A Stochastic Minimum Principle and an Adaptive Pathwise Algorithm for Stochastic Optimal Control

Panos Parpas [a], Mort Webster [b]

[a] *Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2AZ.*

[b] *Massachusetts Institute of Technology, 77 Massachusetts Ave, Building E40-235, Cambridge, MA 02139.*

Abstract

We present a numerical method for finite horizon stochastic optimal control models. We derive a Stochastic Minimum Principle (SMP) and then develop a numerical method based on the direct solution of the SMP. The method combines Monte Carlo pathwise simulation and nonparametric interpolation methods. We present results from a standard LQC model, and a realistic case study that captures the stochastic dynamics of intermittent power generation in the context of optimal economic dispatch models.

Key words: Stochastic Control, Monte Carlo Simulation, Electric Power Systems

1 Introduction

Stochastic optimal control is a useful formalism for developing and analyzing models that have stochastic dynamics. Applications of stochastic optimal control include manufacturing systems, option pricing, portfolio optimization, analysis of climate policies, biological and medical applications, and energy systems modeling.

Realistic models do not admit closed form solutions. As a result a large number of numerical methods have been proposed to approximate their solution. Traditional numerical methods for stochastic optimal control such as value iteration [2], policy iteration [2], and the Markov Chain approximation method in [11] all rely on a mesh. Typically, the mesh is obtained by discretizing the state. This discretization gives rise to a mesh (or a grid) and computation is performed on each point of the mesh. For example, the exact implementation of policy or value iteration requires the specification of a lookup table (or mesh in the language of this paper). Similarly, the Markov Chain approximation method [11] requires the construction of a finite difference grid. Computational complexity of classical stochastic optimal control increases exponentially with the number of state variables and they therefore suffer from the curse of dimensionality. Alternative formulations using multi-stage stochastic programming, also depend on a mesh. In the stochastic programming literature the mesh is referred to as a scenario tree. The number of points on the mesh, or the number of nodes in the scenario tree grows exponentially with the number of time-periods. In this case, stochastic programming algorithms suffer from the curse of dimensionality due to the number of time periods involved in realistic models. The problem that motivated this paper (see Section 5.2) has both a large state vector and a large number of time periods. Therefore new methods are needed to address this class of problems.

1.1 Overview of the proposed algorithm

The method proposed in this paper does not perform any (structured) discretization in the state dimension. Instead of discretizing the state dimension and performing computation on each point of the resulting mesh, the proposed algorithm relies on a three step procedure. Starting with an incumbent control law, the first step consists of forward simulations. In the setting of this paper, forward simulations are computationally inexpensive. The second step consists of backward steps that approximately solve the adjoint equation associated with the model and the incumbent control law. The difference with the classical algorithms such as policy and value iteration is that those algorithms compute the value function or optimal control for each possible state. We only visit "promising states". These promising states are ob-

Email addresses: p.parpas@imperial.ac.uk (Panos Parpas), mort@mit.edu (Mort Webster).

tained during the forward simulation phase. The solution of the system of adjoints is approximate because we only use points that were visited in the forward phase to construct the approximation. The approximation is constructed via a non-parametric method for scattered data approximation. Finally, the third step uses the information gleaned from the forward and backward steps to improve the incumbent control law. The algorithm is described in detail in Section 4.

An exact resolution of the adjoints would require the construction of a structured mesh. However, since the proposed method is used to solve for the SMP associated with the model, only the adjoints along the optimal path are needed. In other words, when the adjoint information is used in conjunction with the optimality conditions of the SMP, the exact resolution of the adjoints is not necessary. An advantage of the proposed method is that the adjoints can be interpreted as prices or hedging strategies[13]. A disadvantage of our approach is that the forward paths need to be stored and searched frequently. However, using appropriate data structures (discussed in Section 4.3) the computational burden associated with this part of the algorithm appears to be manageable even for high dimensional problems. For example, in Section 5 we report on the solution of a 30 dimensional Linear Quadratic Control (LQC) model, and compare our method with the closed form solution. We also present numerical results from a power generator scheduling case study. This application introduces some complexities not present in the LQC model. In particular the case study shows the importance of the appropriate selection of basis functions used in the numerical implementation, and the difficulty of correctly specifying a global parametric class.

1.2 Contributions & Literature Review

This paper makes three contributions. The first is the development of a Stochastic Minimum Principle (SMP) for finite horizon stochastic optimal control models driven by piecewise deterministic Markov Processes (Section 3). The second contribution is the development of the adaptive pathwise algorithm (Section 4). Our final contribution is to show that the proposed method can be applied to realistic models (Section 5). We expand on these contributions in the context of the existing literature below.

The algorithm proposed in this paper is applicable to a class of stochastic processes known as piecewise deterministic. The theoretical properties of these processes have been extensively studied, see e.g. [7]. To enable the development of the numerical method, we need to make use of an appropriate Stochastic Minimum Principle (SMP). Our first contribution is the SMP described in Section 3. A number of papers have developed SMPs for similar models. However, a form of the SMP suitable for numerical computation and appropriate for the application discussed in this paper has not appeared elsewhere. A similar SMP was developed in [17] but it assumes that the Markov processes are governed by a constant generator. Our SMP covers the finite horizon case, and allows for the generator to be both time inhomogeneous and to depend on the controls. Optimality conditions for the infinite horizon case have received more attention, e.g. [9,22]. The proof of the SMP in the latter work is based on the reduction of the problem to a deterministic infinite horizon problem. A minimum principle is then used for the deterministic problem in order to derive the necessary and sufficient conditions for the original model. From the point of view of computation, the deterministic infinite horizon problem is not in a form suitable for efficient numerical approximations. Another contribution related to the SMP, is the time discretization result in Section 4.1. The result in Section 4.1 enables the numerical implementation of the algorithm described in Section 4.2.

