Dynamic Programming, Control, and Computation[*]

Floyd B. Hanson
Department of Mathematics, Statistics, and Computer Science,
University of Illinois at Chicago, and Department of Mathematics,
University of Chicago, Chicago, IL
hanson@math.uic.edu

May 27, 2010

Abstract

The presentation in this chapter is in the formal manner of classical applied mathematics and probability in order to focus on the methods and their implementation. In Section 1, a fairly general model of stochastic dynamic programming in continuous time is outlined. In Section 2, canonical forms, such as a linear dynamics and quadratic cost model in control, that lead to a large reduction in computational effort are given or discussed. In Section 3, finite difference partial differential equation methods are given that are suitable for approximately solving nonlinear Bellman dynamic programming equations. Alternatively, Markov chain approximation probabilistic methods which systematically justify the stability and weak convergence of the approximating Markov chain are summarized in Section 4. In the last section, there is a brief summary and directions to some other approaches.

Keywords: dynamic programming; finite differences; Markov chain approximations, curse of dimensionality

1 Dynamic Programming in Continuous Time

Dynamic programming in application areas of operations research often arise from optimal objectives such as to minimize costs, maximize profits or maximize the utility of wealth and its consumption. These problems invariably are formulated in an uncertain environment since deterministic problems rarely occur without the some sort of random perturbation. Hence, the optimization will necessarily be over an expected or mean value, else an optimum will not be well-posed. Assuming that the perturbing noise is Markovian, then Bellman's [2] principle of optimality fits both stochastic and deterministic dynamic environments, with just some extra analysis in the stochastic case. As a result of taking expectation before optimization,

[*]Preprint to appear in Wiley Encyclopedia of Operations Research and Management Science, edited by James J. Cochran, John Wiley & Sons, 7 pages, 2010.

the equation of dynamic programming is a deterministic partial integro-differential equation (PIDE). See Hanson [3, 4] or Kushner and Dupuis [6] for more details.

Assuming that the objective is a minimal expected cost problem and that the minimum is unique, then the minimum cost [4], starting at any time t in the horizon $[t_0, t_f]$, is

$$\begin{aligned} v^*(\mathbf{x},t) &\equiv \min_{\boldsymbol{U}(t,t_f]}\left[\mathop{\mathrm{E}}_{(\boldsymbol{W},\boldsymbol{P})(t,t_f]}\left[V[\mathbf{X},\mathbf{U},t_f](\mathbf{x},t)\middle|\mathbf{X}(t)=\mathbf{x},\mathbf{U}(t)=\mathbf{u}\right]\right], \\ V[\mathbf{X},\mathbf{U},t_f](\mathbf{x}_0,t) &= \int_t^{t_f} C(\mathbf{X}(s),\mathbf{U}(s),s)ds + Z(\mathbf{X}(t_f),t_f), \end{aligned} \tag{1}$$

where the value of the total cost on $(t,t_f]$ is based on cumulative running cost $C(\mathbf{x},\mathbf{u},t)$ and final cost $Z(\mathbf{x},t_f)$ functions. The $\mathbf{X}(t)$ is the state n_x-vector process on state domain $\mathcal{D}_x$, $\mathbf{U}(t)$ is the control n_u-vector process or optimizing variable on control domain $\mathcal{D}_u$. The set $\{\mathbf{W}(t),\mathbf{P}(t)\}$ comprise fundamental Markov diffusion and jump processes. The diffusion component characterizes the central continuous part and the jump part characterizes the large random, instantaneous, discontinuous rare events, since the diffusion part is insufficient for modeling catastrophic risk events. Obviously, the final optimal value is given by $v^*(\mathbf{x},t_f)=Z(\mathbf{x},t_f)$ for any $\mathbf{x}\in\mathcal{D}_x$ from setting $t=t_f$ in (1).

