Computational Stochastic Control: Basic Foundations, Complexity and Techniques[1]

Floyd B. Hanson[2]

Abstract

Much research in control systems is purely mathematical, but advances in stochastic control problem solving can be used beyond the limits of where theoretical mathematics can help. Theoretical and computational mathematics are complementary. Computation is important where the problem is mathematically intractable, of high dimension as in stochastic dynamic programming or solving the problem is urgent as in competitive financial engineering 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 new and better algorithms, the theoretical side of computation. 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 the postgraduate job is uncertain in the current world.

In this expository paper, a selection of basic computational considerations, high performance computers and some useful algorithms are surveyed. Some of the computational methodology in both algorithms and advanced computers arose from the author's own research. Much of the knowledge 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, if done properly, forms the other half of mathematics, beyond the topics of regular or traditional mathematics courses. Computation has its own algebra, oriented to finite precision arithmetic, and its own analysis that is numerically oriented. The view is that of an applied analyst and computational control scientist explaining the field to those with a regular mathematics background.

[1] Work supported in part by the National Science Foundation Grant DMS-02-07081 in Computational Mathematics at the University of Illinois at Chicago. Computational research and education has been also supported by Argonne National Laboratory, Cornell Theory Center, Los Alamos National Laboratory, National Center of Supercomputing Applications, Pittsburgh Supercomputing Center, San Diego Supercomputing Center, UIC Academic Computing and Communication Center UIC Electronic Visualization Laboratory, and UIC Laboratory for Advanced Computing. This paper has been submitted as an Invited Poster/Interactive Paper in a Control Education Session for the IEEE Conference on Decision and Control, 9-12 December 2003 in Maui, Hawaii.

[2] 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

1. Introduction

There are great demands on students learning about stochastic analysis and stochastic control theory, since they 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 need to be made more accessible to students interested in applications and earlier than usual, as early as entry-level graduate studies. Much of the material can be developed from more concrete analysis than from the more 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 pure analysis.

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. 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.

The introductory study of stochastic control needs to be on 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 regular chain rule for deterministic, continuous processes. Many of the author's students are applied mathematics, computational science and engineering, so that a concrete presentation to these diverse and applied type of students. An applied stochastic processes and control of jump-diffusions text [29] is currently being prepared.

2. 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 structures that are a part of computational mathematics than the idealistic structures that are part of the regular mathematics. Also, there are differing computational goals such that speed and efficiency are important considerations, but they are not in the regular mathematics courses. For instance, in regular mathematics there is an interest in convergence to an answer, but in computation just convergence is not good enough, since both the rate of convergence and convergence in finite time, due to finite stopping tolerances, are critical. The accuracy of an approximate answer is also important. Thus, there are important differences in mathematical structure, efficiency and accuracy. The intent of this section is to present some of the basic ideas of the foundations of numerical analysis to those more familiar with the foundations of mathematical analysis and who are not very aware of the computational side of mathematics.

2.1. Finite Precision Representation

The most basic difference in mathematical structure is due to the computer representation of real numbers. The mathematics of this different number system attracted the attention of both pure and applied mathematicians about four decades ago with Willy Prager and Philip Davis at Brown University, Peter Lax at New York University, George Forsythe at Stanford University and others. Much of their influence was by way of unpublished course notes. Computer representation is quite different from the continuous representation in regular analysis and this difference is fascinating. For the representation of real numbers on computers, the floating point number system is finite, i.e., $p < \infty$ digits, and is 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, and zero is an exception with special representation. This normalization avoids the presence of leading zeros (wasting storage) and the exponent allows the decimal point to float in the representation preserving the order of magnitude of a number. Thus, floating point numbers can be very small or very large in magnitude.

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

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

where $b = 2$, 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 j binary digit in the exponent, q is the number of binary digits in the exponent, $\pm$ is the sign of the fraction and $\pm'$ is the sign of the exponent. The actual computer storage is much more complicated, allowing for special numbers like *NAN*, or 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 would be represented in output in scientific notation as

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

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 e is the computer scientific notation symbol marking the beginning of the exponent. Conversion approximations going back and forth between binary (base 2) and decimal (base 10) provide another difference between computational mathematics and regular mathematics, however slight. More information on floating point representation can be found in numerical texts [10, 30] or on-line [32, 23].