Our second contribution is the development of the three step numerical algorithm described in Section 4. The fact that our method does not construct a grid means that it can potentially address high dimensional problems. Many methods have been proposed to address the curse of dimensionality in stochastic optimization. Methods from the stochastic programming literature include decomposition and aggregation methods (for a review see [3,14]). The method proposed in this paper differs from the numerical methods used in Stochastic Programming in that (a) we deal with continuous time problems, (b) we allow the possibility of the probability distribution of the random variables to depend on the decisions, (c) our method does not depend on convexity assumptions, and (d) we never construct a mesh or a scenario tree. The complexity of Stochastic Programming problems grows exponentially with the number of time periods. As a result, stochastic programming models are used when only a small number of decision periods is required. The power system application that motivated this work has a large number of time periods (288). A stochastic program formulated with the coarsest possible scenario tree of two realizations for each random variable cannot be solved even with aggregation or decomposition algorithms.

We have already explained the differences of the proposed method from the traditional methods of Dynamic Programming (DP) such as value and policy iteration algorithms in Section 1.1. However, there are similarities between the proposed method and the algorithms that belong to the Approximate Dynamic Programing (ADP), Reinforcement Learning (RL) literature and the specialized algorithms developed in the energy systems area (see e.g. [10], [15], [18]). Most ADP algorithms are concerned with approximating the value function. For example Q–learning was originally proposed in the context of value function approximation (see e.g. [16]). There are, however, methods that are based on policy space ~~space~~ approximation. For example, the methods based on perturbation and sensitivity analysis (including the extension of TD and Q-learning algorithms) for policy space approximation described in [4]. The methods described in [4] are developed in the context of infinite horizon or average cost models, whereas we deal with finite horizon models. This important difference requires the development of a different approximation method. Infinite horizon or average cost models are not meaningful for the power systems application that motivated this work because these systems are not adequately described by steady state dynamics (see Section 5).

Another important difference is that sensitivity and perturbation based methods described in [4] estimate the sensitivity of the model to a set of parameters. For example, one could expand the control as a series of basis functions, and then use a perturbation method to find the optimal weight for each element of the basis. The approximation will be parametric, and convergence is within the chosen parametric class. Recently a method similar to the one described in this paper was proposed in [5]. The method in [5] is concerned with the discrete finite horizon case, whereas we deal with the continuous case. Similarly to perturbation methods, the method in [5] requires a global parametric interpolation to be used for the controls and the adjoints. The method proposed in this paper does not rely on a user specified choice of the parametric class.

Finally, ADP, RL and to a large extent perturbation methods are based on Stochastic Approximation (SA). An advantage of algorithms based on SA is that the algorithm avoids the explicit calculation of the conditional expectations that appear in models of stochastic control. Instead SA methods rely on sampling and smoothing in order to estimate these conditional expectations [16]. However, the gradients used in SA methods are stochastic and as a result these algorithms are extremely sensitive to the selection of the step-size strategy. The proposed algorithm also relies on a step-size strategy. However, because we use a Monte Carlo integration method to compute gradients our algorithm exhibits less sensitivity to the step-size strategy (the gradients are still stochastic but the variance can be controlled). To make this point clear consider the following static stochastic optimization problem,

$$\min_{u} \ \mathbb{E}[F(\eta, u)].$$

Where u is the control, and η is a random variable. Methods based on SA consist of iterations of the form,

$$u^{k+1} = u^k - \tau \nabla_u F(\eta, u). \tag{1}$$

Where k is the iteration number and τ is the step-size. The algorithm proposed in this paper is more in the spirit of,

$$u^{k+1} = u^k - \tau \mathbb{E}[\nabla_u F(\eta, u)]. \tag{2}$$

The quantity $\mathbb{E}[\nabla_u F(\eta, u)]$ cannot be estimated exactly, but a noisy estimate can be obtained via Monte Carlo integration. Methods based on (2) are less sensitive to the choice of step size but require more expensive iterations. Methods based on (1) require less expensive iterations, but are sensitive to the choice of step–size. Moreover when the problem is not static (as is the case in the model we study in this paper) the computation of the gradient $\nabla_u F$ is considerably more complicated than in the static case. The gradients in this paper are computed by the adjoint calculations described in Section 4.2.

Finally, we study the numerical performance of the algorithm to models of realistic size. It will be shown in Section 5 that the proposed algorithm can be used to solve problems with a large number of decision stages and a large state space. First we apply the method to a Linear Quadratic Control (LQC) model, for which a closed form solution is available. We then apply the algorithm to a realistic power systems model.

2 Problem Statement & Notation

We use $\eta(t)$ to denote a continuous time, discrete state Markov process. This class of processes is also known as piecewise deterministic [7]. The state space of the Markov process will be denoted by $\mathcal{M}$. The cardinality of $\mathcal{M}$ will be denoted by $|\mathcal{M}|$. For the applications that motivated this paper, such a finite state space will suffice. The theory of piecewise deterministic processes is relatively mature, and we adopt the well known framework and terminology described in [7].

As is well known (see e.g. [23]), the probability transition matrix of η satisfies the following equation,

$$\frac{dP(t,s)}{dt} = P(t,s)Q_t(x_t, u_t), \quad P(s,s) = I_{|\mathcal{M}|}, \tag{3}$$

where I_n denotes an identity matrix of dimension n. The matrix Q is called the generator of η, and is allowed to depend on time t, and on the state control pair (x, u). The $(i, j)^{\text{th}}$ entry of this matrix is denoted by q_{ij}. We will use $\mathcal{J}_s$ to denote the objective function value at time s,

$$\mathcal{J}_0(x_0, \eta_0, u) \triangleq \mathbb{E}\Bigg[\int_0^T e^{-\rho t} G_t(x_t, \eta_t, u_t) dt \\ + e^{-\rho T}\Phi(x_T) \mid x(0) = x_0, \eta(0) = \eta_0 \Bigg],$$

where u is some feasible control, $x_0 \in \mathbb{R}^n$ and $\eta_0 \in \mathcal{M}$ are the initial conditions of x and η respectively. The function $\Phi : \mathbb{R}^n \to \mathbb{R}$ is the terminal condition. With these definitions the full problem we consider in this paper is given below.

$$\begin{aligned} v_0(x_0,\eta_0) &= \min_{u\in U} \mathcal{J}_0(x_0,\eta_0,u) \\ \frac{dx}{dt} &= f(x,\eta,u), \quad x(0)=x_0, \qquad (\mathcal{P}) \\ \eta(0) &= \eta_0, \quad u\in\mathcal{A}_t. \end{aligned}$$

We make the following assumptions concerning the problem above: the feasible set U is a compact subset of $\mathbb{R}^m$, the functions $f:\mathbb{R}^n\times\mathcal{M}\times U\to\mathbb{R}^n$, $G:\mathbb{R}^n\times\mathcal{M}\times U\to\mathbb{R}$ and, $\Phi:\mathbb{R}^n\to\mathbb{R}$, are continuously differentiable in the state variable, and with bounded derivatives. The set $\mathcal{A}_t$ is used to denote the set of $\mathcal{F}_t$-adapted policies that are piecewise continuous in time, and once continuously differentiable with a bounded derivative in the state variable.