The stochastic dynamic constraint is the jump-diffusion stochastic differential equation (SDE),

$$\begin{aligned} d\mathbf{X}(t) &= \mathbf{f}(\mathbf{X}(t),\mathbf{U}(t),t)dt + g(\mathbf{X}(t),\mathbf{U}(t),t)d\mathbf{W}(t) + d\mathbf{J}(t), \\ \mathbf{J}(t) &= \int_0^t\int_{\mathcal{D}_q} h(\mathbf{X}(t),\mathbf{U}(t),t,\mathbf{q})\boldsymbol{\mathcal{P}}(\mathbf{dt},\mathbf{dq}), \qquad \mathbf{P}(t)=\int_0^t\int_{\mathcal{D}_q}\boldsymbol{\mathcal{P}}(\mathbf{dt},\mathbf{dq}), \end{aligned} \tag{2}$$

when $t_0\le t\le t_f$ subject to a given initial state $\mathbf{X}(t_0)=\mathbf{x}_0$, $d\mathbf{W}(t)$ is the n_w-dimensional differential diffusion or Wiener process, $\boldsymbol{\mathcal{P}}(\mathbf{dt},\mathbf{dq})$ is the compound n_p-dimensional Poisson random measure that introduces the Poisson random jump times for the time infinitesimal measure $\mathbf{dt}=(t,t+dt]$ and underlying n_p-dimensional random jump amplitude variables in $\mathbf{dq}=(\mathbf{q},\mathbf{q}+d\mathbf{q}]$ when there is a jump. The diffusion process is normally distributed with zero infinitesimal mean $\mathrm{E}[d\mathbf{W}(t)]=\mathbf{0}$, but correlation between components is allowed, $\mathrm{Cov}[d\mathbf{W}(t),d\mathbf{W}^\top(t)]=[\rho_{i,j}(t)]_{n_w\times n_w}dt$, such that $\rho_{i,i}(t)=1$. The usual Poisson process is denoted by $\mathbf{P}(t)$, such that it is Poisson distributed with vector jump intensity $\boldsymbol{\lambda}(t)$, infinitesimal moments $\mathrm{E}[d\mathbf{P}(t)]=\boldsymbol{\lambda}(t)dt=[\lambda_i(t)]_{n_p\times 1}dt$, $\mathrm{Cov}[d\mathbf{P}(t),d\mathbf{P}^\top(t)]=[\lambda_i(t)\delta_{i,j}]_{n_p\times n_p}dt$ and $\mathrm{E}[\mathcal{P}_j(\mathbf{dt},\mathbf{dq})]=\lambda_j(t)dt\phi_{Q_j}(q_j)dq_j$, assuming no correlations between Poisson component processes since simultaneous jumps are unlikely. The underlying jump amplitude random variables Q_j are IID for each component $j=1{:}n_p$ with density $\phi_{Q_j}(q_j)$. Similarly, there are no correlations between Poisson and Wiener processes. The Poisson jumps are by definition instantaneous, so at the instant of a jump there is zero time for continuous change and hence the discontinuous changes are be calculated independent of continuous changes. At the kth jump of jth Poisson component $P_j(t)$ at random time $T_{j,k}$ with random mark $Q_{j,k}$, $\mathbf{X}(t)$ jumps by the amplitude

$$\mathbf{X}(T_{j,k})-\mathbf{X}(T_{j,k}^-)=\widehat{\boldsymbol{h}}_j(\mathbf{X}(T_{j,k}^-),\mathbf{U}(T_{j,k}^-),T_{j,k}^-,q_{j,k})\equiv\left[h_{i,j}(\mathbf{X}(T_{j,k}^-),\mathbf{U}(T_{j,k}^-),T_{j,k}^-,q_{j,k})\right]_{n_x\times 1},$$

