Computational Stochastic Control: Basic Foundations, Complexity and Techniques

Floyd B. Hanson
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago,851 Morgan St., M/C 249
Chicago, IL 60607-7045, USA
e-mail: hanson@math.uic.edu

***Abstract*— Much research and education in control systems is purely mathematical, but computational advances in stochastic control problem solving can be used beyond the limits of theoretical mathematics. Theoretical and computational mathematics are complementary. Computation is important where the problem is mathematically intractable, highly dimensional as in stochastic dynamic programming or urgently in need of answers as in competitive financial predictions. Many advances in solving large scale control problems have been gained through technical improvements in computing hardware, but as many advances have been made in the development of better algorithms. Both analysis and computation are important in solving problems. Both rely on mathematics, but rely on them in different ways. An important part of educational training is general preparation for problem solving since postgraduate jobs are uncertain in the current world.**

In this expository control education paper, some basic computational considerations, high performance computers and useful algorithms are surveyed. Much of the knowledge gained in research has been transferred to classes in computation and control, so that student instruction is at the leading edge, a buffer against obsolescence. An important general lesson in computational education is that the computation forms the other half of mathematics, beyond regular mathematics courses. Computation has its own algebra, oriented to finite precision arithmetic, and its own numerically oriented analysis. The view is that of an applied analyst and computational control scientist explaining the field to those with a regular mathematics background.

I. INTRODUCTION

There are great demands on students learning about stochastic analysis and stochastic control theory, since the courses are based on very broad mathematical foundations. These foundations include probability theory, stochastic processes, analysis for continuous and discontinuous functions, measure theory, other advanced topics in functional analysis, and differential equations. Stochastic analysis and control needs to be made more accessible to students interested in applications, as early as entry-level graduate studies. Much of the material can be developed from more concrete analysis and computation than from the abstract analysis. Computational methods serve to strengthen the concrete analysis by demonstrating solutions and other properties, while permitting the solution of problems that are beyond the help of abstract analysis. Abstract analysis plays a small role in computations anyway, although important for foundational issues.

When the student is ready for a more abstract approach to studying the finer points of stochastic problems, the student will be able to approach them with more intuition and maturity. The concrete approach should be closer to the spirit of applied mathematics and applied probability, using systematic derivations of results through a chain of justifying formulas, in the sense of Hilbert, without using Hilbert spaces or measure theory. The level of explanation should be understandable to a student who has had a good undergraduate course in analysis or advanced calculus. In addition, computational methods enable the study of large scale problems and lead to skills that enhance the employability of the student in an uncertain job market.

The introductory study of stochastic control needs to be at a more concrete level to facilitate the integration of theory and computation. It is important to emphasize the lack of smoothness in stochastic diffusions and the lack of continuity in jump processes. This lack of continuity and smoothness is important in explaining the differences between the chain rule for stochastic processes and the chain rule for deterministic, continuous processes. An applied stochastic processes and control of jump-diffusions text [14] is currently being prepared that is accessible to entry level graduate students in applied mathematics, computational sciences, engineering, finance and other industries.

II. BASIC COMPUTATIONAL CONSIDERATIONS BEYOND REGULAR MATHEMATICS

In many computational science applications there are concepts from regular mathematics courses, such as calculus, that may lead to blunders when applied to actual computational problems. The gap in knowledge is not just particular to computational control, but is general and has more to do with the different computer-oriented algebraic structures than the idealistic structures of the regular mathematics. Also, there are important goals in computation such as speed and efficiency in computation, but are not in the regular mathematics. For instance, in regular mathematics there is an interest in convergence, existence and uniqueness, but in computation just convergence is not good enough, since both the rate of convergence in finite time are critical. Uniqueness and existence follows from computational construction. The accuracy of an approximate answer is also important. Thus, there are important differences in mathematical structure, speed, efficiency and accuracy. The intent of this section is to present some of the basic ideas of the foundations of numerical analysis to those who are not very aware of the computational side of mathematics.

