_{max,i}$ and $U_{min,i}$ are respectively the maximum and minimum of the i^{th} component in the hypercube control domain $\mathcal{D}_u$.

When $\mathbf{U}_R$ exceeds the constraints of $\mathcal{D}_u$, $\mathbf{U}^*$ is called a bang control denoting banging on the constraints. Applying the optimal control calculation to the minimized functional term and eliminating some terms in favor of the regular control vector in (2.9)

$$S(\mathbf{x},t) = C_0 + \mathbf{F}_0^T \nabla_x V^* + \frac{1}{2}(\mathbf{U}^* - 2\mathbf{U}_R)^T C_2 \mathbf{U}^* \tag{2.12}$$

is obtained. Since $\mathbf{U}_R$ depends linearly on the solution gradient, $\nabla_x V^*$, $\mathbf{U}^*$ is the constrained counterpart of $\mathbf{U}_R$ in (2.9), and the minimized functional (2.12) is quadratic in the controls, the minimized functional (2.12), in general, makes the Bellman equation (2.4) a nonlinear partial differential equation.

3. FORMULATION OF FINITE ELEMENT (GALERKIN) METHOD

In order to solve the Bellman functional PDE (2.4) with (2.12) numerically, a Galerkin approximation in the state-space is used:

$$V^*(\mathbf{x},t) \approx \sum_{j=1}^{N} \widehat{V}_j(t) \cdot \phi_j(\mathbf{x}), \tag{3.1}$$

where $\Phi(\mathbf{x}) = [\phi_i]_{N\times 1}$ denotes a set of N basis functions that are linearly independent, and piecewise smooth. No assumptions are made about essential boundary conditions here.

The Bellman functional PDE is multiplied with a test function ψ and integrated throughout the domain $\mathcal{D}_x$, which forms the variational (or weak) formulation:

$$\begin{aligned} 0 &= \int_{\mathcal{D}_x} \psi \cdot \Big[\frac{\partial V^*}{\partial t} + \mathbf{F}_0^T \nabla_x V^*(\mathbf{x},t) + \tfrac{1}{2} GG^T(\mathbf{x},t) : \nabla_x {\nabla_x}^T V^* \\ &+ \sum_{l=1}^{q} \lambda_l \cdot [V^*(\mathbf{x} + \mathbf{H}_l(\mathbf{x},t),t) - V^*(\mathbf{x},t)] \\ &+ (\tfrac{1}{2}\mathbf{U}^* - \mathbf{U}_R)^T C_2 \mathbf{U}^* \Big] \, \mathbf{dx}. \end{aligned} \tag{3.2}$$

The finite element discretization is obtained by substituting the Galerkin approximation (3.1) for $V^*(\mathbf{x},t)$ and the N basis functions successively for the test function ψ.

The term $\frac{1}{2}\int \phi_i GG^T(\mathbf{x},t) : \nabla_x {\nabla_x}^T V^*(\mathbf{x},t) d\mathbf{x}$ involves second order derivatives in $\mathbf{x}$. One order of derivative in $V^*(\mathbf{x},t)$ is transferred to ϕ_i, so that weaker continuity requirements will be sufficient for the approximating functions $\phi_i(\mathbf{x})$. The transformation is done through *the product rule* and *divergence theorem* (integration by parts):

$$\tfrac{1}{2}\int_{\mathcal{D}_x} \phi_i GG^T(\mathbf{x},t) : \nabla_x {\nabla_x}^T V^*(\mathbf{x},t) d\mathbf{x}$$

$$\begin{aligned}
&= \tfrac{1}{2}\int_{\mathcal{D}_x} \phi_i \sum_k \sum_l \sum_m G_{kl} G_{ml} D_k D_m V^*(\mathbf{x},t) d\mathbf{x} \\
&= \tfrac{1}{2}\sum_k \sum_l \sum_m \int_{\mathcal{D}_x} D_k(\phi_i G_{kl} G_{ml} D_m V^*(\mathbf{x},t)) d\mathbf{x} \\
&\quad -\tfrac{1}{2}\sum_k \sum_l \sum_m \int_{\mathcal{D}_x} D_k(\phi_i G_{kl} G_{ml}) D_m V^*(\mathbf{x},t) d\mathbf{x} \\
&= \tfrac{1}{2}\int_{\mathcal{D}_x} \nabla_x \cdot (\phi_i G G^T \nabla_x V^*(\mathbf{x},t)) d\mathbf{x} \\
&\quad -\tfrac{1}{2}\int_{\mathcal{D}_x} ({\nabla_x}^T(\phi_i G G^T)) \cdot \nabla_x V^*(\mathbf{x},t) d\mathbf{x} \qquad (3.3)\\
&= \tfrac{1}{2}\oint_{\partial\mathcal{D}_x} \widehat{\mathbf{n}}^T \left[\phi_i G G^T \sum_{j=1}^{N} \widehat{V}_j(t) \nabla_x \phi_j\right] ds \\
&\quad -\tfrac{1}{2}\int_{\mathcal{D}_x} ({\nabla_x}^T(\phi_i G G^T)) \cdot \sum_{j=1}^{N} \widehat{V}_j(t) \nabla_x \phi_j \mathbf{dx}
\end{aligned}$$

where

$$G_{ij} = \text{the element of G in the } i^{th} \text{ row and the } j^{th} \text{ column},$$

$$D_i = \text{the derivative with respect to the } i^{th} \text{ component of } \mathbf{x},$$

and

$$\widehat{\mathbf{n}} = \text{unit normal vector to the boundary } \partial\mathcal{D}_x.$$