where the function $h(\mathbf{x},\mathbf{u},t,\mathbf{q})$ is the $n_x\times n_p$ jump amplitude coefficient array. Like for diffusions, the jump-diffusion approximation consistency conditions require that the increment version of $d\mathbf{X}(t)$ in (2) have conditional infinitesimal mean and variance that are $\mathrm{E}[\Delta\mathbf{X}(t)|\mathbf{X}(t)]=\mathrm{O}(\Delta t)$ and $\mathrm{Var}[\Delta\mathbf{X}(t)|\mathbf{X}(t)]=\mathrm{O}(\Delta t)$. The drift or mean appreciation coefficient function is $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ and the diffusion coefficient function is $g(\mathbf{x}, \mathbf{u}, t)$, both, along with $h(\mathbf{x}, \mathbf{u}, t, \mathbf{q})$, are assumed bounded for computational feasibility and commensurate in multiplication with respect to the other variables. Stronger existence results are available assuming Lipschitz continuity, but some important models such as for stochastic volatility are non-Lipschitzian. For pure deterministic problem, set $g \equiv 0$ and $h \equiv 0$, while for diffusion only noise set $h \equiv 0$.

Assuming minimization and expectation operators are decomposable into factors, then Bellman's principle of optimality [4] is

$$\begin{aligned} v^*(\mathbf{x}, t) = \min_{\boldsymbol{U}(t,t+\delta t]} \Bigg[\mathop{\mathrm{E}}_{(\boldsymbol{W},\boldsymbol{P})(t,t+\delta t]} \Bigg[\int_t^{t+\delta t} C(\mathbf{X}(s), \mathbf{U}(s), s) ds \\ + v^*(\mathbf{X}(t+\delta t), t+\delta t) \Bigg| \mathbf{X}(t) = \mathbf{x}, \mathbf{U}(t) = \mathbf{u} \Bigg] \Bigg], \end{aligned} \tag{3}$$

given some small time-step δt, leads in the limit to the Hamilton-Jacobi-Bellman (HJB) equation of stochastic dynamic programming, keeping only δt-terms,

$$0 = v_t^*(\mathbf{x}, t) + \min_{\mathbf{u}} [\mathcal{H}(\mathbf{x}, \mathbf{u}, t)] \equiv v_t^*(\mathbf{x}, t) + \mathcal{H}^*(\mathbf{x}, t), \tag{4}$$

where the *Hamiltonian* functional is given by

$$\begin{aligned} \mathcal{H}(\mathbf{x}, \mathbf{u}, t) \equiv\ & C(\mathbf{x}, \mathbf{u}, t) + \nabla_{\mathbf{x}}^\top [v^*](\mathbf{x}, t) \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \\ & + \tfrac{1}{2} \mathrm{Tr} \left[\left(g R g^\top \right) (\mathbf{x}, \mathbf{u}, t), \nabla_{\mathbf{x}} \left[\nabla_{\mathbf{x}}^\top [v^*] \right] (\mathbf{x}, t) \right] \\ & + \sum_{j=1}^{n_p} \lambda_j(t) \int_{\mathcal{D}_q} \left[v^* \left(\mathbf{x} + \widehat{\boldsymbol{h}}_j(\mathbf{x}, \mathbf{u}, t, q_j), t \right) - v^*(\mathbf{x}, t) \right] \phi_{Q_j}(q_j) dq_j, \\ R = R(t) \equiv\ & \left[\delta_{i,j} + \rho_{i,j}(t)(1 - \delta_{i,j}) \right]_{n_w \times n_w}, \quad \mathrm{Tr}[A, B] \equiv \textstyle\sum_{i=1}^n \sum_{i=1}^n A_{i,j} B_{i,j}. \end{aligned} \tag{5}$$

Along with appropriate boundary conditions completes the formal specification of the boundary-final value problem, the final value condition and the placement of the time derivative implying that the dynamic programming problem is backward in time. Thus, the HJB equation (4) is in general a functional partial integro-differential equation (PIDE), so possesses properties beyond the usual PDEs in that there is global dependence due to the integral properties whereas the partial derivatives impart only local dependence. In addition, the minimization can introduce nonlinear behavior, as in the case of the often assumed quadratic control costs. These and other properties leads to many computational problems beyond the the usual linear partial differential equation.