A. Finite Precision Representation

The most basic difference in algebraic structure is due to the computer finite precision representation of real numbers. The algebra of this fascinating piece-wise constant number system attracted the attention of both pure and applied mathematicians about four decades ago with Prager and Davis at Brown University, Lax at New York University, Forsythe at Stanford University and others. Much of their influence was by way of unpublished course notes. For the representation of real numbers on computers, the floating point number system is finite, i.e., $p < \infty$ digits, and is primarily based upon the binary number system, i.e., base 2, at the machine level. Also, the representation is a *relative* one with a normalization requiring the first digit to be nonzero, i.e., one in binary, except for the number zero which has a special representation. This normalization avoids the presence of leading zeros and the exponent allows the decimal point to float to represent numbers very small to very large in magnitude.

Definition: A *normalized, floating point representation* of the nonzero number x can be represented mathematically as

$$\text{float}[x] = \pm_f \left(1 + \sum_{i=2}^{p} d_i/b^i\right) * b^{\pm_e \sum_{j=1}^{q} e_j * b^{j-1}},$$

where usually $b = 2$, p is the number of digits in binary precision, d_i is the ith binary digit (0 or 1 bit) but rounded in binary from the input x, e_j is the jth binary digit in the exponent, q is the number of binary digits in the exponent, $\pm_f$ is the sign of the fraction and $\pm_e$ is the sign of the exponent.

The actual computer storage is much more complicated, allowing for special numbers like $\texttt{NAN}$ (Not A Number), which represents an improper result. The normalized leading binary digit $d_1 = 1$ when $x \neq 0$ is not stored since it can be uniquely recovered from the representation which can be 24 bits in single or float precision (roughly 7 decimal digits) and 53 in double precision (roughly 16 decimal digits). There are also extended and quadruple precision. The exponents are bounded above and below by the amount of over-flow and under-flow permitted in the precision.

The *user representation* is usually quite different since the floating point number is represented in output in decimal scientific notation as

$$\text{float}[x] \simeq \pm_f D_1.D_2D_3\ldots D_P\text{e}\pm_e E_1\ldots E_Q,$$

where the D_i, for $i = 1 : P$, are the converted and rerounded decimal digits, $1 \leq D_1 \leq 9$ if $x \neq 0$, but $0 \leq D_i \leq 9$, for $i = 1 : P$, the E_j for $j = 1 : Q$ are the corresponding decimal exponent digits, and $\texttt{e}$ is the computer scientific notation symbol marking the beginning of the decimal exponent. More information on floating point representation can be found in numerical texts [3] or on-line [11].

B. Machine Epsilon

Due to the finite representation, there is a minimum positive number called the machine epsilon, $\texttt{maceps}$, that when added to one gives a result that is greater than one, i.e.,

Definition: $\texttt{maceps} = \min[\epsilon|\epsilon > 0, \text{float}[1+\epsilon] > 1]$.

Theorem: For floating point representation computations with rounding in base b and precision p, $\texttt{maceps} = b^{1-p}/2$.

On almost all current computing systems, the base is binary, $b = 2$, and the usual precision should be double in professional computations, so $p = 53$ in the IEEE 754 standard or *maceps* $= 2^{-53} \simeq 1.11 \times 10^{-16}$. As a result of this finite precision, familiar algebraic laws such as the distributive law no longer hold, except approximately.

C. Catastrophic Cancellation

If the limits of the floating point representation are unknown to the user, then mistakes, big and small can be made. If two calculations are performed by different order of operations, then different results are possible, even if the two different orders are mathematically equivalent. However, the results usually differ in the order of magnitude of some multiple of the machine epsilon, except in the case of *catastrophic cancellation*. Catastrophic cancellation is when many of the most significant digits are cancelled in floating point calculations.

Derivatives: The best catastrophic example is the simulation of a derivative by Newton's quotient, Q_N,

$$f'(x) \simeq Q_N = (f(x+h) - f(x))/h, \ |h| > 0,$$

by taking h sufficiently small, but not too small. If h is too small then the machine epsilon property is operative, i.e., $\text{float}[x+h] = \text{float}[x]$, for $0 < |h| < \texttt{maceps}/|x|$, $\text{sgn}(h) = \text{sgn}(x)$, $|x| > 0$, and the numerator is zero in floating point precision. Hence, the instruction in the regular calculus or analysis class to *take h as small as you please* would be wrong and catastrophic in floating point arithmetic. In general, the smaller the step size h, the greater are floating point errors due to the piece-wise constant system and the more cumulative errors present:

Folk Theorem: The rule of thumb in numerical computations is to take the step size not too small or not to big, i.e., to rely on roughly half the number of digits in the precision p.