Using (3.3) together with the Galerkin approximation (3.1), with $\psi = \phi_i$, leads to the form

$$\begin{aligned} 0 &= \int_{\mathcal{D}_x} \phi_i \sum_{j=1}^{N} \frac{d\widehat{V}_j(t)}{dt} \phi_j \mathbf{dx} \\ &+ \int_{\mathcal{D}_x} \phi_i \mathbf{F}_0^T \sum_{j=1}^{N} \widehat{V}_j(t) \nabla_x \phi_j \mathbf{dx} \\ &- \tfrac{1}{2} \int_{\mathcal{D}_x} (\nabla_x{}^T(\phi_i G G^T)) \sum_{j=1}^{N} \widehat{V}_j(t) \nabla_x \phi_j \mathbf{dx} \\ &+ \tfrac{1}{2} \oint_{\partial\mathcal{D}_x} \widehat{\mathbf{n}}^T \left[(\phi_i G G^T) \sum_{j=1}^{N} \widehat{V}_j(t) \nabla_x \phi_j \right] ds \\ &+ \int_{\mathcal{D}_x} \phi_i \sum_{j=1}^{N} \sum_{l=1}^{q} \lambda_l \widehat{V}_j(t) [\phi_j(\mathbf{x} + \mathbf{H}_l(\mathbf{x}, t)) - \phi_j(\mathbf{x})] \mathbf{dx} \\ &+ \int_{\mathcal{D}_x} \phi_i \cdot (\tfrac{1}{2}\mathbf{U}^* - \mathbf{U}_R)^T C_2 \mathbf{U}^* \mathbf{dx}, \end{aligned} \tag{3.4}$$

for $i = 1$ to N.

Inner products are defined as

$$(\psi, \varphi) = \int_{\mathcal{D}_x} \psi(\mathbf{x}, \cdot) \cdot \varphi(\mathbf{x}, \cdot) d\mathbf{x}, \tag{3.5}$$

and

$$(\psi, \varphi)_\partial = \oint_{\partial\mathcal{D}_x} \psi(\mathbf{x}, \cdot) \cdot \varphi(\mathbf{x}, \cdot) ds, \tag{3.6}$$

for some bounded state domain $\mathcal{D}_x$ and its boundary $\partial\mathcal{D}_x$. In addition, the following simplifying matrix notations are defined:

$$\widehat{\mathbf{V}}(t) = [\widehat{V}_i(t)]_{N\times 1},$$

$$\phi_{j,l} = \phi_{j,l}(\mathbf{x}, t) \equiv \phi_j(\mathbf{x} + \mathbf{H}_l(\mathbf{x}, t)),$$

$$\mathbf{S}(t) = [(\phi_i, (\tfrac{1}{2}\mathbf{U}^* - \mathbf{U}_R)^T C_2 \mathbf{U}^*)]_{N\times 1} \tag{3.7}$$

and

$$\mathbf{Q}(t) = \tfrac{1}{2}[(\phi_i, \widehat{\mathbf{n}}^T G G^T \nabla_x \phi_j)_\partial]_{N\times N} \widehat{\mathbf{V}}(t).$$

These nner product and matrix notations convert this intermediate form into the nonlinear algebraic system

$$\begin{aligned} 0 &= [(\phi_i, \phi_j)]_{N\times N} \frac{d\widehat{\mathbf{V}}(t)}{dt} \\ &+ [(\phi_i, \mathbf{F}_0^T \nabla_x \phi_j)]_{N\times N} \widehat{\mathbf{V}}(t) \\ &- \tfrac{1}{2}[(\nabla_x{}^T(\phi_i G G^T), \nabla_x \phi_j)]_{N\times N} \widehat{\mathbf{V}}(t) \\ &+ \sum_{l=1}^{q} \lambda_l \cdot [(\phi_i, (\phi_{j,l} - \phi_j))]_{N\times N} \widehat{\mathbf{V}}(t) \\ &+ \mathbf{S}(t) + \mathbf{Q}(t) \;=\; 0. \end{aligned} \tag{3.8}$$

If the costs are taken to be quadratic in $\mathbf{U}$ as shown in Section 2, then the special Galerkin approximation to the regular control and optimal control, respectively, are

$$\mathbf{U}_R(\mathbf{x}, t) \approx -C_2^{-1} \cdot \left(\mathbf{C}_1 + \sum_{j=1}^{N} \widehat{V}_j(t) F_1^T \nabla_x \phi_j(\mathbf{x}) \right), \tag{3.9}$$

and

$$U_i^*(\mathbf{x}, t) = min[U_{max,i}, max[U_{min,i}, U_{R,i}(\mathbf{x}, t)]], \tag{3.10}$$

where U_i^* is the i^{th} component of the optimal control vector $\mathbf{U}^*$, $U_{max,i}$ and $U_{min,i}$ are, respectively, the maximum and minimum of the i^{th} component in a hypercube control domain $\mathcal{D}_u$. Note quadratic costs imply that the switch matrix $\mathbf{S}(t)$ will, in general, have cubic nonlinearity in the basis functions.