For specification of the minimization of the Hamiltonian in (5), a unique minimum and continuous control derivatives up to second order such that the Hessian is positive-definite, i.e., $\nabla_u [\nabla_u^\top [\mathcal{H}]](\mathbf{x}, \mathbf{u}, t) > 0$, are assumed. The critical condition $\nabla_u [\mathcal{H}](\mathbf{x}, \mathbf{u}, t) = \mathbf{0}$ yields the unconstrained minimal or regular control $\mathbf{u}^{(\mathrm{reg})}(\mathbf{x}, t) = \mathrm{argmin}_{\mathbf{u}} [\mathcal{H}(\mathbf{x}, \mathbf{u}, t)]$, the primary in purely mathematical problems, but a preliminary to the constrained optimal control in applications. Real problems have constraints, so assuming hypercube constraints for simplicity, that the constrained optimal control $\mathbf{u}^*(\mathbf{x}, t)$ satisfies the constraints $U_i^{(\min)} \leq u_i^*(\mathbf{x}, t) \leq U_i^{(\max)}$ for $i = 1{:}n_u$ components. Thus,

$$\mathbf{u}^*(\mathbf{x}, t) = \left[\min \left[\max \left[u_i^{(\mathrm{reg})}(\mathbf{x}, t), U_i^{(\min)} \right], U_i^{(\max)} \right] \right]_{n_u \times 1}. \tag{6}$$

2 Computational Reducing Canonical Forms

There are some simplifications that can be used to make the computation of the optimal control simpler. An example is to use a linear drift dynamics and quadratic control cost model, i.e., a LQ model in control only [4], so that

$$\begin{aligned}
\mathbf{f}(\mathbf{x},\mathbf{u},t) &= \mathbf{f}_0(\mathbf{x},t) + f_1(\mathbf{x},t)\mathbf{u},\\
g(\mathbf{x},\mathbf{u},t) &= g_0(\mathbf{x},t), \quad h(\mathbf{x},\mathbf{u},t,q) = h_0(\mathbf{x},t,q),\\
C(\mathbf{x},\mathbf{u},t) &= c_0(\mathbf{x},t) + c_1(\mathbf{x},t)\mathbf{u} + \mathbf{u}^\top c_2(\mathbf{x},t)\mathbf{u},\\
\mathcal{H}(\mathbf{x},\mathbf{u},t) &= \mathcal{H}_0(\mathbf{x},t) + \mathcal{H}_1(\mathbf{x},t)\mathbf{u} + \tfrac{1}{2}\mathbf{u}^\top \mathcal{H}_2(\mathbf{x},t)\mathbf{u},
\end{aligned} \tag{7}$$

where all functions are assumed to be specified and that c_2 is positive definite. Since c_2 and $\mathcal{H}_2$ appear only in quadratic forms they might as well be symmetric, then the regular optimal control is explicit,

$$\mathbf{u}^{(\mathrm{reg})}(\mathbf{x},t) = -c_2^{-1}(\mathbf{x},t)\left(c_1^\top(\mathbf{x},t) + f_1^\top(\mathbf{x},t)\nabla_x[v^*](\mathbf{x},t)\right) \tag{8}$$

and the optimal control $\mathbf{u}^*(\mathbf{x},t)$ is still given by the formula in (6).

In the classical LQ case, i.e., LQ in both control and state space, then both the Hamiltonian and the optimal value v^* are quadratics, $\mathcal{H}$ in both control and state, while v^* can be shown to be a quadratic in state alone, so that the solution for v^* involves solving ODE final value problems only for coefficient functions of time t (see [1, 4] for the well-known details).