Floating point representation is piecewise constant, but the pieces are quite small so that on a scale that is much larger than the $\texttt{maceps}$, it is a reasonable linear approximation to the real number system, i.e., $\text{float}[x] \simeq x$ in the large.

D. Symbolic Computation: A Disclaimer

On many computers, symbolic computing systems such as Maple$^{\text{TM}}$ and Mathematica$^{\text{TM}}$ are available and an approximation to infinite precision arithmetic can be simulated on the computer, although all computers are finite. Thus, the computation of a large number of digits in an answer will take a correspondingly large amount of computer time. Obviously, many of the comments about finite floating point precision made here do not apply to symbolic computation.

E. *Inefficient Linear Algebra Methods*

Several linear algebra methods taught in regular linear algebra courses are fine in theory, but very inefficient or wrong in computational practice. The standard computational method for solving systems of linear algebraic equations, $A * \vec{x} = \vec{b}$ given n vector $\vec{b}$ and coefficient matrix A, is *Forward Gaussian Elimination with Back Substitution*. For nth order systems, the number of floating point operations is the order of $2n^3/3$. However, row pivoting and row scaling or full row-column pivoting is necessary to make the method robust in accuracy and stability. Closely related methods are the variants *LU (Lower and Upper triangular forms) Decomposition*, which are more efficient for production problems since they only reduce the coefficient matrix A once for many right hand sides.

Cramer's Rule: Often students get the idea of using Cramer's rule to solve systems larger than $n > 4$, say, either because they have only used it to solve small, toy homework problems or because it is used in linear algebra theory regardless of the dimension without practical qualifications. It can be shown by the properties of the exponential series it has exponential order computational complexity for nth order system solution, i.e., $O(e * (n + 1)!)$, floating point operations [10].

Gauss-Jordan Elimination: In the theoretical linear algebra courses the Gauss-Jordan diagonalization form of Gaussian elimination is taught, but this is twice as costly as Forward Gaussian Elimination with Back Substitution, so this would be unwise for large scale industrial production systems.

III. COMPUTATIONAL DETERMINISTIC CONTROL VERSUS COMPUTATIONAL STOCHASTIC CONTROL

This paper is about about computational stochastic control, but computational deterministic control is mentioned briefly for contrast.

A. *Computational Deterministic Control*

Deterministic control problems [15] can be cast as systems of ordinary differential equations (ODEs) so many standard numerical methods can be used. For example, if $\vec{X}(t)$ is the *state* n-vector of the system in continuous time t, $\vec{U}(t)$ is the *control* m-vector, the ODE equation for the dynamics is

$$d\vec{X}(t)/dt = \vec{F}(\vec{X}(t), \vec{U}(t), t), \tag{1}$$

where $\vec{F}(\vec{x}, \vec{u}, t)$ is the *plant function*, possibly nonlinear, and the *objective* is the minimal cumulative running costs $C(\vec{x}, \vec{u}, t)$ on (t_0, t_f) plus terminal cost $Z_f(\vec{x}, t)$ at t_f, i.e.,

$$V[\vec{X}, \vec{U}] = \int_{t_0}^{t_f} C\left(\vec{X}(t), \vec{U}(t), t\right) dt, + Z_f\left(\vec{X}(t_f), t_f\right). \tag{2}$$

In the *Hamiltonian formulation* [15], the objective integrand and the dynamics and combined and optimization is performed on the *Hamiltonian*,

$$H(\vec{X}(t), \vec{U}(t), \vec{\lambda}(t), t) \equiv C(\vec{X}(t), \vec{U}(t), t) + \vec{\lambda}^T(t)\vec{F}(t)(\vec{X}(t), \vec{U}(t), t),$$

where $\vec{\lambda}(t)$ is the *Lagrange multiplier*. The optimal trajectory triple critical point, $(\vec{x}^*(t), \vec{u}^*(t), \vec{\lambda}^*(t))$ assuming sufficient smoothness conditions, are given by *Hamilton's equations*,