The boundary conditions vary heavily with application, depending on whether the boundary $\partial\mathcal{D}_x$ is absorbing, reflecting or combinations of these. In many cases, there are no simple boundary specifications, so that boundary values must be obtained by integrating the Bellman equation along the boundary. For the simplicity of the following derivation, we avoid the specification of the boundary conditions by assuming the coefficient $G(\mathbf{x}, t)$ vanishes at the boundary, so that $\mathbf{Q}(t) \equiv 0$.

4. CRANK-NICOLSON PREDICTOR-CORRECTOR SCHEME

As noted in the previous section, the finite variational form of the Bellman functional PDE has a cubic nonlinearity in the basis functions; consequently, the resulting nonlinear matrix ODE (3.8) cannot be solved numerically with a single step scheme. The time approximation used here is basically a Crank-Nicolson discretization scheme and function values for each time step are iterated to obtain a reasonable approximation of the nonlinear switching term $\mathbf{S}$. The backward time discretization is given as

$$T_k = t_f - (k-1)\cdot DT,$$

for $k = 1$ to K.

Using $\widetilde{\mathbf{V}}_k = \widehat{\mathbf{V}}(T_k)$, the $O(k^2)$ central finite difference of the vector time derivative $d\widehat{\mathbf{V}}(t)/dt$ at $T_{k+\frac{1}{2}}$ is given as

$$\frac{\left(\widetilde{\mathbf{V}}_{k+\frac{1}{2}+\frac{1}{2}} - \widetilde{\mathbf{V}}_{k+\frac{1}{2}-\frac{1}{2}}\right)}{-DT} = \frac{(\widetilde{\mathbf{V}}_{k+1} - \widetilde{\mathbf{V}}_k)}{-DT}.$$

Under this time discretization scheme, the functions $\mathbf{F}_0$ and G are evaluated at $T_{k+\frac{1}{2}}$, and are denoted by $\mathbf{F}_{0,k+\frac{1}{2}}$ and $G_{k+\frac{1}{2}}$, respectively. The linear approximation $\widetilde{\mathbf{V}}_{k+\frac{1}{2}} = \frac{1}{2}(\widetilde{\mathbf{V}}_{k+1} + \widetilde{\mathbf{V}}_k)$, is accurate to $O(k^2)$. The nonlinear term $\mathbf{S}_{k+\frac{1}{2}}$ must be successively approximated by a predictor-corrector method to obtain a reasonable degree of accuracy that is comparable to that of the linear algebraic term to avoid numerical pollution. After the time discretization, the nonlinear algebraic system equation (3.8) is converted to a Crank-Nicolson algebraic system

$$A_{k+\frac{1}{2}}(\widetilde{\mathbf{V}}_{k+1} - \widetilde{\mathbf{V}}_k) = DT \cdot (B_{k+\frac{1}{2}}\widetilde{\mathbf{V}}_k + \mathbf{S}_{k+\frac{1}{2}}) \tag{4.1}$$

where

$$\begin{aligned} A_{k+\frac{1}{2}} &= [(\phi_i, \phi_j)]_{N\times N} - \tfrac{1}{2}DT \cdot B_{k+\frac{1}{2}}, \\ B_{k+\frac{1}{2}} &= -\tfrac{1}{2}[(\nabla_x{}^T(\phi_i G_{k+\frac{1}{2}} G^T_{k+\frac{1}{2}}), \nabla_x \phi_j)]_{N\times N} \end{aligned}$$

$$+ [(\phi_i, \mathbf{F}^T_{0,k+\frac{1}{2}} \nabla_x \phi_j)]_{N\times N} \tag{4.2}$$
$$+ \sum_{l=1}^{q} \lambda_l \cdot [(\phi_i, (\phi_{j,l,k+\frac{1}{2}} - \phi_j))]_{N\times N},$$

and $\mathbf{S}_{k+\frac{1}{2}}$ is the corresponding nonlinear switch term. The difference $(\widetilde{\mathbf{V}}_{k+1} - \widetilde{\mathbf{V}}_k)$ is calculated instead of $\widetilde{\mathbf{V}}_{k+1}$ to reduce catastrophic cancellation in finite precision arithmetic in computers. Exact formulation of this $O(DT)$ difference in the solution loses far less precision than calculating the solution at the new time step alone when it will likely be the same order as the solution at the old time step.

The predictor-corrector method begins with a convergence accelerating *extrapolator (x)* start which helps to cut down the number of corrections for each time step. For the new $(k+1)^{st}$ time step, before the new value of $\widetilde{\mathbf{V}}_{k+1}$ is evaluated, the old values of $\widetilde{\mathbf{V}}_k$ and $\widetilde{\mathbf{V}}_{k-1}$ are assumed to be known, with the starting condition $\widetilde{\mathbf{V}}_0 = \widetilde{\mathbf{V}}_1$. The extrapolated value

$$\widetilde{\mathbf{V}}^{(x)}_{k+\frac{1}{2}} = \tfrac{1}{2}(3 \cdot \widetilde{\mathbf{V}}_k - \widetilde{\mathbf{V}}_{k-1}), \tag{4.3}$$