3 Finite Difference PDE Computational Methods

Finite differences have the advantage of straightforward application, but with a significant difficulty in convergence if the ratio of the time to space meshes is not sufficiently small. The first step is the discretization, where the time $t \in [t_0, t_f]$ is partitioned into N_t discrete backward time nodes and assuming for simplicity that the spatial variable is one-dimensional ($n_x = 1$) for $x \in [x_0, x_f]$,

$$\begin{aligned}
T_k &= t_f - (k-1)\cdot\Delta T \quad \text{for} \quad k = 1{:}N_t, \quad \text{with} \quad \Delta T \equiv (t_f - t_0)/(N_t - 1),\\
X_j &= x_0 + (j-1)\cdot\Delta X \quad \text{for} \quad j = 1{:}N_x \quad \text{with} \quad \Delta X \equiv (x_f - x_0)/(N_x - 1).
\end{aligned}$$

Next central finite differences (CFD) are used for second order accuracy for spatial derivatives included in the Crank-Nicolson implicit (CNI) method along with half-steps in time to obtain a simple second order accurate form for the time derivative,

$$\begin{aligned}
v^*(X_j, T_k) &\to V_{j,k},\\
v_t^*(X_j, T_{k+0.5}) &\to (V_{j,k+1} - V_{j,k})/(-\Delta T),\\
v_x^*(X_j, T_k) &\to \mathrm{DV}_{j,k} = 0.5(V_{j+1,k} - V_{j-1,k})/\Delta X,\\
v_{xx}^*(X_j, T_k) &\to \mathrm{DDV}_{j,k} = (V_{j+1,k} - 2V_{j,k} + V_{j-1,k})/(\Delta X)^2,\\
u^{(\mathrm{reg})}(X_j, T_k) &\to \mathrm{UR}_{j,k} = -\left(C_{1,j,k} + F_{1,j,k}\mathrm{DV}_{j,k}\right)/C_{2,j,k},\\
u^*(X_j, T_k) &\to \mathrm{US}_{j,k} = u_0(\mathrm{UR}_{j,k}) = \min[\max[\mathrm{UMIN}, \mathrm{UR}_{j,k}], \mathrm{UMAX}],\\
v^*(X_j + h_0(X_j, T_k, q), T_k) &\to \mathrm{VH}_{j,k}(q) \ \ \& \ \ \Lambda_k = \lambda(T_k),\\
F_{i,j,k} = f_i(X_j, T_k) \ \ &\& \ \ C_{i,j,k} = c_i(X_j, T_k) \ \ \& \ \ G_{0,j,k} = g_0(X_j, T_k),\\
F_{j,k} = f(X_j, \mathrm{US}_{j,k}, T_k) \ \ &\& \ \ C_{j,k} = c(X_j, \mathrm{US}_{j,k}, T_k),\\
\mathrm{UMIN} = U^{(\min)} \ \ &\& \ \ \mathrm{UMAX} = U^{(\max)},
\end{aligned} \tag{9}$$

where the LQ assumption has been used for explicitness in the optimal control or otherwise different procedures will be needed [4]. The integral part for the jump amplitude distribution is probably not familiar and requires at least linear interpolation between a post-jump position and nearest nodes, so

$$\mathrm{IVH}_{j,k} \equiv \int_{\mathcal{D}_q} \mathrm{VH}_{j,k}(q)\phi_Q(q)dq \simeq \sum_{i=1}^{N_q} w_i \cdot \left((1-\theta_i)V_{j+\ell_i,k} + \theta_i V_{j+\ell_i+1,k}\right), \tag{10}$$

where the $\{j+\ell_i, j+\ell_i+1\}$ are the nearest neighbor x-nodes corresponding to the post-jump position $X_j + h_0(X_j, T_k, q_i)$ for $i = 1{:}N_q$ mark nodes, the θ_i are the corresponding linear interpolation weights and the w_i are the numerical integration weights assigned to the mark nodes q_i with respect to the density $\phi_Q(q)$. See [4] for additional information, procedures and examples.