$$\left(\frac{\partial H}{\partial \vec{\lambda}}\right)^* = \frac{d\vec{x}^*(t)}{dt}, \quad \left(\frac{\partial H}{\partial \vec{x}}\right)^* = -\frac{d\vec{\lambda}^*(t)}{dt}, \quad \left(\frac{\partial H}{\partial \vec{u}}\right)^* = \vec{0},$$

a set of three vector ODEs using optimal control $\vec{u}^*(t)$. The side conditions depend on the application and are tabulated nicely in [15]. The $\vec{x}^*(t)$ satisfies an initial conditions at t_0 and $\vec{\lambda}^*(t)$ satisfies a final conditions at t_f. Except for simple or analytical homework problems, usually numerical discretization and backward-forward iterations are required until the solution converges to prescribed accuracy. If there are M discrete time nodes, $T_j = t_0 + (j - 1)\Delta T$ for $j = 1 : M$ with $\Delta T = (t_f - t_0)/(M - 1)$, then the n dimensional state vector $\vec{x}^*(t)$ is discretized as $\vec{x}^*(T_j) = \vec{X}_j = [X_{i,j}]_{n \times M}$ or $n \cdot M$ discrete variables. For the three vector solution the computational complexity or the order of the computational cost is $O(3n \cdot M)$ per iteration, i.e., bi-linear in the dimension and number of time nodes, a very manageable computational problem, even for today's powerful personal computers.

MATLAB$^{\text{TM}}$ has a good number of control Toolboxes to handle problems. There are also several good on-line tutorials available, such as *Control Tutorials for MATLAB And Simulink* [17]. Bernstein has several insightful essays, e.g., *Classical Control* [2]. A longer list of on-line control information is given on the *Control Tools* web-page [12].

B. *Computational Stochastic Control*

By contrast, computation for stochastic control is often quite different and can concern partial differential equations (PDEs) [14]. Many of these stochastic control computational methods are formulated in terms of the *PDE dynamic programming* of Bellman [1]. However, this PDE formulation leads to exponential complexity numerically, called the *Curse of Dimensionality* when compared to the bilinear complexity in the deterministic case.

For the case where the added stochasticity is a stochastic jump-diffusion, the dynamical system is a stochastic differential equation (SDE) and continuous time has the form:

$$\begin{aligned} d\vec{X}(t) &= \vec{F}(\vec{X}(t), \vec{U}(t), t)dt + \vec{G}(\vec{X}(t), t)d\vec{W}(t) \\ &\quad + \vec{J}(\vec{X}(t), t)dP(t), \end{aligned} \tag{3}$$

where $\vec{G}(\vec{x}, t)$ is the volatility array; $d\vec{W}(t)$ is the standard differential diffusion process, normally distributed with mean $\vec{0}$ and diagonal covariance $I_r dt$, I_r is the rth order identity; and $\vec{J}(\vec{X}(t), t)$ is the random jump amplitude associated with the Poisson differential process $dP(t)$ which has a common mean and variance portional to the jump rate, i.e., $\lambda(t)dt$. The $dP(t)$ and J are assumed independent given a jump, while dP and $\vec{J}$ are independent of dW.

The object in the stochastic control case is taken to be like the deterministic case (2) for comparison. The main change is that the cost objective is a functional of the the diffusion $\vec{W}(t)$ and jump $P(t)$ processes through the state $\vec{X}(t)$ process and that the initial time t_0 is replaced by t in order to vary time for dynamic programming, so

$$V[\vec{X},\vec{U},P,\vec{W},t]=\int_t^{t_f} C\left(\vec{X}(\tau),\vec{U}(\tau),\tau\right)d\tau+Z_f\left(\vec{X}(t_f),t_f\right).$$