The differential equation that drives the system in $(\mathcal{P})$ is stochastic with (right) jump discontinuities. In order to be exact, one must define the class of functions that can satisfy the system in $(\mathcal{P})$. In this paper we will use the framework in [12] (Section 2.2) to study the dynamics of the system ($(\mathcal{P})$ is also called a hybrid system in the optimal control literature). The solution of the differential equation in $(\mathcal{P})$ is a cadlag function (i.e. right continuous with left limits), and the solutions of the differential equation are interpreted as Caratheodory solutions. We will use the notation $\psi_t^u(s,x_s,\eta_s)$ to denote the solution to the state equation of $(\mathcal{P})$ at time t, with the initial conditions $(x(s)=x_s,\eta(s)=\eta_s)$, and an admissible control u. If η is fixed to θ, then the state equation of $(\mathcal{P})$ is a deterministic differential equation, and we denote its solution with $\hat{\psi}_t^{u,\theta}(s,x_s)$.

3 A stochastic minimum principle

In this section we derive necessary and sufficient optimality conditions for $(\mathcal{P})$. We start by deriving a recursive equation that the objective function value of $(\mathcal{P})$~~,~~ must satisfy (Theorem 1). We will then use this recursive equation to reformulate the problem as a standard optimal control problem for which optimality conditions are already known (~~see~~ Theorem 2).

Theorem 1 *The objective function of $(\mathcal{P})$ satisfies the equation,*

$$\mathcal{J}_0(x_0,\eta_0,u) = \int_0^T z(t)\Bigg(G_t(\hat{\psi}_t^{u,\eta_0},\eta_0,u_t) + \sum_{\eta'\neq\eta_0} q_{\eta_0\eta'}(\hat{\psi}_t^{u,\eta_0},u_t)\mathcal{J}_t(\hat{\psi}_t^{u,\eta_0},\eta',u_t)\Bigg)dt + \Phi(\hat{\psi}_T^{u,\eta_0})z(T) \tag{4}$$

where $z(t)=\exp[-\int_0^t \rho + q_{\eta_0}(\hat{\psi}_k^{u,\eta_0},u_k)dk]$, and

$$q_{\eta_0}(y,u) \triangleq -q_{\eta_0,\eta_0}(y,u) = \sum_{\eta'\neq\eta_0} q_{\eta_0\eta'}(y,u).$$

PROOF. Let $\tilde{G}_t(x,\eta,u)=G_t(x,\eta,u)1_{\{t<T\}}$. Then for any given feasible control u, we can rewrite the objective function as follows,

$$\mathcal{J}_0(x_0,\eta_0,u) = \mathbb{E}\Bigg[\int_0^\infty e^{-\rho s}\tilde{G}_s(x_s,\eta_s,u_s)ds + e^{-\rho T}\Phi(x_T)\Bigg| x(0)=x_0,\ \eta(0)=\eta_0\Bigg].$$

In the preceding equation let $\mathcal{J}_0^1(x_0,\eta_0,u)$, $\mathcal{J}_0^2(x_0,\eta_0,u)$ denote the first, and second term respectively. Let τ be the first jump time of η after time zero. Suppose that $\eta(0)=\eta_0$, then the probability that the first jump occurs in time $[t,t+\Delta t]$, is given by,

$$\mathbb{P}[t<\tau<t+\Delta t] = q_{\eta_0}(x_t,u_t)e^{-\int_0^t q_{\eta_0}(x_s,u_s)ds}\Delta t + o(\Delta t), \tag{5}$$

When $s\in[0,\tau)$ then the stochastic solution $\psi_s^u(0,x_0,\eta_0)$ coincides with the deterministic solution $\hat{\psi}_s^{u,\eta_0}(0,x_0)$. Below we drop the dependence of $\hat{\psi}_s^{u,\eta_0}(0,x_0)$ on (x_0,η_0), and use $\hat{\psi}_s^u$ for brevity. Using the law of total expectation and the Markov property, we can write $\mathcal{J}_0^1$ as follows,

$$\mathcal{J}_0^1 = \mathbb{E}\Bigg[\int_0^\tau e^{-\rho s}\tilde{G}_s(\hat{\psi}_s^u,\eta_s,u_s)ds + e^{-\rho\tau}\mathcal{J}_\tau^1(\hat{\psi}_\tau^u,\eta_\tau,u_\tau)\Bigg| x(0)=x_0,\ \eta(0)=\eta_0\Bigg].$$

The expectation above is taken with respect to the probability of jumping at time τ to state η_τ given that we started at time 0 at state $x(0)=x_0$ and $\eta(0)=\eta_0$. This expectation can be expanded using (5) as follows,

$$\mathcal{J}_0^1 = \int_0^\infty q_\eta(\hat{\psi}_t^u,u_t)e^{-\int_0^t q_\eta(\hat{\psi}_s^u,u_s)ds} \times\left(\int_0^t e^{-\rho s}\hat{G}(\hat{\psi}_s^u,\eta_0,u_s)ds + I_t\right)dt \tag{6}$$

where,

$$I_t = e^{-\rho t} \sum_{\eta' \neq \eta_0} \frac{q_{\eta_0 \eta'}(\hat{\psi}_t^u, u_t)}{q_{\eta_0}(\hat{\psi}_t^u, u_t)} \mathcal{J}_t^1(\hat{\psi}_t^u, \eta', u_t).$$