2.2. Machine Epsilon

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

Definition:

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

Thus, *maceps* is the smallest positive floating point number next to zero.

Theorem: For floating point representation computations with rounding in base b and precision p,

$$\textit{maceps} = \frac{1}{2} b^{1-p}.$$

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

$$\textit{maceps} \quad = \quad 2^{-53} \simeq 1.11022 \times 10^{-16} = 1.11022e - 16.$$

As a result of this finite precision, familiar algebraic laws such as the distributive law no longer hold, except approximately.

2.3. Catastrophic Cancellation

If the limits of the floating point representation are unknown to the user, then mistakes, large and small, can happen. If two calculations are performed by different orders of operations,

then they can lead to different results, even if the two different orders are mathematically equivalent. However, the results should 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 leading or most significant digits are cancelled in floating point calculations.

2.4. Derivatives

The best example is the simulated computation of a derivative by Newton's quotient,

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

by taking h sufficiently small, but not too small. If h is too small then the machine epsilon property is operative,

$$\begin{aligned} \text{float}[x+h] &= \text{float}[x], \ 0 < |h| < \textit{maceps}/|x|, \\ &\quad \text{sgn}(h) = \text{sgn}(x), \ |x| > 0, \end{aligned}$$

and the numerator is zero in floating point precision. Hence, the instruction in the regular calculus or analysis class to *take h (or ϵ) as small as you please* would be wrong in floating point arithmetic. In general, the smaller the step size h, the greater are floating point errors 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, excluding problems with catastrophic cancellation.

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

2.5. Symbolic Computation: A Disclaimer

On many computers, symbolic computing systems such as Maple™ [2] and Mathematica™ [2] 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. Even when the symbolic system is used for strictly numerical purposes, the results are not as predictable as the results from the IEEE 754 floating point standard [31]. Even though symbolic computing systems are based on the same computing platforms, the symbolic floating point scheme has been characterized as strange, to say the least. (See the on-line notes of William Kahan [32] at the University of California at Berkeley, the primary advocate for floating point and general computational correctness.)

2.6. Inefficient Linear Algebra Methods

Several linear algebra methods taught in regular linear algebra courses are fine for use in theory, but very inefficient 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 pivoting of both rows and columns is necessary to make the method robust with regard to accuracy. The necessity of these refinement can easily be demonstrated by a few floating point counter examples. 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 any number of right hand side vectors $\vec{b}^{(k)}$.

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 qualifications for practical applications. It can be shown by the properties of the exponential function series and Stirling's asymptotic formula for the factorial function $n!$ that the computational complexity in terms of floating point operations for computing the necessary determinants to solve an nth order system by Cramer's rule is of exponential order. That is, the number of floating point operations for the full solution is asymptotic to $e * (n+1)!$ for the full solution [22].

Gauss-Jordan Elimination: In the theoretical linear algebra course the diagonalization form of Gaussian elimination is taught, but this is twice as costly as Forward Gaussian Elimination with Back Substitution. For large scale systems or for production problems in industry, this greater cost would be unwarranted.

3. Computational Deterministic Control versus Computational Stochastic Control

This paper is about about computational stochastic control, but computational deterministic control will be mentioned briefly to provide some contrast to stochastic control problems.

3.1. Inefficient Linear Algebra Methods

For deterministic control problems (see [42, 33]), many can be cast as systems of ordinary differential equations so many standard numerical methods can be used. For example, if $\vec{X}(t)$ is the state n-vector of the deterministic system in continuous time t, $\vec{U}(t)$ is the control m-vector, the differential equation for the dynamics is

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

where $\vec{F}(\vec{x}, \vec{u}, t)$ is called the plant function which could be nonlinear, and the object is to achieve the minimal cumulative

running costs $C(\vec{x},\vec{u},t)$ on (t_0,f_f) plus terminal cost function $Z_f(\vec{x},t)$ in the objective,