The conditional expectation is taken to smooth out the stochastic fluctuations for a well-defined optimization. Then the optimal expected total cost starting at time t is

$$v^*(\vec{x},t) \;=\; \min_{\vec{u}}\left[\mathop{\mathrm{E}}_{P,\vec{W}}\left[V\left[\vec{X},\vec{U},P,\vec{W},t\right]\middle|\,\vec{X(t)}=\vec{x},\vec{U(t)}=\vec{u}\right]\right],$$

and the argument of the minimum being the optimal feedback control vector $\vec{u}^*=\vec{u}^*(\vec{x},t)$. Upon application of Bellman's Principle of Optimality [1] and the stochastic chain rule (the nonanalytic properties of $P(t)$ and $\vec{W}(t)$ introduce extra terms beyond the regular calculus chain rule), the *PDE of Stochastic Dynamic Programming* is

$$\begin{aligned}0 \;=\; & \frac{\partial v^*}{\partial t}(\vec{x},t)+\frac{1}{2}\mathrm{Trace}\left[GG^T\nabla_x\left[\nabla_x^T\left[v^*\right]\right]\right]\\ & +\min_u\left[C(\vec{x},\vec{u},t)+\vec{F}^T(\vec{x},\vec{u},t)\nabla_x\left[v^*(\vec{x},t)\right]\right. \qquad (4)\\ & \left.\lambda\int_Q\left[v^*(\vec{x}+\widehat{J}(\vec{x},q,t),t)-v^*(\vec{x},t)\right]\phi_Q(q)dq\right],\end{aligned}$$

where $0<t<t_f$, Q and q are the Poisson amplitude mark random and realized variables on the mark space $\mathcal{Q}$ with density $\phi_Q(q)$, and $\widehat{J}(\vec{x},q,t)$ is the jump amplitude kernel. The final condition is $v^*(\vec{x},t_f)=Z(\vec{x},t_f)$. Eq. (4) is no regular PDE, due to the presence of the control minimization. Thus, the dual output of the dynamic program is the value of the optimal cost $v^*(\vec{x},t)$ and the optimal feedback control vector $\vec{u}^*(\vec{x},t)$. Another difference is that the last term in (4) is the Poisson jump integral which leads to global dependence unlike the local dependence of the optimal cost gradient $\nabla_x[v^*(\vec{x},t)]$ and diffusion term.

It is state-time vector solution set dependence $\{v^*(\vec{x},t),\vec{u}^*(\vec{x},t)\}$ on $\vec{x}$ and t, that makes the stochastic case more computationally complex than $x(t)$ the deterministic case. If time is fixed at a single discrete value T_k, where $k=1:N_t$, the independent discretization of the n-dimensional state $\vec{x}$ is $\vec{X}_{\vec{j}}=[X_{i,j_i}]_{n\times1}$ where $\vec{j}=[j_i]_{n\times1}$, $j_i=1:N_x$ for $i=1:n$ and N_x is the common number of state nodes, However, $\vec{X}_{\vec{j}}$ only represents one point in state space and there are a total N_x^n numerical nodes in n dimensions. Thus, total numerical representation of $v(\vec{x},T_k)$ is

$$V^{(k)}=[V^{(k)}_{j_1,j_2,\cdots j_n}]_{N_x\times N_x\times\cdots\times N_x},$$

per time step k, so that the computational complexity is

$$CC(N_x,n)=O(N_x^n)=O(\exp(n\ln(N_x))), \qquad (5)$$

exponential in the dimension symbolizing the exponential computational complexity of *Curse of Dimensionality*. This is also the exponential order of the complexity for solving multi-dimensional PDEs. For the optimal control vector the order in n times this order, but that does not change the exponential order complexity. Further, for second order finite difference errors, the total error will be $E_T(N_x,n)=O(N_x^{-2})$. So even if the order of the complexity is fixed, i.e., $N=N_x^n$, then

$$E_T(N^{1/n},n)=O\left(N^{-2/n}\right)\to O(1) \qquad (6)$$

as $n\to+\infty$ for fixed N, i.e., diminishing accuracy in the limit of large dimension.

There are many other computational issues but there is not enough space here to discuss them (see the chapter [8]). Also, as with any PDE, the computational mesh ratio must be selected carefully or the computational PDE method will not converge. Another problem can arise when the drift or the deterministic plant function $\vec{F}$ dominates the stochastic terms so that the type of PDE changes from the diffusion-like parabolic type to wave-like hyperbolic type, leading to nonuniform numerical oscillations, which can be handled by *Upwinding*, using forward or backward finite differences [16]. When Poisson jump processes are used in the stochastic dynamical model, then a globally dependent jump integral can appear in the *PDE of Stochastic Dynamic Programming* (4), so the numerical approximations may require interpolation techniques [21]. Still another problem is the proper handling of boundary conditions for stochastic processes which require boundary reflections [16] using auxiliary processes.