Note that in the derivation of (6) we have used the fact that the probability of jumping from η_0 to state η' given that the process jumped at time τ is given by,

$$\mathbb{P}(\eta(\tau) = \eta' \mid \tau, \eta(0) = \eta_0) = -\frac{q_{\eta_0 \eta'}(\hat{\psi}_\tau^u, u_\tau)}{q_{\eta_0}(\hat{\psi}_\tau^u, u_\tau)}$$

Integrating by parts the first term in (6) and using the fact that $\tilde{G}_t = \mathcal{J}_t^1 = 0$ for $t > T$, we obtain,

$$\mathcal{J}_0^1 = \int\limits_0^T z(t)\Big(G_t(\hat{\psi}_t^u, \eta_0, u_t) + \sum_{\eta' \neq \eta_0} q_{\eta_0 \eta'}(\hat{\psi}_t^u, u_t)\mathcal{J}_t^1(\hat{\psi}_t^u, \eta', u_t)\Big)dt.$$

where $z(t) = \exp[-\rho t - \int\limits_0^t q_{\eta_0}(\hat{\psi}_s^u, u_s)ds]$. A similar argument establishes the following expression for $\mathcal{J}_0^2(y, \eta_0)$

$$\mathcal{J}_0^2 = \Phi(\hat{\psi}_T^u)z(T) + \sum_{\eta' \neq \eta_0} \int_0^T z(t)q_{\eta_0 \eta'}(\hat{\psi}_t^u, u_t)\mathcal{J}_t^2(\hat{\psi}_t^u, \eta', u_t)dt$$

Adding the two expression for $\mathcal{J}_0^1$ and $\mathcal{J}_0^2$ we obtain the required result in (4).

□

The Hamilton Jacobi Bellman equation associated with $(\mathcal{P})$ is given below. A rigorous proof can be found in Appendix A.4 of [23].

$$\rho v_t(x, \eta) = \min_u \left\{G_t(x, \eta, u) + \nabla_x v(x, \eta)^\top f(x, \eta, u)\right\} + \frac{\partial v_t}{\partial t} + \sum_{\eta' \neq \eta} q_{\eta' \eta}(v_t(x, \eta') - v_t(x, \eta)) \tag{7}$$

We use the shorthand notation $u(t) = u(t, x_t, \eta_t)$, and $\lambda(t) = \lambda(t, x_t, \eta_t)$ to denote the control and adjoint respectively. Explicit notation will be used when confusion might arise. We end this section by showing how the previous result is related to the necessary and sufficient conditions for optimality for $(\mathcal{P})$.

Theorem 2 *A necessary and sufficient condition for a control u^* to be optimal for $(\mathcal{P})$ is that there exist a solution to the following backwards equation,*

$$\begin{aligned} \frac{d\lambda_t}{dt} &= -\nabla_x \mathcal{H}_t(\hat{\psi}_T^{u^*,\eta}, \eta, u^*, \lambda_t) + (\rho + q_{\eta\eta})\lambda_t \\ \lambda_T(x, \eta) &= \nabla_x \Phi(\hat{\psi}_T^{u^*,\eta}). \end{aligned} \tag{8}$$

For any control adjoint pair (u, λ), the function $\mathcal{H}_t$ is defined as follows,

$$\mathcal{H}_t(x, \eta, u, \lambda) = G_t(x, \eta, u) + \lambda^\top f(x, \eta, u) - \sum_{\eta' \neq \eta} q_{\eta\eta'} v_t(x, \eta)$$

In addition u^* must satisfy:

$$\mathcal{H}_t(x, \eta, u^*, \lambda) - \mathcal{H}_t(x, \eta, u, \lambda) \leq 0 \quad \forall u \in U. \tag{9}$$

PROOF. It follows from Theorem 1 that $(\mathcal{P})$ is equivalent to

$$v_0(x_0, \eta_0) = \min_{u \in U} \left\{ \int\limits_0^\top z(t) \left(G_t(x_t, \eta_t, u_t) + \sum_{\eta' \neq \eta_0} v_t(x, \eta') \right) dt + z(T)\Phi(x_T) \right\}$$

$$\frac{dx}{dt} = f(x, \eta, u), \quad x(0) = x_0, \ \eta(0) = \eta_0 \tag{$\tilde{\mathcal{P}}$}$$

Since $(\tilde{\mathcal{P}})$ is a deterministic finite horizon problem the result follows from the application of the deterministic maximum principle on $(\tilde{\mathcal{P}})$. □

4 An Adaptive Pathwise Algorithm

In this section we show how the SMP derived in the previous section can be used to develop an efficient algorithm. There are three key elements of the proposed algorithm. The first is the time discretization scheme of the continuous time problem in $(\mathcal{P})$. Integrating the state equation forward in time using a simple Euler scheme is sufficient. However, more care needs to be taken when integrating the adjoints backward. We describe our time discretization scheme in Section 4.1. The second element of our approach is the application of the minimum principle. In order to apply the minimum principle we first simulate forward in time the incumbent control. We call this the forward simulation phase. We then proceed to improve the incumbent policy by calculating the adjoint process. This step amounts to the solution of a backwards stochastic differential equation (BSDE). We call this the backward simulation phase. This is the most expensive part of the algorithm. The third, and final element of our approach, is the scattered data interpolation algorithm. Although the algorithm is based on a Monte Carlo simulation methodology, we still need a mesh, or some sort of grid, in order to represent the incumbent control and the adjoint process. If we used a traditional grid (e.g. a finite difference grid) for this part of the algorithm, our approach would not scale to problems of realistic size. For this reason we use the method described in Section 4.3 that does not require the construction of a structured mesh.

4.1 Time discretization

Applying the Euler–Maruyama [24] scheme to the state equation of $(\mathcal{P})$ we obtain the following discrete time dynamics.

$$x_{t+\delta t} = f^{\Delta}(x_t, u_t, \eta_{t+\delta t}) \triangleq x_t + f(x_t, u_t, \eta_{t+\delta t})\delta t. \quad (10)$$

Note that $x_{t+\delta t}$ is, by definition, adapted to the filtration generated at time t. A simple Euler scheme can not be directly applied backwards in time because it will not yield an $\mathcal{F}_t$ adapted adjoint process. However, we show below how an $\mathcal{F}_t$ adapted process can be constructed by a discrete time approximation and an expectation that can be calculated with Monte Carlo simulation. An expectation taken with respect to,

$$\mathbb{P}^{\Delta}(t, t+\delta t) = \exp\left[\int_t^{t+\delta t} Q(s, u_s)ds\right].$$

is denoted by $\mathbb{E}^{\Delta}[\cdot]$, where $\exp[\cdot]$ above denotes the matrix exponential.