The basic CNI step approximates the midpoint in backward time by the average preserving second-order accuracy,

$$\begin{aligned} V_{j,k+1} &\simeq V_{j,k} + \Delta T \cdot \mathcal{H}_{j,k+0.5} \simeq V_{j,k} + 0.5\Delta T \cdot \left(\mathcal{H}_{j,k} + \mathcal{H}_{j,k+1}\right), \\ \mathcal{H}_{j,k} &= C_{j,k} + F_{j,k}\mathrm{DV}_{j,k} + 0.5G_{0,j,k}^2\mathrm{DDV}_{j,k} + \Lambda_k\left(\mathrm{IVH}_{j,k} - V_{j,k}\right). \end{aligned} \tag{11}$$

Although Crank-Nicolson is unconditionally stable and has significant algebraic simplifications for linear diffusions problems, the general complexity of the jump-diffusion problem with quadratic costs here requires iterations to handle the nonlinearities and jump integrals in the optimal value function. An extrapolated-predictor-corrector method has been found to be robust for a variety of applications [4]. An abbreviated pseudo-algorithm for the method is as follows,

```
1. Given constants: $\{N_t, N_x, N_q, N_\gamma, \mathrm{tol}_v, t_f, \mathrm{UMIN}, \mathrm{UMAX}, x_0, x_f\}$;
2. Get $T_k$ $\forall k$; Get $X_j$, $\forall j$;
3. Given functions: $\{c_0, c_1, c_2, f_0, f_1, g_0, h_0, \lambda_0, u_0, Z\}$;
4. Get $V_{j,1} = Z(X_j, t_f)$ $\forall j$; Get $\Lambda_k = \lambda_0(T_k)$, $\forall k$;
5. For $k = 1{:}N_t - 1$ do
6.    Extrapolation Step: $V^{(1)}_{j,k+0.5} = V_{j,1}$; If $k > 1$, then $V^{(1)}_{j,k+0.5} = 0.5(3V_{j,k} - V_{j,k-1})$, endif;
      plus $\mathcal{H}^{(1)}_{j,k+0.5}$, $\forall j$, using (11);
7.    Prediction Step: $V^{(2)}_{j,k+1} = V_{j,k} + \Delta T \cdot \mathcal{H}^{(1)}_{j,k+0.5}$ $\forall j$;
8.    Predictor Evaluation Step: $V^{(2)}_{j,k+0.5} = 0.5\left(V^{(2)}_{j,k+1} + V_{j,k}\right)$, plus $\mathcal{H}^{(2)}_{j,k+0.5}$, $\forall j$, using (11);
9.    For $\gamma = 2{:}N_\gamma$ do
10.      Correction Steps: $V^{(\gamma+1)}_{j,k+1} = V_{j,k} + \Delta T \cdot \mathcal{H}^{(\gamma)}_{j,k+0.5}$, $\forall j$;
11.      Corrector Convergence Check Step given sufficient relative tolerance $\mathrm{tol}_v$:
         If   $\max_j\left[V^{(\gamma+1)}_{j,k+1} - V^{(\gamma)}_{j,k+1}\right] < \mathrm{tol}_v \cdot \max_j\left[V^{(\gamma)}_{j,k+1}\right]$
              then $k = k+1$, $V_{j,k+1} = V^{(\gamma+1)}_{j,k+1}$, $\forall j$, break from loop $\gamma$ to return to step 6;
         endif;
12.      Corrector Evaluation Step: $V^{(\gamma+1)}_{j,k+0.5} = 0.5\left(V^{(\gamma+1)}_{j,k+1} + V_{j,k}\right)$, plus $\mathcal{H}^{(\gamma+1)}_{j,k+0.5}$, $\forall j$, using (11);
13.   end loop $\gamma$;
14. end loop $k$;
```

However, more discussion of stability and its effect on convergence is needed. Using the central finite difference for the first order derivative in (9) means that diffusion dominance has been tacitly assumed and the time step is selected by a mesh ratio condition, i.e.,

$$\begin{array}{c}\min_{j,k}\left[G^2_{0,j,k}-|F_{j,k}|\,\Delta X\right]\geq 0,\\ \Delta T<(\Delta X)^2/\max_{j,k}\left[G^2_{0,j,k+0.5}\right].\end{array}\tag{12}$$

If the jump-diffusion is not diffusion dominated, then it is advisable to use an unwinding finite difference for the first order difference to correspond to the direction of the drift,

$$\mathrm{DV}_{j,k}=(V_{j+s,k}-V_{j,k})/(s\Delta x),\quad s=\mathrm{sgn}(F_{j,k}),\quad \mathrm{sgn}(0)\equiv 1,\tag{13}$$

even though this approximation is only first order in Δx. Second order forward and backward finite differences of second order in Δx can be used as they would be used for derivative boundary conditions [4] to preserve second order accuracy.