IV. HIGH PERFORMANCE COMPUTING ADVANCES

High performance computing, using massively parallel computers and vector supercomputers can alleviate but not overcome the *Curse of Dimensionality*. Parallel and vector computation can permit the solution of higher dimension than was previously possible and thus permit more realistic dynamic programming applications. Large scale problems of great importance are called *Grand or National Challenge* problems [9] of high performance computing. The availability of high performance parallel processors have made it possible to compute optimal policies and values of control systems for much larger dimensions than was possible earlier. Today's parallel computer clusters of networked personal computers and workstations are making parallel computers a less expensive [9]. Advances in algorithms have also played a comparable role.

The 2002 panel on the *Future Directions in Control, Dynamics, and Systems* [18] places a lot more emphasis on control education and computation plays an integral role in the investigation of control problems throughout the report.

A. High Performance Computers

Much of the initial motivation in computational stochastic control for this author was the optimal harvesting of multi-dimensional or multi-species natural resources with multiple economics in a disastrous environment [5]. The control in these models is the species commercial and recreational harvesting effort, the harvest rates per unit species population. These models used Poisson processes to simulate species disastrous events, so were much more complicated analytically than the smooth, local stochastic diffusion processes. However, it was the multi-dimensional resources problem that caused time consuming computations and made local main-frame computing resources inadequate, requiring the need for advanced computing facilities. The advanced large scale computational demands required new and more efficient numerical methods [7] that were quite different and sometimes contrary to the standard methods for computation on serial computers.

As the number of accessible vector and parallel processors increased, solving stochastic dynamic programming problems larger dimensions became feasible. Cray vector supercomputers with a few and up to more than a dozen processors, massively parallel Thinking Machines CM-2 and CM-5 as well a massively parallel Cray T3D/T3Es have been used on these problems. Much of the effort was making the vector or parallel code more efficient by reducing data dependencies, making the code more transparent to the compilers, eliminating unnecessary overhead and other techniques. Many of advanced computing techniques are summarized in the chapter [8]. An important spin-off benefit was starting a course on parallel processing and supercomputing in 1986 [9] as technology transfer from this large-scale control research, and also to help train graduate students in computational stochastic control research.

B. Advanced Data Structures

Further, advanced parallel and vector optimizations are dependent on the application data structures. For certain applications, the code runs faster if arrays are accessed the way they are stored in memory, e.g., by columns for Fortran arrays or by rows in C arrays. In early applications [6], a vector data structure was developed to represent the full n-dimensional state space to make it relatively easy to change code dimensions and also to vastly increase the parallel work load balance by maintaining a large pool of work.

C. Scientific Visualization in High Dimensions

The stochastic dynamic programming output in n-dimensions is the optimal cost scalar $v^*(\vec{x},t)$ and the optimal feedback control m-vector $\vec{u}^*(\vec{x},t)$. They both depend on $(n+1)$ space-time variables. In addition, these solutions can depend significantly on k parameters, e.g., volatility, jump rates and jump amplitudes. Hence, displaying the resulting solution can be challenging beyond several dimensions. For this reason, our research group developed a real implementation of multi-dimensional control problem output called *I/O view* [19], an inner and outer world view. In the inner world the optimal value or an optimal control component can be displayed in a 3-dimensional surface projection versus two other independent variables or parameters. In the outer world three additional values of three other variables or parameters can be selected.

V. ALGORITHMS ADVANCES

While there have been many algorithmic advances in the recent decades, we will concentrate on control relevant algorithmic advances.

A. Canonical Models

Many control problems can be reduced in analytical and computational complexity if a canonical model is appropriate and leads to a separation of variables decomposition with a reduction of dimensionality.