Theorem 3 *Suppose that there exists a function λ^{δ} that satisfies the following equation,*

$$\lambda^{\delta}_t(x_t, \eta_t) = \mathbb{E}^{\Delta}\Bigg[\nabla_x G(x_t, \eta_{t+\delta t}, u_t)\delta t - \rho\delta t \lambda^{\delta}_{t+\delta t}(\eta_{t+\delta t}) \\ \nabla_x f^{\Delta}(x_t, u_t, \eta_{t+\delta t})\lambda^{\delta}_{t+\delta t}(\eta_{t+\delta t}) | x_t, \eta_t\Bigg] \quad (11)$$

and the boundary condition, $\lambda^{\delta}(T) = \nabla\Phi(x_T)$. Then as $\delta t \to 0$, λ^{δ} is also a solution to (8).

PROOF. We proceed by breaking up (11) into the following three terms which we analyze in turn.

$$\begin{aligned} I_1(t) =& \mathbb{E}^{\Delta}[\nabla_x G(x_t, \eta_{t+\delta t}, u_t)\delta t] \\ I_2(t) =& -\lambda^{\delta}_t(\eta_t) + \mathbb{E}^{\Delta}[\nabla_x f^{\Delta}(x_t, u_t, \eta_{t+\delta t})\lambda^{\delta}_{t+\delta t}(\eta_{t+\delta t})] \\ I_3(t) =& \mathbb{E}^{\Delta}[-\rho\delta t \lambda^{\delta}_{t+\delta t}(\eta_{t+\delta t})] \end{aligned}$$

If λ^{δ} is chosen to satisfy (11) then $(I_1 + I_2 + I_3)/\delta t = 0$. Next note that,

$$\frac{I_1}{\delta t} = \frac{1}{\delta t}\Big[\sum_{\eta' \neq \eta} q_{\eta\eta'}(u_t)(\nabla_x G(x_t, \eta, u_t) - \nabla_x G(x_t, \eta', u_t))\delta t^2 \\ + \nabla_x G(x_t, \eta, u_t)\delta t\Big] + o(\delta t)$$

$I_2(t)/\delta t$ can be expanded as follows,

$$\begin{aligned} \frac{I_2}{\delta t} &= \frac{1}{\delta t}[-\lambda^{\delta}_t(\eta_t) + E^{\Delta}[(\nabla_x f^{\Delta}(x_t, u_t, \eta_{t+\delta t})\delta t)\lambda^{\delta}_{t+\delta t}(\eta_{t+\delta t})] \\ &= \frac{1}{\delta t}\Big[\nabla_x f(x, \eta_t)\lambda^{\delta}_t(\eta_t)\delta t + \frac{d\lambda^{\delta}(t, \eta_t)}{dt}\delta t + \\ &\sum_{\eta' \neq \eta} q_{\eta\eta'}(u_t)(\lambda^{\delta}_t(\eta') - \lambda^{\delta}_t(\eta))\delta t\Big] + o(\delta t). \end{aligned}$$

Finally $I_3(t)/\delta t$ can be expanded as follows,

$$\frac{I_3}{\delta t} = \frac{1}{\delta t}\Big[-\rho\delta t \lambda^{\delta}_t(\eta_t) + \rho\delta t^2 \sum_{\eta' \neq \eta} q_{\eta\eta'}(u_t)(\lambda^{\delta}_t(\eta') - \lambda^{\delta}_t(\eta))\Big]$$

Collecting all the terms we conclude that as $\delta t \to 0$ then $\lambda^{\delta}_t(\eta_t)$ satisfies (8). □

The discretized adjoint process is equivalent to the adjoint process of the discrete time minimum principle (see c.f. Theorem 1.23 in [19]).

4.2 Algorithm ~~Decsciption:~~ APA (Adaptive Pathwise Algorithm)

Using the result from Theorem 3, the algorithm is specified below. We will refer to this algorithm as APA (Adaptive Pathwise Algorithm) in the numerical experiments section. In the description of the algorithm below we use the index "k" for the iteration counter, and "j" is used to index different paths. For example u^k_j is shorthand for the value of the incumbent control $u(t_j, x_j, \eta_j)$ at iteration k.

`[Initialization]`: Let *tol* be a user specified convergence tolerance parameter. Set $k \leftarrow 0$, $t \leftarrow 0$ and let $u^k(t, x, \eta)$ be arbitrary.
`[Forward Simulation - k]`: Simulate M sample paths using the state transition equation and u^k

$$x_j(t + \Delta t) = x_j(t) + f(x_j(t), u^k_j, \eta^{t+\Delta t}_j)\Delta t$$

Let $\mathcal{G}^{\Delta}$ denote the set of sample paths generated during the forward phase.
`[Backward Simulation - k]`: For each path $j \in \mathcal{G}^{\Delta}$
(1) Apply boundary condition.

$$\lambda^k(T) \leftarrow \nabla_x \Phi(x_j(T)) \quad (12)$$

(2) Let $t_j \leftarrow T - \Delta t$, and perform backward steps:

$$\lambda^k(t_j) \leftarrow E^{\Delta}\Big[\nabla_x G(x_j, \eta_j^{t+\Delta t}, u_j^k)\Delta t +$$
$$(\nabla_x f^{\Delta}(x_j, u_j^k, \eta_j^{t+\Delta t}) - I\rho\Delta t)\lambda(t_j + \delta t, \eta_j^{t+\Delta t}) \mid x_j, \eta_j\Big]$$
$$D^k(t_j, \eta_j) \leftarrow E^{\Delta}\Big[\nabla_u G(x_j, \eta_j^{t+\Delta t}, u_j^k) +$$
$$\nabla_u f^{\Delta}(x_j, u_j^k, \eta_j^{t+\Delta t})\lambda(t_j + \Delta t, \eta_j^{t+\Delta t}) \mid x_j, \eta_j\Big]$$

(3) Set $t \leftarrow t - \Delta t$, and perform backward steps until $t = 0$.