is used to calculate the values of

$$\mathbf{U}^{(x)}_{R,k+\frac{1}{2}}(\mathbf{x}) \approx -C_2^{-1} \cdot \left(\mathbf{C}_1 + \sum_{j=1}^{N} \widehat{V}^{(x)}_{j,k+\frac{1}{2}} F_1^T \nabla_x \phi_j(\mathbf{x}) \right), \tag{4.4}$$

and

$$U^{(x)}_{i,k+\frac{1}{2}}(\mathbf{x}) = min[U_{max,i}, max[U_{min,i}, U^{(x)}_{R,i,k+\frac{1}{2}}(\mathbf{x})]] \tag{4.5}$$

where $U^{(x)}_{i,k+\frac{1}{2}}$ is the i^{th} component of the extrapolated value of the optimal control vector $\mathbf{U}^{(x)}_{k+\frac{1}{2}}$ and $U^{(x)}_{R,i,k+\frac{1}{2}}$ is the i^{th} component of the extrapolated value of the regular control vector $\mathbf{U}^{(x)}_{R,k+\frac{1}{2}}$, all evaluated at mid-point time $T_{k+\frac{1}{2}}$. The extrapolated values of the controls are in turn used to update the value of the switch term

$$\mathbf{S}^{(x)}_{k+\frac{1}{2}} = [(\phi_i, (\tfrac{1}{2}\mathbf{U}^{(x)}_{k+\frac{1}{2}} - \mathbf{U}^{(x)}_{R,k+\frac{1}{2}})^T C_2 \mathbf{U}^{(x)}_{k+\frac{1}{2}})]_{N\times 1} \tag{4.6}$$

The extrapolated values are put into (4.1) to obtain the *extrapolated predictor (xp)* linear algebraic system:

$$A_{k+\frac{1}{2}}(\widetilde{\mathbf{V}}^{(xp)}_{k+1} - \widetilde{\mathbf{V}}_k) = DT \cdot (B_{k+\frac{1}{2}}\widetilde{\mathbf{V}}_k + \mathbf{S}^{(x)}_{k+\frac{1}{2}}). \tag{4.7}$$

The $(k+\frac{1}{2})^{st}$ time step values are then obtained using the extrapolated predictor values $\widetilde{\mathbf{V}}_{k+1}^{(xp)}$ in the *predictor evaluation (xpe)* step:

$$\widetilde{\mathbf{V}}_{k+\frac{1}{2}}^{(xpe)} = \tfrac{1}{2}(\widetilde{\mathbf{V}}_{k+1}^{(xp)} + \widetilde{\mathbf{V}}_k), \tag{4.8}$$

which is then used to update the optimal control $\mathbf{U}_{k+\frac{1}{2}}^{(xpe)}$ and the switch term $\mathbf{S}_{k+\frac{1}{2}}^{(xpe)}$. The updated values are then used in the *corrector* (*xpec*) steps which the $(\gamma+1)$st correction is obtained from the system

$$A_{k+\frac{1}{2}}(\widetilde{\mathbf{V}}_{k+1}^{(xpec,\gamma+1)} - \widetilde{\mathbf{V}}_k) = DT\cdot(B_{k+\frac{1}{2}}\cdot\widetilde{\mathbf{V}}_k + \mathbf{S}_{k+\frac{1}{2}}^{(xpec,\gamma)}), \tag{4.9}$$

with successive *corrector evaluation* (*xpece*) steps:

$$\widetilde{\mathbf{V}}_{k+\frac{1}{2}}^{(xpece,\gamma+1)} = \tfrac{1}{2}(\widetilde{\mathbf{V}}_{k+1}^{(xpec,\gamma+1)} + \widetilde{\mathbf{V}}_k), \tag{4.10}$$

to be used in successive evaluation of the optimal control and the nonlinear switching term. The predictor step is the *zeroth* corrector step $\widetilde{\mathbf{V}}_{k+\frac{1}{2}}^{(xpec,0)} = \widetilde{\mathbf{V}}_{k+\frac{1}{2}}^{(xpe)}$. The corrector procedures are repeated for $\gamma = 0$ to γ_{max} or until a predetermined stopping criterion is met.

Parallelism is effective in the Crank-Nicolson computation loops over the state space indices in the numerical equations (4.1-4.10). The data dependence across the time index k loops make it difficult to parallelize the code beyond the state loops. The finite element structure saves on memory requirements over finite differences, but the parallelization properties are similar to that fo the implementation for finite differences.

5. STABILITY AND CONVERGENCE

In [14], the convergence and stability of the finite difference method for solving the Bellman functional PDE is derived. The analysis is based on von Neumann's Fourier stability applied to a linearized comparison equation. In the analysis of the convergence and stability of the finite element method, we adapted to the same single state linearized, constant coefficients, nonfunctional PDE comparison equation. Instead of using von Neumann Fourier stability, eigenfunctions expansion is used in the analysis. The one-dimensional heuristic comparison equation takes the form

$$0 = V_t + \widehat{B}\cdot V_x + \widehat{A}\cdot V_{xx} \tag{5.1}$$

where

$$\widehat{A} \geq \max[|\frac{1}{2}G^2(x,t)|]$$
$$\widehat{B} \geq \max[|F_0(x,t) + \tfrac{1}{2}F_1(x,t) \cdot u|]$$

are nonnegative constants, which are different form $A_{k+1/2}$ and $B_{k+1/2}$ in the previous section. This equation, with maximal coefficients, may be thought of as the worst possible case of the Bellman's equation (2.4), but we ignore the inhomogeneous costs and Poisson related terms, which would make our analysis intractable if we included them.