LQGP Problem: One canonical problem, often used in stochastic control, is the LQG problem which has linear (L) dynamics, quadratic (Q) costs and Gaussian (G) or diffusion noise. The LQGP or JLQG problem is an extension to Poisson jump processes [20]. Both problems preserve the dynamic programming features due to the Markov stochastic properties. Our reseach group have developed refinements and computational implementation of the LQGP problem [20] for a stochastic multi-stage manufacturing system. The form of the system leads to a separation of variables form such that variables separate into an explicit quadratic function of the state, but with coefficients that implicitly depend on time,

$$\begin{aligned} v^*(\vec{x},t) &= \tfrac{1}{2}\vec{x}^T S(t)\vec{x} + \vec{D}^T(t)\vec{x} + E(t), \\ \vec{u}^*(t) &= -K(t)\left[S(t)\vec{x} + \vec{D}(t)\right], \end{aligned} \tag{7}$$

in the case of unconstrained control, where the $K(t)$ is the control gain matrix. The coefficients, matrix $S(t)$, vector $\vec{D}$ and scalar $E(t)$, are uni-directionally coupled through matrix ODEs and need to be determined.

CRRA Utility in Finance and Economics: A similar separation of state and time dependence can be achieved in an optimal portfolio with wealth consumption problems, if a *Constant Relative Risk Aversion (CRRA)* utility is the cost function, i.e., the running cost C is a power function $\mathcal{U}(w) = w^\gamma/\gamma$, where w is the wealth state variable. The optimal value of the investment portfolio separates into a known function of the wealth, the CRRA utility, and an unknown function of time, $v_0(t)$, i.e., $v(w,t) = \mathcal{U}(w)v_0(t)$, again reducing the dimensionality since the wealth w dependence is given. The optimal control variables are the stock fraction $u^*(w,t)$ and the consumption $c^*(w,t)$ also have a given but more complicated form for jump-diffusions [13].

B. Markov Chain Approximations

Kushner and co-workers [16] have developed many numerical approaches to stochastic control, with special emphasis on the *Markov chain approximation* method. The state process $\vec{x}(t)$ is discretely approximated by a Markov chain $\vec{\xi}(n,h)$ with the time interpolation step $\Delta t(n,h)$ where n is the discrete time index and h is the scale of the state change, like Δx. The interpolation time step $\Delta t(n,h)$ is chosen so that the Markov chain is “locally consistent” in probability with the stochastic process in continuous time. When applied to the approximation of the optimal value function, the coefficients are selected according to transition probabilities between chain stages of the Markov chain by choice of the step $\Delta t(n,h)$. The Markov chain approximation is compared with the PDE finite difference method in [8]. See the comprehensive book [16] for the complete details.

C. Pseudo and Quasi Monte Carlo Methods

When the state space dimension is greater than four or so, Monte Carlo methods can be more efficient than finite difference or Markov chain approximation methods, since these methods suffer from the curse of exponential complexity (5) or diminishing accuracy (6). On the other hand, if the problem as an integral over the state space and taking a random state space sample using a pseudo random number generator [4], then *Monte Carlo Approximation* for the N-point sample average approximates the integral average over state volume V. That is,

$$\int_V f(\vec{x})d\vec{x} \ \simeq \ \langle\ f\ \rangle_N \ \equiv \frac{1}{N}\sum_{i=1}^{N} f(\vec{x}_i). \qquad (8)$$

The average probable error is $E_N \ = \ O\left(1/\sqrt{N}\right)$, independent of dimension n for identically distributed normal random samples and N is sufficiently large. Compared to the finite difference error in (6), the Monte Carlo approximation will be more accurate and faster converging for larger dimensions.

There are many variations of the Monte Carlo method. One is the *Quasi-Monte Carlo Method*, using a quasi-random number generator [4]. It uses a systematic covering of state space with an error that can approach $O\left(\ln(N)/N\right)$, a smaller order than $O(1/\sqrt{N})$ for large N. Other techniques [4] often used are *importance sampling* which uses a weight function $g(\vec{x})$ that captures the important features of the integrand $f(\vec{x})$ or *stratified sampling* which uses domain decomposition to increase efficiency. In stochastic financial engineering there are many results using Monte Carlo methods [4].

VI. CONCLUSIONS

Computational stochastic control has been briefly surveyed from foundations in numerical analysis to some of the current computational efforts in solving stochastic control problem applications. The computational complexity of deterministic and stochastic control have been compared, explaining the greater complexity of stochastic control problems. Some examples of advanced computers and algorithms are given to illustrate how curse of dimensionality in stochastic dynamic programming can be alleviated or avoided.