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

In the Hamiltonian formulation [33], the optimization is combined with the dynamics and is performed on the Hamiltonian function

$$H(\vec{X}(t),\vec{U}(t),\vec{\lambda}(t),t) \equiv C(\vec{X}(t),\vec{U}(t),t) + \vec{\lambda}^T(t)\vec{F}(t)(\vec{X}(t),\vec{U}(t),t), \quad (3)$$

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

$$\left(\frac{\partial H}{\partial \vec{\lambda}}\right)^* = \frac{d\vec{x}^*(t)}{dt},\ \left(\frac{\partial H}{\partial \vec{x}}\right)^* = -\frac{d\vec{\lambda}^*(t)}{dt},\ \left(\frac{\partial H}{\partial \vec{u}}\right)^* = \vec{0}, \quad (4)$$

forming a set of three vector ordinary differential equations for the optimal trajectory under the optimal control $\vec{u}^*(t)$. The side conditions depend on the application and are tabulated in Kirk [33]. The state vector $\vec{x}^*(t)$ 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 iterations are required until the solution $(\vec{x}^*(t),\vec{u}^*(t),\vec{\lambda}^*(t))$ converges to some 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 into $\vec{x}^*(T_j) = \vec{X}_j = [X_{i,j}]_{n\times M}$ 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.

In addition, MATLAB™ [41] has a good number of control Toolboxes to handle problems. There are also several good on-line tutorials available, such as Tilbury and Messner's [47, 40] *Control Tutorials for MATLAB And Simulink*. Bernstein has several insightful essays on *Classical Control* [6] and *Concrete Control Education* [7]. A longer list of on-line control information is given in the author's *Control Tools* web-page [24]. There are early surveys on computational methods for optimal control problems by Larson [37], Dyer and McReynolds [9], and Polak [43] for instance.

3.2. Computational Stochastic Control

By contrast, computation for stochastic control is often quite different and can concern partial differential equations [38, 46]. Numerical methods using approximations of Markov chains for stochastic control problems in continuous time have been throughly explored by Kushner [34] and Kushner and Dupuis [35]. The author has a chapter on high performance computing techniques for stochastic dynamic programming [20]. Many of these stochastic control computational methods are formulated in terms of dynamic programming, originally formulated by Bellman [4]. However, the PDE formulation leads to exponential complexity numerically, called the *Curse of Dimensionality* [5].

For the case where the added stochasticity is a stochastic diffusion, the dynamical system is called a stochastic differential equation (SDE) and can have the form in continuous time as

$$d\vec{X}(t) = \vec{F}(\vec{X}(t),\vec{U}(t),t)dt + \vec{G}(\vec{X}(t),t)d\vec{W}(t) + \vec{J}(\vec{X}(t),t)dP(t), \quad (5)$$

where $\vec{G}(\vec{x},t)$ is the volatility array function and $d\vec{W}(t)$ is the vector differential of the standard diffusion or Wiener process with mean $\vec{0}$ and covariance $I_r dt$, the r component processes are pairwise independent for simplicity, I_r is the $r\times r$ identity matrix, and $\vec{J}(\vec{X}(t),t)$ is the Poisson random jump amplitude associated with the Poisson differential process $dP(t)$ having a common mean and variance of λdt. It is assumed that $dP(t)$ and J are independent except that $dP(t)$ must jump to trigger the random amplitude. They are both independent of $dW(t)$. Note that $d\vec{W}(t)$ and $dP(t)$ should be written as differentials rather than as derivatives since the covariances imply that $\vec{W}(t)$ is not a smooth function and $P(t)$ is not continuous in this ideal model. The object in the stochastic control case is taken to be essentially the same as in the deterministic case (2) for the purposes of comparison. The main changes are that the cost objective is now a functional of the the diffusion $\vec{W}(t)$ and jump $P(t)$ processes that come with the dependence on the state $\vec{X}(t)$ process and that the initial time t_0 is replaced by t in order to vary time, 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 or mean needs to be taken to smooth out the stochastic or random fluctuations to form a more well-defined optimization. Then the optimal or minimal, expected total cost at time t is