For a standard finite element analysis, consider the following self-adjoint alternate form of (5.1),

$$0 = e^{\widehat{B}x/\widehat{A}} V_t + \left(\widehat{A} e^{\widehat{B}x/\widehat{A}} V_x\right)_x , \tag{5.2}$$

where $\exp(\widehat{B}x/\widehat{A})$ is an integrating factor or the spatial part of (5.1. Applying the Galerkin approximation (3.1) to the variational formulation of (5.1), we have the backward matrix ODE

$$0 = \widehat{M}\frac{d\widehat{\mathbf{V}}}{dt}(t) + \widehat{K}\widehat{\mathbf{V}}(t), \tag{5.3}$$

where

$$\widehat{M} = [(\phi_i, e^{\widehat{B}x/\widehat{A}}\phi_j)]_{N\times N}$$
$$\widehat{K} = -\widehat{A} \cdot [(\phi_i', e^{\widehat{B}x/\widehat{A}}\phi_j')]_{N\times N}, \tag{5.4}$$

are, respectively, the mass matrix and stiffness matrix. ϕ_j' is the derivative of the basis function ϕ_j with respect to x.

Suppose the whole domain is divided into N_e elements. The same matrix ODE is applied to both a single element and the whole domain. Let k_e and m_e be the stiffness and mass matrix for the ODE of a single element respectively. Then

$$0 = m_e \frac{d\widehat{\mathbf{v}}_e}{dt} + k_e \widehat{\mathbf{v}}_e, \tag{5.5}$$

where

$$m_e = [(\phi_i, e^{\widehat{B}x/\widehat{A}}\phi_j)]_{n_e \times n_e}$$
$$k_e = -\widehat{A} \cdot [(\phi_i^{'}, e^{\widehat{B}x/\widehat{A}}\phi_j^{'})]_{n_e \times n_e}, \tag{5.6}$$

n_e is the number of nodes in a single element. The nodal values vectors, $\widehat{\mathbf{v}}_e$, are related to the element global nodal values vector, $\widehat{\mathbf{V}}$, by

$$\widehat{\mathbf{v}}_e = A_e \widehat{\mathbf{V}}, \tag{5.7}$$

where A_e are Boolean mapping matrices which maps the global numbering of the node in the whole domain into the local numberering of a node in an element.

The global stiffness matrix $\widehat{K}$ and global mass matrix $\widehat{M}$ can be *assembled* from the corresponding element matrices as

$$\widehat{K} = \sum_{e=1}^{N_e} A_e^T k_e A_e, \tag{5.8}$$
$$\widehat{M} = \sum_{e=1}^{N_e} A_e^T m_e A_e. \tag{5.9}$$

Consider the corresponding symmetric eigenproblems for the finite element approximation of the self-adjoint form (5.2,

$$\widehat{K}\widehat{\mathbf{y}} = \widehat{\lambda}\widehat{M}\widehat{\mathbf{y}}, \tag{5.10}$$

and

$$k_e \widehat{\mathbf{y}}_e = \widehat{\lambda}_e m_e \widehat{\mathbf{y}}_e, \tag{5.11}$$

for the global and element problems, respectively.

Consider the linear finite element space, in order to motivate the use of general matrix analysis. A typical finite element is given as

$$x_1 \leq x \leq x_1 + h = x_2, \tag{5.12}$$

where h is the size of an element. The linear basis functions on an element are then given as

$$\phi_1(x) = \frac{x_2 - x}{h},$$
$$\phi_2(x) = \frac{x - x_1}{h}.$$

The element mass matrix and element stiffness matrix for the self-adjoint problem are given respectively as

$$m_e = \frac{\widehat{A}}{\widehat{B}} e^{\widehat{B}x_1/\widehat{A}} \left\{ \frac{2}{\widehat{h}^2} \left(e^{\widehat{h}} - 1 \right) \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} + \frac{1}{\widehat{h}} \begin{pmatrix} -2 & 1 + e^{\widehat{h}} \\ 1 + e^{\widehat{h}} & -2e^{\widehat{h}} \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 0 & e^{\widehat{h}} \end{pmatrix} \right\}, \quad (5.13)$$

which is positive definite, where $\widehat{h} \equiv \widehat{B}h/\widehat{A}$, and

$$k_e = -\frac{\widehat{B}}{\widehat{h}^2} e^{\widehat{B}x_1/\widehat{A}} \left(e^{\widehat{h}} - 1 \right) \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}. \quad (5.14)$$

which is negative semi-definite, corresponding to the backward time property of the problem. Computations show that

$$\begin{aligned} &\max[\widehat{\lambda}_e^{(j)}] = 0, \\ &\min[\widehat{\lambda}_e^{(j)}] = -\frac{\widehat{A}}{h^2} \frac{\left(e^{\widehat{h}} - 1 \right)^2}{\frac{2}{\widehat{h}^2}(e^{\widehat{h}} - 1)^2 + \left(\frac{4}{\widehat{h}^2} - 1 \right) e^{\widehat{h}} - \frac{1}{\widehat{h}^2} \left(e^{\widehat{h}} + 1 \right)^2} \\ &\qquad \longrightarrow -\frac{12}{h^2}, \end{aligned} \quad (5.15)$$

as $\widehat{h} \equiv \widehat{B}h/\widehat{A} \to 0^+$.