VII. ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation Grant DMS-02-07081 in Computational Mathematics. Any views expressed are those of the author and do not necessarily reflect those of the NSF. Computational research and education has been also supported by a large number of national and local computing laboratories.

VIII. REFERENCES

[1] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[2] D. S. Bernstein, "A Student's Guide to Classical Control," *IEEE Control Systems Magazine*, vol. 17, pp. 96-100, August 1997.

[3] G. E. Forsythe, M. A. Malcolm and C. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[4] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, Springer-NY, New York, NY, 2003.

[5] F. B. Hanson, "Bioeconomic model of the Lake Michigan alewife fishery," *Can. J. Fish. Aquat. Sci.*, vol. 44, suppl. 2, pp. 298-305, 1987.

[6] F. B. Hanson, "Stochastic Dynamic Programming: Advanced Computing Constructs," *Proceedings of 28th IEEE Conference on Decision and Control*, vol. 1, pp. 901-903, 1989.

[7] F. B. Hanson., "Computational dynamic programming on a vector multiprocessor," *IEEE Trans. Automatic Control*, vol. 36(4), pp. 507-511, April 1991.

[8] F. B. Hanson, "Computational Stochastic Dynamic Programming" in *Stochastic Digital Control System Techniques*, within series *Control and Dynamic Systems: Advances in Theory and Applications*, vol. 76, C. T. Leondes (Editor), Academic Press, New York, NY, pp. 103-162, April 1996.

[9] F. B. Hanson, "Local Supercomputing Training in the Computational Sciences Using Remote National Centers," *Future Generation Computer Systems: Special Issue on Education in the Computational Sciences*, to appear, 21 pages in ms., 13 January 2003.

[10] F. B. Hanson, *CAUTION: Cramer's Rule is Computationally Expensive*, `http://www.math.uic.edu/~hanson-/cramers.html`

[11] F. B. Hanson, *Basic Floating Point Representation*, `http://www.math.uic.edu/~hanson/mcs471-/FloatingPointRep.html`

[12] F. B. Hanson, *Control Tools*, `http://www.math.uic.edu-/~hanson/math574/math574tools.html`

[13] F. B. Hanson, and J. J. Westman, "Stochastic Analysis of Jump–Diffusions for Financial Log–Return Processes," *Stochastic Theory and Control, Proceedings of a Workshop*, held in Lawrence, Kansas, October 18-20, 2001, *Lecture Notes in Control and Information Sciences*, B. Pasik-Duncan (Editor), Springer-Verlag, New York, pp. 169-184, July 2002.

[14] F. B. Hanson, and J. J. Westman, *Applied Stochastic Processes and Control for Jump-Diffusions: Modeling, Analysis and Computation*, SIAM Books: Advances in Design and Control Series, 194 page draft, September 2003.

[15] D. E. Kirk, *Optimal Control Theory: An Introduction*, Prentice-Hall, Englewood Cliffs, NJ, 1970.

[16] H. J. Kushner and P. Dupuis, *Numerical Methods of Stochastic Control Problems in Continuous Time*, Springer-Verlag, New York, 2001.

[17] W. C. Messner and D. M. Tilbury, *Control Tutorials for MATLAB And Simulink: User's Guide*, Addison-Wesley Publ. Co., 2002.

[18] R. M. Murray, et al., "Future Directions in Control in an Information-Rich World: A Summary of the Report of the Panel on Future Directions in Control, Dynamics, and Systems," *IEEE Control Systems Magazine*, vol. 23 (2), pp. 20-33, April 2003.

[19] Pratico, C. J., F. B. Hanson, H. H. Xu, D. J. Jarvis and M. S. Vetter, "Visualization for the management of renewable resources in an uncertain environment," *Proc. Supercomputing '92*, pp. 258-266 & 843, November 1992.

[20] Westman, J. J. and F. B. Hanson, "The LQGP problem: a manufacturing application," *Proceedings of 1997 American Control Conference*, vol. 1, pp. 566-570, June 1997.

[21] Westman, J. J. and F. B. Hanson, "Nonlinear State Dynamics: Computational Methods and Manufacturing Application," *International Journal of Control*, vol. 73, pp. 464-480, April 2000.