`[Update Policy - `k`]`

$$u^{k+1}(t, x^j, \eta^j) \leftarrow \Pi_U[u^k - \kappa D^k(t_j)].$$

Where $\Pi_U[\cdot]$ denotes the projection on the set U.

`[Convergence Test]` Stop if,

$$\max_j \frac{|u^{k+1}(0, x^j, \eta^j) - u^k(0, x^j, \eta^j)|}{|u^{k+1}(0, x^j, \eta^j)|} < tol$$

Otherwise set $k \leftarrow k+1$ and go to `Forward simulation` step.

In the `[Forward Simulation]` phase we discretize the system as discussed in Section 4.1 (see (10)). After the terminal time is reached (i.e. $t = T$) we apply the appropriate boundary condition to the adjoint equation (see (12)). The algorithm then proceeds to solve the adjoint equation in the `[Backward Simulation]` phase. The equation for λ in the Backward Phase is given by (11) and is derived in Theorem 3. The equation for D in the Backward Phase follows from (9). The `[Backward Simulation]` phase terminates when $t = 0$. In the third phase, `[Update Policy]`, we use the information gleaned from the Backward Phase to update the control. The notation $\Pi_U[\cdot]$ denotes the projection on the set U, and κ denotes the step-size. In our numerical results in Section 5 we use a constant step-size. Note that because we use a Monte Carlo integration method to compute gradients, our algorithm exhibits less sensitivity to the step-size strategy compared to stochastic approximation algorithms.

In Step (2) of the Backward Phase, the adjoint λ and the gradient D are not evaluated at all possible states (as in e.g. policy iteration) but only at the states visited during the forward phase. As a result when Monte Carlo is used to estimate the expectations in Step (2), the algorithm needs to estimate the value of the adjoint at states that have not been visited during the forward phase. This challenging problem is solved using the method described next.

The derivative $D^k(t_j)$ is defined in Step (2) of the `[Backward Simulation]` phase of the algorithm and used in Step (3) `[Update Policy - `k`]` needs to be motivated further. To this end, suppose that (λ^*, u^*) is the optimal ~~control/adjoint~~ pair. Then ~~using similar arguments as in Theorem 4~~,

$$\mathcal{J}_t(x, \eta, u^*) = E\Big[G(x, \eta_{t+\delta t}, u^*)\delta t + (1 - \rho\delta t)(v_t(x, \eta_{t+\delta t})$$
$$+ \nabla_x V(x, \eta_{t+\delta t})\delta x + \frac{\partial v}{\partial t}\delta t)|x,\ \eta\Big] + o(\delta t)$$

It follows that,

$$0 = \nabla_u \mathcal{J}_t(x, \eta, u^*)$$
$$= \mathbb{E}[\nabla_u \Bigg(G(x, \eta_{t+\delta}, u)\delta t + (1 - \rho\delta t)(v_t(x, \eta_{t+\delta t})$$
$$+ \nabla_x V(x, \eta_{t+\delta t})\delta x + \frac{\partial v}{\partial t}\delta t)|x,\ \eta]\} \Bigg) + o(\delta t)$$

Dividing by δt, and taking limits we obtain,

$$\nabla_u \mathcal{J}_t(x, \eta, u^*) = \mathbb{E}\big[\nabla_u G(x, \eta_{t+\delta}, u^*)$$
$$+ \nabla_u f(x, \eta_{t+\delta t}, u^*)\lambda^*(x, \eta_{t+\delta t})|x,\ \eta\big]$$

Note that many of the terms drop out since u^* is assumed to be optimal. Therefore the quantity D is the derivative of $\mathcal{J}$ along the optimal path. Of course, u and λ are not optimal. The basic idea behind the algorithm then is to use the approximate information currently known about u~~,~~ and λ ~~and~~ compute D. Then the algorithm takes a step along the direction suggested by D. The result is to drive D to zero. Under the assumptions of this paper when D is zero then $\nabla_u J = 0$. Therefore, even if D is not the same as the gradient of $\mathcal{J}$ with respect to u, when the algorithm converges (under convexity assumptions) it should converge to an optimal solution. A complete convergence proof of the algorithm is beyond the scope of this paper.

4.3 Scattered Data Approximation

After the forward phase of the proposed algorithm we have, at time T, a set of unstructured data points $\mathcal{G}^{\Delta} = \cup_j \mathcal{G}_j^{\Delta}$. Where $\mathcal{G}_j^{\Delta}$ represents the j^{th} path generated by the algorithm given by $\mathcal{G}_j^{\Delta} = \{(x_j^0, \eta_j^0, 0), \ldots (x_j^T, \eta_j^T, T)\}$. In order to make the algorithm implementable we need to solve two important practical problems. The first is how to interpolate between the data sites in order to do the backwards simulation. The second issue is how to structure the data generated from the MC steps so that the algorithm is tractable.

For the interpolation problem we use a non-parametric method called Moving Least Squares (MLS). This is a standard method and we refer the interested reader to

[8] for the details. The application of the MLS method to this class of problems where little is known about the functional form of the optimal control, and quite often the optimal control is merely piecewise continuous is expedient because the choice of basis functions does not play a crucial role in the quality of the approximation. In the next section we show that even though we only use linear interpolation, because of the local nature of the weight functions, and the possibility to recompute the weights we can interpolate non-linear functions accurately. In order to ensure that the coefficients in the MLS method can be computed efficiently we need to implement the appropriate data structures for large sets of unstructured data. We use kd-trees in order to solve this problem (the use of kd-trees is standard in the area of scattered data approximation [21]). The basic idea behind kd-trees is to split the number of data sites into a small number of subsets such that each subset contains a comparable number of points. Once such a data structure is built, a range search (for example), only takes $O(\log N)$ time.