Let $\widehat{\lambda}_e^{(1)}$ and $\widehat{\lambda}_e^{(n)}$ be the largest and smallest eigenvalue of (5.11), respectively. Since m_e is symmetric positive definite and k_e is symmetric negative semi-definite, for linear elements, Rayleigh's Theorem [8] implies

$$\widehat{\lambda}_e^{(n)} \widehat{\mathbf{y}}_e^T m_e \widehat{\mathbf{y}}_e \leq \widehat{\mathbf{y}}_e^T k_e \widehat{\mathbf{y}}_e \leq \widehat{\lambda}_e^{(1)} \widehat{\mathbf{y}}_e^T m_e \widehat{\mathbf{y}}_e. \quad (5.16)$$

Assembling k_e into $\widehat{K}$, we have

$$\widehat{\mathbf{y}}^T \widehat{K} \widehat{\mathbf{y}} = \textstyle\sum_{e=1}^{N_e} \widehat{\mathbf{y}}_e^T k_e \widehat{\mathbf{y}}_e \quad \leq \max_e[\widehat{\lambda}_e^{(1)}] \displaystyle\sum_{e=1}^{N_e} \widehat{\mathbf{y}}_e^T m_e \widehat{\mathbf{y}}_e = \max_e[\widehat{\lambda}_e^{(1)}] \widehat{\mathbf{y}}^T \widehat{M} \widehat{\mathbf{y}}, \quad (5.17)$$

and

$$\widehat{\mathbf{y}}^T \widehat{K} \widehat{\mathbf{y}} \quad \geq \min_e [\widehat{\lambda}_e^{(n)}] \sum_{e=1}^{N_e} \widehat{\mathbf{y}}_e^T m_e \widehat{\mathbf{y}}_e = \min_e [\widehat{\lambda}_e^{(n)}] \widehat{\mathbf{y}}^T \widehat{M} \widehat{\mathbf{y}}. \tag{5.18}$$

Therefore

$$\min_e [\widehat{\lambda}_e^{(n)}] \leq \widehat{\mathbf{y}}^T \widehat{K} \widehat{\mathbf{y}} / \widehat{\mathbf{y}}^T \widehat{M} \widehat{\mathbf{y}} \leq \max_e [\widehat{\lambda}_e^{(1)}], \tag{5.19}$$

that is, the range of the eigenvalue $\widehat{\lambda}$ for the eigenproblem (5.10) lies within range of the eigenvalues of the element eigenproblem (5.11).

Since Bellman functional PDE is a final time problem, therefore (5.3) is a backward time ODE. Following the Crank-Nicolson time discretization scheme, Eq. (5.3) in the k^{th} step becomes

$$0 = \widehat{M} \cdot \frac{\widetilde{\mathbf{V}}_k - \widetilde{\mathbf{V}}_{k-1}}{-DT} + \widehat{K} \cdot \frac{\widetilde{\mathbf{V}}_k + \widetilde{\mathbf{V}}_{k-1}}{2}, \tag{5.20}$$

i.e.

$$\widetilde{\mathbf{V}}_k = (I - \frac{DT \cdot \widehat{M}^{-1} \widehat{K}}{2})^{-1} (I + \frac{DT \cdot \widehat{M}^{-1} \widehat{K}}{2}) \widetilde{\mathbf{V}}_{k-1}, \tag{5.21}$$

where $\widetilde{\mathbf{V}}_k = \widehat{\mathbf{V}}(T_k)$. Suppose the starting final guess, $\widetilde{\mathbf{V}}_0 = \widehat{\mathbf{V}}_f = \widehat{\mathbf{V}}(T_f)$, is expanded in terms of the discrete orthonormal eigenfunction $\widehat{\mathbf{y}}_j$ of $\widehat{M}^{-1} \widehat{K}$, i.e.

$$\widetilde{\mathbf{V}}_0 = \sum_{j=1}^{N} \widehat{c}_j \widehat{\mathbf{y}}_j, \tag{5.22}$$

where

$$\widehat{c}_j = \widetilde{\mathbf{V}}_0^T \widehat{M} \widehat{\mathbf{y}}_j.$$

Substituting into (5.21) recursively, we have

$$\widetilde{\mathbf{V}}_k = \sum_{j=1}^{N} (\mu_j)^k \cdot \widehat{c}_j Byh_j, \tag{5.23}$$

where the amplification factor μ_j is given as

$$\mu_j = \frac{1 + \widehat{\lambda}_j \frac{DT}{2}}{1 - \widehat{\lambda}_j \frac{DT}{2}}. \tag{5.24}$$

Since $\widehat{\lambda}_j$ is nonpositive for linear elements, therefore

$$|\mu_j| \leq 1, \tag{5.25}$$

and the Crank-Nicolson scheme is asymptotically stable as $k \to \infty$.