Stochastic dynamic programming can become computationally demanding as the state dimension n_x and the common number of nodes per dimension N_x grow, as it does with PDEs, because the demand per time step grows exponentially with exponent $n_x\ln(N_x)$ and this growth is called *Bellman's Curse of Dimensionality* [4]. Advanced computing techniques like massively parallel and vector processor are needed to treat large dimensions [3].

4 Probabilistic Markov Chain Approximation

The Markov chain approximation (MCA) [5, 6] uses the discrete Markov chain process transition probabilities implied by finite differences to generate proper computational time steps and and weak convergence properties, developed by Kushner in the 1970s. This method, as in the previous section, is applicable to deterministic as well as stochastic processes of the Markov type.

Assuming the same optimization model (1) and stochastic dynamics (2), but $n_x=1$ and $n_u=1$, the Bellman SDP equation (4) with (5) and forward discrete time value function $V_k(x)\simeq v^*(x,t_k)$ becomes

$$V_{k-1}(x)=V_k(x)+\Delta t_{k-1}\min_u\left[C_k(x,u)+F_k(x,u)V_k'(x)+\frac{1}{2}G_k^2(x)V_k''(x)+\Lambda_k(\mathrm{IVH}_k(x)-V_k(x))\right],\tag{14}$$

for $k=1{:}N_t-1$, $\Delta t_{k-1}=t_k-t_{k-1}$, $t_1=t_0$, $t_{N_t}=t_f$ and $V_1(x)=Z(x,t_f)$, using similar notation of the previous sections. Let ξ_k denote the kth stage of the Markov chain fo $k=1{:}N_1$ such that $\Delta\xi=\xi_{k+1}-\xi_k=\mathrm{O}(\Delta X)$. The Markov chain transition probabilities must satisfy the proper conditions of nonnegativity and conservation of probability, as well the jump-diffusion approximation $\mathrm{O}(\Delta t_k)$ consistency conditions mentions in Section 1.

For the self or nearest-neighbor transition probabilities associated with the diffusion component processes and the finite difference grid with step ΔX using the central form for $v_k''(x)$ as in (9) and the stabilizing upwinded form for $v_k'(x)$ as in (13) with $s=\mathrm{sgn}(f_k(x,u))$,

$$\begin{array}{rl}p_k^{(\mathrm{D})}(X,X|X=x,u_{k-1})=&1-\Delta t_{k-1}\left(g_k^2(x)+\Delta X|f_k(x,u_{k-1})|\right)/(\Delta X)^2,\\ p_k^{(\mathrm{D})}(X,X\pm\Delta X|X=x,u_{k-1})=&\Delta t_{k-1}\left(0.5g_k^2(x)+\Delta X\max[\pm f_k(x,u_{k-1})]\right)/(\Delta X)^2.\end{array}\tag{15}$$

The global time interpolation mesh condition for diffusive convergence is

$$\Delta t_{k-1}\leq(\Delta X)^2/\min_{x,u}\left[g_k^2(x)+\Delta X|f_k(x,u_{k-1})|\right].\tag{16}$$