5 Numerical Experiments

In this Section we discuss the numerical implementation of the algorithm. The numerical experiments were run on a standard desktop computer. We performed $M = 10000$ forward simulations for all the results reported below. The step size κ is held constant to 0.01 for all the results reported below. We used a a tolerance of 1% to check for convergence (see Step 3 of the algorithm). In Section 5.1 we validate our implementation on the LQC model. The solution of the LQC model is known in closed form. The results obtained with the LQC model are useful since they validate the proposed approach. For the LQC model the control and adjoints are linear functions of the state. For this reason, one of the main advantages of the algorithm i.e. its non-parametric nature is not clear. In order to illustrate the usefulness of the non-parametric approach we also report results from a real application in Section 5.2. Interesting applications have controls and adjoints that are in general non-linear, therefore the results from Section 5.2 will be useful to other applications as well.

5.1 Validation with the LQC model

The LQC model is well known, and we refer the interested reader to [2] for the problem specification and its solution. We selected the coefficients of the model at random, but we ensured that the system is stable. We used a discretization parameter of Δt of 0.005, for a horizon $T = [0, 1]$ (i.e. 200 time periods). In Figure 5.1 (n denotes the dimensionality of the state vector, and m denotes the dimensionality of the control vector) we show the convergence of the algorithm for the controls. Note that both the control and the adjoints are high dimensional

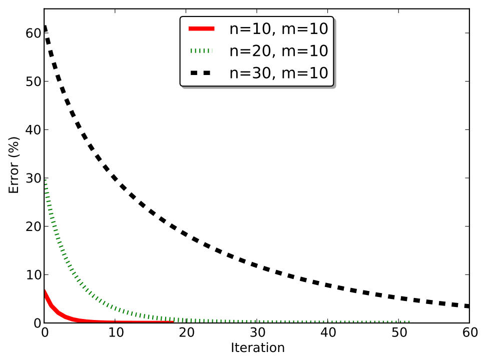


Fig. 1. Convergence of the controls. We use n to denote the dimensionality of the state vector, and m denotes the dimensionality of the control vector. Error is measured as $\|u_k(x_0) - u^*(x_0)\|/\|u^*(x_0)\|$, where x_0 denotes the initial point, and $u^*(x_0)$ is the optimal decision at the first time period. Convergence of the value function and adjoint functions behave in a similar manner.

Table 1
Solution times with 1% error tolerance

n	m	Iterations	CPU Time (secs)
10	10	20	5.51
20	10	54	47.23
20	20	183	164.65
30	10	178	331.58
30	30	263	505.23

objects ($n = 10, 20$ and 30) but still the algorithm finds the optimal solution in a reasonable amount of time. The amount of time required to find the solution within 1% is shown in Table 1.

5.2 Application to a Power Systems Case Study

In this section, we illustrate the application of the algorithm to a more complex model, an economic dispatch model with high penetration of renewables. The economic dispatch model is a standard power systems model that is used for production scheduling between different generators so that costs are minimized [6]. A review of different issues concerning economic dispatch can be found in [6]. We will focus on the case where the generator mix contains a large amount of intermittent wind generation. Traditionally economic dispatch models are deterministic. Even though some stochastic extensions have been proposed [20], most authors consider a small number of time periods or a rolling horizon formulation. Due to the computational complexity involved with solving models with a short time step and a large number of state variables, stochastic optimal control has not been conventionally used in this area. In what follows we show that the algorithm described in this paper can be effectively implemented for this application. As this is a new algorithm, the size of the problem we consider is moderate (six dimensional state space). Given the solution times for the current model, larger systems could be solved with the proposed method. In Section 6 we discuss the computational and mathematical extensions required so that the method can be scaled up to more realistic system sizes.

5.2.1 The model

We use $x_i(t)$ to denote the output from generator i, and $u_i(t)$ to denote the change in output from generator i at time t. We use $d(t)$ to denote the deterministic part of demand, and $\eta_0(t)$ specifies a stochastic disturbance in demand. We use the Markov processes $\eta_i(t), i = 1, \dots, n$ taking values in $[0, 1]$ to represent the stochastic availability of each generator $i \geq 1$.

$$\min_u E\Big\{ \int_0^T \sum_{i=1}^n c_i x(t) + c^+ x^+(t) + c^- x^-(t) dt \Big\}$$

$$\frac{dx_i}{dt} = u_i(t), \tag{13a}$$

$$\sum_{i=1}^n x_i(t) + x^+(t) - x^-(t) = d(t) + \eta_0(t), \tag{13b}$$

$$-R_i \leq u_i(t) \leq R_i \tag{13c}$$

$$x_i(t) + u_i(t) \geq 0, x^+(t) \geq 0,\ x^-(t)) \geq 0 \tag{13d}$$

$$x_i(t) \leq \bar{x}_i \tag{13e}$$

$$u_i(t) \leq (\bar{x}_i - x_i)\eta_i(t) \tag{13f}$$

The constraint in (13b) specifies that demand must equal supply at all time periods. The positive variables x^+, and x^- account for overproduction, and underproduction of electricity respectively. Both quantities are penalized in the objective by $c^{+,-}$. This type of constraint is not covered by the theory developed in Section 3. In order to incorporate it into the algorithm, an exact penalty function with exponential smoothing is used [1]. The bounds in (13c), (13d), (13e) enforce the ramping limits ~~for each generator~~, the minimum and maximum output from each generator respectively. The objective function minimizes the cost associated with meeting demand. We use typical cost parameters from [6].

5.2.2 Solution of a deterministic model

When η is fixed to some constant then the model in (13) is deterministic and can be solved with a linear programming solver. In order to derive the linear program, the objective function was discretized using a quadrature rule (in our case a simple rectangle rule was used). The state equation was discretized with a simple first order Euler scheme. Note that in the deterministic case, the optimal control is open loop, and that x_t does not depend on η. With these discrete time approximations the model can be solved with a standard linear programming solver. Setting η to some constant (η is allowed to be time dependent), the algorithm proposed in this paper can also be used to solve the deterministic version of (13). The results tabulated in Table 2 show that the average (averaged over time) difference between the control obtained between the two methods is very small. We use $\bar{u}_i^S$ ($\bar{u}_i^D$) to denote the average stochastic (deterministic) policy for generator i. The solution times in Table 3 (top 2 entries) suggest that the algorithm is competitive with the deterministic algorithm. In the deterministic case the APA algorithm is initialized using a random feasible point.