The rate of convergence can also be determined from the corresponding eigenfunction expansion in the space-time domain [17]. The orthonormal eigenfunction expansions of the exact solution and the approximation, recalling the backward time nature of the problem, are given as

$$V^*(\mathbf{x}, t) \approx \sum_{j=1}^{\infty} c_j e^{\lambda_j (t_f - t)} Y_j(\mathbf{x}), \tag{5.26}$$

where

$$\begin{aligned} \int_{\mathcal{D}_x} Y_i(\mathbf{x}) Y_j(\mathbf{x}) d\mathbf{x} &= \delta_{i,j}, \\ c_j = \int_{\mathcal{D}_x} V(\mathbf{x}, t_f) Y_j(\mathbf{x}) d\mathbf{x}&, \end{aligned}$$

and

$$\widehat{V}(\mathbf{x}, t) = \sum_{j=1}^{N} \widehat{c}_j e^{\widehat{\lambda}_j (t_f - t)} \widehat{Y}_j(\mathbf{x}), \tag{5.27}$$

where

$$\begin{aligned} \int_{\mathcal{D}_x} \widehat{Y}_i(\mathbf{x}) \widehat{Y}_j(\mathbf{x}) d\mathbf{x} &= \delta_{i,j}, \\ \widehat{c}_j = \int_{\mathcal{D}_x} V(\mathbf{x}, t_f) \widehat{Y}_j(\mathbf{x}) d\mathbf{x}&, \end{aligned}$$

respectively.

There are two sources of error: the *initial error*, which is the error of the interpolated initial value, and the *evolution error*, which is the error that evolves with time [17]. When the interpolating functions in the finite element space are of order $k-1$, the initial error $\mathbf{V}_0 - \widehat{\mathbf{V}}_0$ is of order $(\Delta x)^k$. The evolution error is found when the same initial condition $\widehat{\mathbf{V}}_0$ is used as in both equations (5.26) and (5.27). The error in the eigenvalues and eigenfunctions are given [8, 17] as

$$\widehat{\lambda}_j - \lambda_j = O(h^{2(k-1)} \lambda_j^k), \tag{5.28}$$

and

$$||\widehat{Y}_j - Y_j||_2 = O(h^k \lambda_j^{k/2}), \tag{5.29}$$

as $h \to 0$. The error in the eigenfunctions is measured in the L_2 norm, where

$$||\widehat{Y}_j - Y_j||_2 = \left[\int_{\mathcal{D}_x} (\widehat{Y}_j - Y_j)^T (\widehat{Y}_j - Y_j) d\mathbf{x}\right]^{\frac{1}{2}}, \tag{5.30}$$

for some domain $\mathcal{D}_x$. Similarly, the difference in the weights [17] is

$$\widehat{c}_j - c_j = \int_{\mathcal{D}_x} \widehat{V}(\mathbf{x}, t_f)(\widehat{Y}_j - Y_j) d\mathbf{x} = O(h^k), \tag{5.31}$$

as $h \to 0$, ignoring the eigenvalue dependence. Therefore, comparing $\mathbf{V}$ and $\widehat{\mathbf{V}}$ in (5.26) and (5.27), the evolution error is also of order h^k.

While our application of the estimates of [8, 17] is quite straight-forward, our reduction of the more complex Bellman equation to a more standard PDE, that is more amenable to analysis, is not standard and is our more significant contribution, as our numerical results demonstrate.

6. MEMORY REQUIREMENTS

We have seen in the previous section that the order of accuracy of the finite element method depends on the degree of the interpolating function. For cubic interpolating functions, the error is of order $(\Delta x)^4$. In the conventional finite difference scheme, the local truncation error is of order $(\Delta x)^2$. With this observation, one can conclude that the numerical efficiency of n *mesh points* per state in the finite element method with cubic interpolating function is comparable with n^2 *mesh points* in the finite difference scheme.

It has been shown that the storage requirement for the finite difference method solution of the Bellman's functional PDE (2.4) grows exponentially as the number of states in the problem. If m is the number of states and M is the number of mesh points per state, then the storage requirement is given as

$$S_{fdm}^{m,M} = K_{fdm} \cdot m \cdot M^m. \tag{6.1}$$

In [5], we have shown that the asymptotic value of K_{fdm} in (6.1) is approximately 13. In the finite element (Galerkin) method, the resulting coefficient matrices ($A_{k+1/2}$ and $B_{k+1/2}$ in (4.2)) are sparse and the average number of nonzero entries per row depends on the dimension and the order of the interpolating function, if $H \equiv 0$. However, with the Poisson noise in the stochastic model and the corresponding functional argument $\mathbf{x} + \mathbf{H}_l$, sparseness and the usual local DE behavior is no longer assured. In case that the Poisson terms do not contribute significantly to array denseness, the extra nonzero entries due to the Poisson term is small, it is reasonable to assume that the storage requirement is given as

$$S_{fem}^{m,M} = K_{fem} \cdot M^m, \tag{6.2}$$

if only the matrices are stored in a compressed form where only the nonzero entries in each row are stored. Now for large state space problems, the storage requirement is dominated by matrices $A_{k+1/2}$ and $B_{k+1/2}$ in (4.2) and a pointer array with the same dimension as $A_{k+1/2}$ and $B_{k+1/2}$. Thus the asymptotic value of K_{fem} in (6.2) is given as

$$K_{fem} \approx 3 \times (\text{number of nonzero entries per row in } A_{k+1/2}, B_{k+1/2}). \tag{6.3}$$

Consider, for example, a 3 state problem with 64 mesh points in the finite difference solution. The number of mesh points per state in the finite element (Galerkin) method with comparable order of accuracy should be 8 and the number of nonzero entries per row in the coefficient matrix is 165 for a cubic interpolating function. The storage requirements for the two methods are

$$S_{fdm}^{3,64} \approx 3 \times 13 \times 64^3 \approx 10.2 \times 10^6 \tag{6.4}$$

$$S_{fem}^{3,8} \approx 3 \times 165 \times 8^3 \approx 0.25 \times 10^6 \tag{6.5}$$

This gives an approximate 40 times saving, a substantial improvement in terms of storage requirement.