Provided that this time step is sufficiently small, then the well-known infinitesimal Poisson transition probabilities, for $j = 0, 1, 2$ or more jumps, are given by

$$p_{j,k}^{(\mathrm{J})} = \left\{ \begin{array}{ll} 1 - \lambda_k \Delta t_{k-1} + \mathrm{o}(\lambda_k \Delta t_{k-1}), & j = 0 \text{ jumps} \\ \lambda_k \Delta t_{k-1} + \mathrm{o}(\lambda_k \Delta t_{k-1}), & j = 1 \text{ jump} \\ \mathrm{o}(\lambda_k \Delta t_{k-1}), & j \geq 2 \text{ jumps}, \end{array} \right\} = p_{0,k}^{(\mathrm{J})} \delta_{j,0} + p_{1,k}^{(\mathrm{J})} \delta_{j,1} + \mathrm{o}(\lambda_k \Delta t_{k-1}), \quad (17)$$

more specifically for the mean jump count $\lambda_k \Delta t_{k-1} \ll 1$ and ignoring the $\mathrm{o}(\lambda_k \Delta t_{k-1})$ yields the $0 - 1$ Bernoulli process transition probability terms, $\{p_{0,k}^{(\mathrm{J})} = 1 - p_{1,k}^{(\mathrm{J})}, p_{1,k}^{(\mathrm{J})} = \lambda_k \Delta t_{k-1}\}$, that are consistent with the jump-diffusion approximation. Finally, adding the transition probability for the node-interpolated jump amplitude when there is a jump leads to this MCA dynamic programming computation formulation:

$$\begin{aligned} V_{k-1}^{(\mathrm{JD})}(x) \simeq \min_u \Bigg[& C_k(x,u)\Delta t_{k-1} + p_{0,k}^{(\mathrm{J})} \sum_{\ell=1}^{3} p_k^{(\mathrm{D})}(x, x+(\ell-2)\Delta X | x, u) V_k^{(\mathrm{JD})}(x+(\ell-2)\Delta X) \\ & + p_{1,k}^{(\mathrm{J})} \widetilde{\mathrm{IVH}}_k^{(\mathrm{JD})}(x,u) \Bigg], \end{aligned} \quad (18)$$

where $\widetilde{\mathrm{IVH}}_k^{(\mathrm{JD})}$ is the node-interpolated version of the integral interpolation in (10). For more details see Kushner and Dupuis [6].

5 Summary and Some Other Approaches

This article covers the fairly general computational stochastic dynamic programming for Markov stochastic processes, that includes deterministic, diffusion and jump-diffusion models as subsets. Due to space limitations, it has not been possible to cover some related topics. There have been recent results for convergence rates of computational dynamic programming in diffusion problems, most recently by Song and Yin [8], who also refer to prior convergence rate work of others. Genuine stochastic difference equation models have not been covered, but Sennott [7] supplies a wealth of discrete applications and other kinds of objective functions. See also the section in Hanson [3] reviewing the discrete version of differential dynamic programming with alternative backward and forward successive approximations.

References

[1] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*, Prentice–Hall, Englewood Cliffs, NJ, 1990.

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

[3] F. B. Hanson, *Computational Stochastic Dynamic Programming*, in Stochastic Digital Control System Techniques, Control and Dynamic Systems: Advances in Theory and Applications, vol. 76, C. T. Leondes, ed., Academic Press, New York, NY, 1996, pp. 103–162.

[4] F. B. Hanson, *Applied Stochastic Processes and Control for Jump–Diffusions: Modeling, Analysis and Computation*, Series in Advances in Design and Control, vol. DC13, SIAM Books, Philadelphia, PA, 2007.

[5] H. J. Kushner, *Numerical Methods for Stochastic Control Problems in Continuous Time*, SIAM J. Control Optim., vol. 28, 1990, pp. 999–1048.

[6] H. J. Kushner and P. G. Dupuis, *Numerical Methods for Stochastic Control Problems in Continuous Time*, 2nd ed., Springer-Verlag, New York, NY, 2001.

[7] L. I. Sennott, *Stochastic Dynamic Programming and the Control of Queueing Systems*, John Wiley & Sons, Inc., New York, NY, 1999.

[8] Q. S. Song and G. Yin, *Rates of Convergence of Numerical Methods for Controlled Regime-Switching Diffusions with Stopping Times in the Costs*, SIAM J. Control Optim., vol. 48, 2009, pp. 1831-1857.