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

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 [4] and the chain rule for Markov stochastic processes (the discontinuities of $P(t)$, the nonsmoothness of $\vec{W}(t)$ and the infinitesimal covariances introduce extra terms not in the chain rule of deterministic, regular calculus), the *Partial Differential Equation of Stochastic Dy-*

namic Programming is

$$\begin{aligned} 0 &= \frac{\partial v^*}{\partial t}(\vec{x},t) + \frac{1}{2}\mathrm{T}race\left[GG^T\nabla_x\left[\nabla_x^T\left[v^*\right]\right]\right] \\ &+ \min_u\left[C(\vec{x},\vec{u},t) + \vec{F}^T(\vec{x},\vec{u},t)\nabla_x\left[v^*(\vec{x},t)\right]\right. \quad (6) \\ &\left.\lambda\int_Q\left[v^*(\vec{x}+\widehat{J}(\vec{x},q,t),t) - v^*(\vec{x},t)\right]\phi_{\mathcal{Q}}(q)dq\right], \end{aligned}$$

where $0 < t < t_f$, q is the space-time Poisson amplitude mark variable on the mark space $\mathcal{Q}$, Q is the corresponding the mark random variable with density $\phi_Q(q)$, and $\widehat{J}(\vec{x},q,t)$ is the realized vector jump amplitude. The final condition is

$$v^*(\vec{x},t_f) = Z(\vec{x},t_f).$$

This is no regular partial differential equation (PDE), due to the presence of the minimization operation, which means that the output of the dynamic program is not just the value of the optimal cost $v^*(\vec{x},t)$, but also the optimal feedback control vector $\vec{u}^*(\vec{x},t)$. Another difference is that the last term in (6) is the Poisson distributed jump integral which leads to global dependence unlike the point-wise or local dependence of the optimal cost gradient $\nabla_x[v^*(\vec{x},t)]$.

It is this state-time vector valued functional form of the solution set, $\{v^*(\vec{x},t),\vec{u}^*(\vec{x},t)\}$, given independent variables $\vec{x}$ and t, that makes the stochastic case quite different from the deterministic case of vector functions of time. If time is fixed at a single discrete value T_k, where $k = 1:N_t$, the independent discretization of the n-dimensional state vector $\vec{x}$ is replaced by $\vec{X}_{\vec{j}} = [X_{i,j_i}]_{n\times 1}$ where $\vec{j} = [j_i]_{n\times 1}$, $j_i = 1:N_x$ for $i = 1:n$ and N_x is the number of state nodes, simply taken to be the same for each component (otherwise, N_x could be the geometric mean of n node counts N_i). However, $\vec{X}_{\vec{j}}$ only represents one point in state space and there are a total N_x^n numerical nodes or points in n dimensions. Thus, total numerical representation optimal cost $v(\vec{x},T_k)$ is

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

per time step k, so that the computational complexity is

$$CC(N_x,n) = O(N_x^n) = O(\exp(n\ln(N_x))), \quad (7)$$

exponential in the dimension with an exponent coefficient depending on the logarithm of the common number of nodes, symbolizing the exponential computational complexity of Bellman's 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 dependency. Further, for second order finite difference errors, the total error will be

$$E_T(N_x,n) = O(N_x^{-2}). \quad (8)$$

So even if the order of the complexity is fixed, i.e., $N = N_x^n$ is a constant, then $N_x = N^{1/n}$ and