7. CONCLUSIONS

It has been shown that the order of accuracy of the finite element method for the Bellman functional PDE depends on the degree of the interpolating basis function. For an interpolating function of degree $(k-1)$, the order of accuracy is $O(\Delta x)^k$. Since the order of accuracy for the finite element method can be controlled through the degree of interpolating function, higher order finite element methods can be used in high state-space problems. This has the effect that the same degree of accuracy can be maintained while coarser mesh points can be used, alleviating Bellman's curse of dimensionality. In Section 6, we have demonstrated that an almost 40 times saving in memory requirements is achieved when the finite element method is compared with the finite difference method for a 3-state problem.

The *National Computing Initiative* [15] noted that stochastic dynamic programming is computationally demanding. As technology advances, stochastic control problems with larger size can be handled through the use of advanced computing facilities and computing methods. Preliminary results with the finite element method indicates that the method provides advantages for reducing dimensionality problems in existing hardware. The solutions for stochastic dynamic programming are far from perfect. Much more needs to be done to solve problems of higher dimensions. They include

- more advanced supercomputing methods such as the use of data parallel methods which can achieve scalar, linear growth in both memory and computational requirements [18] ;
- development of more general and efficient code for the finite element method to cut down the number of nodes per state;
- multigrid methods;
- domain decomposition methods.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under Grants DMS 88 06099 and DMS 91 02343, the Advanced Computing Research Facilities of the Argonne National Laboratory, the National Center for Supercomputing Applications of the University of Illinois at Urbana, the UIC Workshop Program on Scientific Supercomputing and the UIC Software Technologies Research Center (SofTech).

REFERENCES

[1] M. Akian, J.P. Chancelier and J.P. Quadrat, Dynamic programming complexity and applications, *Proc. 27th IEEE Conf. on Decision and Control* 2 : 1551-1558, Dec. 1988.

[2] L. Arnold, *Stochastic Differential Equations : Theory and Applications*, Wiley, New York, 1974.

[3] A.E. Bryson and Y.C. Ho, *Applied Optimal Control*, Hemisphere Publ. Corp., New York, 1975.

[4] S.L. Chung and F.B. Hanson , Parallel Optimization for Computational Stochastic Dynamic Programming, in *Proc. 1990 Int. Conf. Parallel Processing, Algorithms and Applications*, (Pen-Chung Yew, Editor), Pennsylvania State University Press, University Park, vol. 3: 245-260, Aug. 1990.

[5] S.L. Chung and F.B. Hanson , Optimization Techniques for Stochastic Dynamic Programming, *Proc. 29th IEEE Conf. on Decision and Control*, 4:2450-2455, Dec. 1990.

[6] S.L. Chung, Supercomputing Optimization of Stochastic Dynamic Programming, Ph.D. Thesis, University of Illinois at Chicago, Aug. 1991.

[7] W.H. Fleming and R.W. Rishel, *Deterministic and Stochastic Optimal Control*, Springer-Verlag, New York, 1975.

[8] I. Fried, *Numerical Solution of Differential Equations*, Academic Press, New York, 1979.

[9] I.I. Gihman and A.V. Skorohod, *Stochastic Differential Equations*, Springer-Verlag, New York, 1972.

[10] I.I. Gihman and A.V. Skorohod, *Controlled Stochastic Processes*, Springer-Verlag, New York, 1979.

[11] F.B. Hanson, Computational Dynamic Programming for Stochastic Optimal Control on a Vector Multiprocessors, *Technical Memorandum No. 113, Mathematics and Computer Science Division, Argonne National Laboratory*, June 1988.

[12] F.B. Hanson, Parallel computation for Stochastic Dynamic Programming: Row versus Column code orientation, in *Proc. 1988 Int.Conf. Parallel Processing, Algorithms and Applications*, (D.H. Bailey, Editor), Pennsylvania State University Press, University Park, vol. 3:117-119, Aug., 1988.

[13] R.E. Larson, *State Increment Dynamic Programming*, American Elsevier, New York, 1968.

[14] K. Naimipour and F.B. Hanson, Convergence of Numerical Method for the Bellman Equation of Stochastic Optimal Control with Quadratic Costs and Constrained Control, submitted 1991.

[15] H.J. Reveché, Chair, A National Computing Initiative: The Agenda for Leadership, Society for Industrial and Applied Mathematics, Philadelphia, 1987.

[16] Z. Schuss, *Theory and Applications of Stochastic Differential Equations*, Wiley, New York, 1980.

[17] G. Strang, G.J. Fix, *An Analysis of the Finite Element Method*, Englewood Cliffs, Prentice-Hall, New York 1973.

[18] H.H. Xu , F.B. Hanson and S.L. Chung, Parallel Optimal Data Parallel Methods for Stochastic Dynamic programming, accepted in *Proc. 1991 Int. Conf. Parallel Processing* , Pennsylvania State University Press, University Park, Aug. 1991.