$$E_T(N_x(N),n) = O\left(N^{-2/n}\right) \to O(1) \quad (9)$$

as $n \to +\infty$ for fixed N and accuracy, 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. Many of these are covered in the author's computational stochastic dynamic programming chapter [20] or in Kushner and Dupuis' numerical book [35]. Most importantly, as with any PDE, the computational mesh ratio must be selected carefully or a computational PDE method will not converge. In the *Markov Chain Approximation* of Kushner [34, 35] convergence conditions are automatically satisfied by choosing normalizing discretization coefficients to conserve probability of the Markov chain. 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. The method to handle these oscillations is called *Upwinding*, in which the finite difference scheme is selected, forward or backward, in the directions of the gradient of the value function (see Kushner and Dupuis [35] for use in control applications). 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 (6) among the locally dependent diffusion and drift derivative terms. Hence, the numerical approximations may require interpolation and integration techniques (see for instance our papers [49] and other jump or jump-diffusion papers). Still another problem is the proper handling of boundary conditions for stochastic processes which are quite different than those in the deterministic case due to things like boundary over-shoots, but Kushner and Dupuis [35] have developed a systematic method using an auxiliary stochastic process to enforce the proper boundary behavior for the original stochastic processes.

4. 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 [15] of high performance computing. The availability of high performance vector supercomputers and massively 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 computer and workstation are making parallel computers a commodity item [21] reducing costs by a factor of ten or more for large scale systems. Advances in algorithms have also played a comparable role.

The 1988 report of the Fleming panel on the *Future Direc-*

tions in Control Theory [11] confirmed the need for advanced scientific computing, both parallelization and vectorization, in control problems. The *National Computing Initiative* [1] called stochastic dynamic programming computationally demanding. The 2002 Murray panel on the *Future Directions in Control, Dynamics, and Systems* [12] places a lot more emphasis on control education and computation plays an integral role in the investigation of control problems throughout the report.

Another effort in this area has been in France. Quadrat and his coworkers [3] at INRIA developed an expert system, called *Pandora*, and produced a multitude of results for stochastic differential equations with Gaussian noise, provided that discounting is constant and the problem can be transformed to a stationary one. It was based on the numerical methods of Kushner [34]. However, it appears that it is no longer available, disappearing when the computer workstation it was developed on disappeared.

4.1. 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 disastrous environments [16]. The control in these models is the species harvesting effort, the harvest rate per unit species population. These models used Poisson processes to simulate disastrous processes, so were much more complicated analytically than the locally dependent and smooth stochastic diffusion processes discussed above since the jumps lead to globally dependent functional jump terms. However, it was the multi-dimensional resources problem that caused time consuming computations and made local mainframe computing resources inadequate, requiring the need for advanced computing facilities. The large scale computational demands required new and more efficient numerical methods [19] that were quite different and sometimes contrary to the standard methods for computation on serial computers.

Starting with a moderate number of parallel processors and Cray vector supercomputers with a few processors, solving stochastic dynamic programming problems in several dimensions became feasible with reasonable turn-around time. Larger Cray vector supercomputers permitted the solution in even higher dimensions, but the size of the problems that could be run on the massively parallel Thinking Machines CM-2 and CM-5 surpassed those on the Crays. Much of the effort on these machines was making the vector of parallel code more efficient by reducing data dependencies, making the code more transparent to the compilers, eliminating unnecessary overhead and other techniques. Many of these advanced computing techniques are summarized in the chapter on computational stochastic dynamic programming [20], so the reader is referred there for more details. An important spin-off benefit was starting a course on parallel processing and supercomputing in 1986 [18, 21] as technology transfer from this large-scale control research, and also to help train the author's graduate students in this computational stochastic control research. The course related papers [18, 21] give even more detail on advanced parallelization and vectorization techniques. The current class is using low cost, large-scale parallel clusters of networked computer systems [21].

4.2. Advanced Data Structures

Further, advanced parallel and vector optimizations on advanced computers are dependent on the data structure. For certain applications, the code runs faster if arrays are accessed the way they are stored in memory, e.g., Fortran arrays are stored linearly by columns while C code is stored linearly by rows, in absence of memory partitioning. In some early applications [17], the author developed a vector data structure to represent the full n-dimensional state space to make it relatively easy to change code from one dimension to another and also to vastly increase the work load balance of vector registers and parallel processors by maintaining a large pool of work. This technique was also used later by Kushner and Jarvis [36].

4.3. 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 solution functions may depend significantly on k parameters, some of which may be associated with stochastic processes like volatility, jump rates and jump amplitudes. Hence, displaying the resulting solution can be challenging once the total dimension of the system, counting both dependent and independent variables and parameters, goes beyond three dimensions (the practical limit for projection onto two-dimensional media). For this reason, our research group has developed a refinement and real implementation of multi-dimensional control problem output called *I/O view* [44, 25] or 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. This system was originally designed for optimal resource management so that the resource manager could optimally select values of variables to manage the resource system. However, the visualization system can be used to visualize almost any multi-dimensional output.

5. Algorithmic Advances

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

5.1. Canonical Models

Many control problems can be reduced in analytical and computational complexity if a canonical or standard model is appropriate and leads to a separation of variables type of decom-

position.

LQGP Problem: One canonical problem, often used in stochastic control applications, is the LQG problem which has linear (L) dynamics, quadratic (Q) costs and Gaussian (G) or diffusion noise. The extension of the canonical problem to Poisson jump processes, which preserves and generalizes the Markov nature of the stochastic diffusion of the LQG, is called the LQGP problem. It is also known as the JLQG problem, emphasizing the jump character. We have developed refinements and computational implementation of the LQGP problem in [48] 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 vector, but with coefficients that implicitly depend on time,

$$\begin{aligned} v^*(\vec{x},t) &= \frac{1}{2}\vec{x}^T S(t)\vec{x} + \vec{D}^T(t)\vec{x} + E(t), \qquad (10) \\ \vec{u}^*(t) &= -K(t)\left[S(t)\vec{x} + \vec{D}(t)\right], \end{aligned}$$

in the case of unconstrained control, where the gain matrix $K(t)$ is derivable from the original LQGP quadratic cost coefficient and linear control coefficient. Hence, only time coefficients need to be determined. These coefficients are the matrix $S(t)$, the vector $\vec{D}$ and the scalar $E(t)$. They satisfy a uni-directionally coupled set of matrix ordinary differential equations. These equations are nontrivial [48] due to the complexity of the jump processes used in the application. The benefit is essentially a reduction in the computational curse of dimensionality since the form of the state space dependence is explicitly given.

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 which is the state variable in the problem. The optimal value of the investment portfolio separates into a known function of the wealth, the CRRA utility, and an unknown function of time, $F(t)$, i.e.,

$$v(w,t) = \mathcal{U}(w)F(t), \qquad (11)$$

again avoiding the dimensionality and additional dependence due to the state space. The optimal control variables are the stock fraction $u^*(w,t)$ and the consumption $c^*(w,t)$. These have complicated forms, especially for jump-diffusions and these forms can be found in our computational jump-diffusion finance papers [26, 27, 28]. The original portfolio model is due to Merton [39].

5.2. Markov Chain Approximations

Kushner and co-workers [34, 35, 36] have described many numerical approaches to stochastic control, with special emphasis on the well-developed *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 or stage and h is the scale of the state change, like Δx. The interpolation step $\Delta t(n,h)$ is chosen so that the Markov chain is "locally consistent" with the stochastic dynamic process in continuous time. When applied to the approximation of the control problem value function, the coefficients are selected according to transition probabilities between chain stages or steps of the Markov chain by proper choice of the interpolation step $\Delta t(n,h)$ for the application problem. Weak convergence follows from this construction of the Markov chain and, as already mentioned, numerical convergence follows. A method comparison of the Markov chain approximation of Kushner and co-workers with the PDE finite difference methods of Hanson and co-workers if given in [20]. See the comprehensive book of Kushner and Dupuis [35] for the complete details.

5.3. 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 [20], finite element [8], or Markov chain approximation methods [35], since these methods suffer from the curse of exponential complexity (7) or diminishing accuracy (9). On the other hand, if we could cast the problem as an integral over the state space and take a random state space sample using a pseudo random number generator (there being no such thing as a real random number, see [45] for instance), then the *Pseudo Monte Carlo Approximation*, or *Monte Carlo Approximation* for the N-point sample average approximates the integral average over state volume V. That is,

$$\int_V f(\vec{x})d\vec{x} \;\simeq\; \langle\, f\,\rangle_N \;\equiv \frac{1}{N}\sum_{i=1}^{N} f(\vec{x}_i). \qquad (12)$$

The average probable error is

$$E_N = O\left(1/\sqrt{N}\right), \qquad (13)$$

according to the Central Limit Theorem, independent of dimension n, assuming that the picks in the sample are identically distributed normal random variables and N is sufficiently large. Compared to the finite difference error in (9), the Monte Carlo approximation will be more accurate and faster converging for larger dimensions, provided the number of sample points can be kept reasonable.

There are many variations of the Monte Carlo method. One is the *Quasi Monte Carlo Method*, using a quasi random number generator [45]. It uses a deterministic and systematic covering of state space, i.e., even more deterministic than the pseudo random number generator using linear congruential generator simulations, but has an error that can approach

$$O\left(\ln(N)/N\right),$$

a smaller order than that of (13), since $\ln(N) \ll \sqrt{N}$ for $N \gg 1$. Other techniques [45] 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, but there is little stochastic control, other than Black-Scholes type hedging [39]. For instance, see Glasserman and co-workers papers on importance and stratified sampling [13, 14].

6. 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 advanced algorithms are given to illustrate how Bellman's curse of dimensionality in stochastic dynamic programming can be alleviated or avoided.

References

[1] *A National Computing Initiative: The Agenda for Leadership*, H. J. Revecheé (Chair), Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[2] Abell, M L. and J. P. Braselton, *The Maple V Handbook*, Academic Press, New York, NY, 1994.

[3] Akian, M., J. P. Chancelier and J. P. Quadrat, "Dynamic Programming Complexity and Applications," *Proceedings of 27th IEEE Conference on Decision and Control*, vol. 2, pp. 1551-1558, 1988.

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

[5] Bellman, R. E., *Adaptive Control Processes: A Guided Tour*, Princeton University Press, Princeton, NJ, 1961.

[6] Bernstein, D. S., "A Student's Guide to Classical Control," *IEEE Control Systems Magazine*, vol. 17, pp. 96-100, August 1997; *URL: http://www.engin.umich.edu/dept/aero-/people/faculty/bernstein/classic_control.pdf*, Aerospace Engineering, University of Michigan, Ann Arbor, MI, 8 pages.

[7] Bernstein, D. S., "Concretizing Control Education, *URL: http://www.engin.umich.edu/dept/aero/people/faculty-/bernstein/Control Education.pdf*, Aerospace Engineering, University of Michigan, Ann Arbor, MI, 8 pages.

[8] Chung, S.-L., F. B. Hanson and H. H. Xu, "Parallel Stochastic Dynamic Programming: Finite Element Methods," *Lin. Alg. Applic.*, vol. 172, pp. 197-218, July 1992.

[9] Dyer, P. and S. R. McReynolds, *The Computation and Theory of Optimal Control*, Academic Press, New York, NY, 1970.

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

[11] Future Directions in Control Theory Panel, *Report of the Panel on Future Directions in Control Theory: A Mathematical Perspective*, W. H. Fleming, chairman, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

[12] Future Directions in Control, Dynamics, and Systems Panel, *Control in an Information Rich World: Report of the Panel on Future Directions in Control, Dynamics, and Systems*, R. M. Murray, chair, textitURL: http:-//www.cds.caltech.edu/∼murray/cdspanel/, California Institute of Technology, April 2002.

[13] Glassman, P. and P. Heidelberger and P. Shahabuddin, "Variance Reduction Techniques for Estimating Value-at-Risk," *Management Science*, vol. 46(10), pp. 1349-1364, October 2000.

[14] Glassman, P. and P. Heidelberger and P. Shahabuddin, "Portfolio Value-at-Risk with Heavy-Tailed Risk Factors," *Math. Finance*, vol. 12, pp. 239-269, 2002.

[15] *Grand Challenges: High Performance Computing and Communications: The FY 1993 U.S. Research and Development Program*, Federal Coordinating Council for Science, Engineering, and Technology, Committee on Physical, Mathematical, and Engineering Sciences, c/o National Science Foundation (CISE), Washington, DC, 1993.

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

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

[18] Hanson, F. B., "A Real Introduction to Supercomputing: A User Training Course," in *Proceedings of Supercomputing '90*, pp. 376-385, November 1990,

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

[20] Hanson, F. B., "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.

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

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

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

[24] Hanson, F. B., *Control Tools*, URL: http:-//www.math.uic.edu/∼hanson/math574/math574tools.html

[25] Hanson, F. B., C. J. Pratico, M. S. Vetter, and H. H. Xu, "Multidimensional Visualization Applied to Renewable Resource Management," *Proc. Sixth SIAM Conference on Parallel Processing for Scientific Computing*, vol. 2, pp. 1033-1036, March 1993.

[26] Hanson, F. B., and J. J. Westman, "Optimal Consumption and Portfolio Policies for Important Jump Events: Modeling and Computational Considerations," *Proceedings of 2001 American Control Conference*, June 2001, pp. 4556–4561.

[27] Hanson, F. B., 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.

[28] Hanson, F. B., and J. J. Westman, "Optimal Consumption and Portfolio Control for Jump-Diffusion Stock Process with Log-Normal Jumps," *Proceedings of 2002 American Control Conference*, pp. 4256-4261, May 2002.

[29] Hanson, F. B., and J. J. Westman, *Applied Stochastic Processes and Control for Jump-Diffusions: Modeling, Analysis and Computation*, SIAM Books: Advances in Design and Control Series, 68 page proposal, contract accepted, November 2002.

[30] Heath, M. T., *Scientific Computing: An Introductory Survey*, McGraw-Hill, New York, 2002.

[31] Kahan, W., "William Kahan Homepage," *URL: http:-//http.cs.berkeley.edu/∼wkahan*, EECS, University of California at Berkeley.

[32] Kahan, W., " Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic," *URL: http://http.cs.berkeley.edu/∼wkahan/ieee754status-/ieee754.ps*, EECS, University of California at Berkeley.

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

[34] Kushner, H. J., "Numerical Methods for Stochastic Control in Continuous Time," *SIAM J. Control and Optimizations*, vol. 28, pp. 999-1048, 1990.

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

[36] Kushner, H. J. and D. J. Jarvis, "Large-Scale Computations for High Dimension Control Systems," *Proceedings of 33rd IEEE Conference on Decision and Control*, vol. 1, 6 pages, 1994.

[37] Larson, R. E., "A Survey of Dynamic Programming Computational Procedures," *IEEE Transactions on Automatic Control*, vol. AC-16, pp. 767-774, 1967.

[38] Lewis, F. L., *Optimal Estimation with an Introduction to Stochastic Control Theory*, John Wiley, New York, NY, 1986.

[39] Merton, R. C., *Continuous-Time Finance*, Basil Blackwell, Cambridge, MA, 1992.

[40] Messner, W. C. and D. M. Tilbury, *Control Tutorials for MATLAB And Simulink: User's Guide*, Addison-Wesley Publ. Co., 2002. (See also *URL: http://www.engin.umich.edu-/group/ctm/ or http://www.engin.umich.edu/group/ctm/*)

[41] Moler, C., et al., *Using MATLAB*, vers 6., Mathworks, Natick, MA, 2000.

[42] Nise, N. S., *Control Systems Engineering*, John Wiley, New York, NY, 2000.

[43] Polak, E., "An Historical Survey of Computational Methods in Optimal Control," *SIAM Review*, vol. 15, pp. 553-584, 1973.

[44] 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, color plates p. 843, November 1992.

[45] Press, W. H., S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C++: The Art of Scientific Computing*, 2nd Edition, Cambridge University Press, Cambridge, UK, 2002.

[46] Stengel, R., *Stochastic Optimal Control: Theory and Application*, Dover Publications, New York, NY, 1994.

[47] Tilbury, D. M. and W. C. Messner, "Control Tutorials for Software Instruction over the World Wide Web," *IEEE Trans. Automatic Control*, vol. 42(4), pp. 237-246, November 1999.

[48] 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.

[49] 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.

[50] Wolfram, S., *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley, Reading, MA, 1988.