Table 2
Error

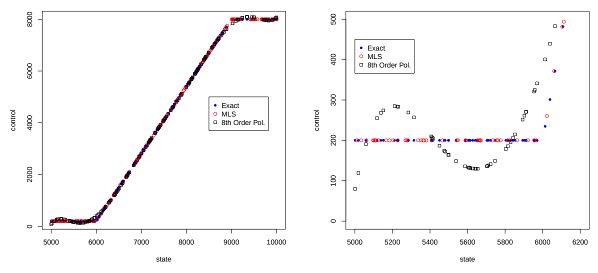
i	$\lvert\bar{u}_i^S - \bar{u}_i^D\rvert$
1	0.26%
2	0.03%
3	0.15%
4	0.10%
5	0.02%

Table 3
CPU Time (in secs)

Algorithm	Stochasticity	Time
LP-Approx.	–	84
APA	–	201
APA	Wind	1152
APA	Wind & Dem.	1204

5.2.3 Interpolation with the MLS method

It is instructive to compare the MLS method with global regression methods in the context of stochastic optimal control. This test was performed on the one–dimensional version of the deterministic version of (13). The reason we made this test on such a small problem is that the solution can be obtained in closed form. In Figure 5.2.3 we plot the optimal control, and the interpolated values using the moving least square method described in section 4.3, and a linear regression scheme using a 8th order polynomial. In the MLS approximation we use a linear basis. The global fit is shown in Figure 2(a). In order to understand the differences better we plotted the differences for a smaller range in Figure 2(b). The latter figure illustrates the difficulty of using a global regression method in optimal control methods. The optimal control in this case is linear in the state for the range $[6000, 9000]$ and constant when outside this range. The control has this piecewise linear form due to the ramping constraints. The MLS scheme, even though a linear basis is used, manages to deal with the nature of the optimal control very well. On the other hand, a regression method is very accurate in some regions, but in some regions it does very poorly. Due to the gradient descent nature of the proposed algorithm, global regression methods create numerical instabilities that force the algorithm to oscillate and never converge. This is not just a property of our algorithm but also for other algorithms that rely on gradient information (e.g. Stochastic Approximation, Approximate Dynamic Programming algorithms [16], sensitivity/perturbation methods [4]). Thus the non-parametric nature of the proposed method is fundamental to the numerical performance of this class of algorithms.



(a) (b)

Fig. 2. Comparison of Moving Least Squares (MLS) with linear basis and global regression with 8^{th} order polynomial.

5.2.4 *Comparison of stochastic and deterministic solutions*

In our final test we compared the solutions obtained with the deterministic and stochastic version of the model. The results are shown in Figure 3. When the stochastic model was run for this test, the only stochastic parameter was the availability of wind. Thus demand is deterministic in both models. Both models were used to generate optimal policies. Then the two optimal policies were used during the forward phase of the algorithm (no backward steps were performed since the algorithm was used in simulation mode only). If for the deterministic optimal policy the amount of available power was not enough to meet demand, the gas turbine generator ($i = 4$) was used in order to meet the demand at every time period. Finally the results at each time period were averaged in order to obtain the graph in Figure 3.

It can be seen from these results that the stochastic and deterministic model on average use the same amount of wind energy. However, the stochastic model anticipates the unavailability of wind and as a result ramps up the coal and CCGT generators sooner. Therefore in the stochastic model we can address the fluctuations in wind generation without resorting to the expensive gas turbine generator. In the deterministic model wind output was assumed to be a deterministic function of time. However, when run in the "real" setting where wind availability is stochastic, the coal and CCGT generators cannot be ramped up fast enough and so the expensive GT generator is used instead. The solution time for the stochastic model is shown in the third row of Table 3. In the final row of Table 3 we show the solution time when both demand and wind are uncertain.

6 Conclusions

We introduced a new numerical method based on the solution of a Stochastic ~~Maximum~~ Principle ~~(SMP)~~ and showed how the SMP can be discretized in order to derive an implementable algorithm. The proposed method

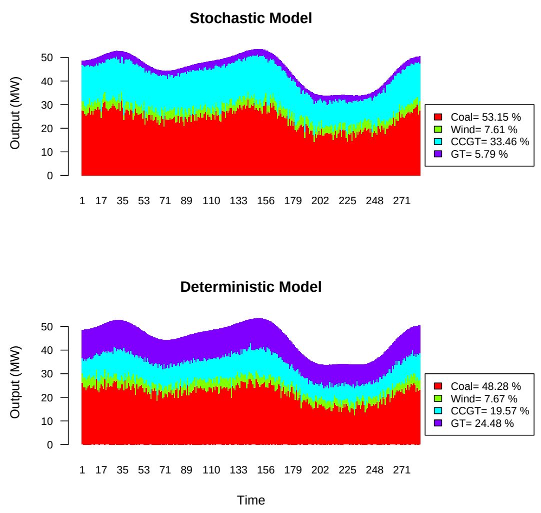


Fig. 3. Comparison of the deterministic with the stochastic model. The stochastic model ramps up coal and CCGT units earlier and therefore makes less use expensive GT (Gas Turbines).

does not perform any discretization in the state dimension, and does not perform computation at every possible state. Instead it relies on an iterative forward/backward scheme. We showed how nonparametric interpolation methods can be used to estimate the value of the adjoint at states that have not been visited. We validated the proposed algorithm on a standard LQC model. We have also shown that the method can be applied to a realistic model from power systems. We have used a scattered data interpolation technique in our algorithm. There are many other approaches that could be explored, including methods that interpolate derivatives. Finally, given the pathwise nature of the algorithm, it will be straightforward to implement in parallel. Finding an efficient way to perform the interpolation of the adjoints will be a challenge. However given that interpolation relies on neighborhood of points, this step can be done efficiently.

References

[1] D.P. Bertsekas. *Nonlinear programming.* Athena Scientific Belmont, MA, 1999.

[2] D.P. Bertsekas. *Dynamic programming and optimal control, vol. I & II.* Athena Scientific, 2007.

[3] J.R. Birge and F. Louveaux. *Introduction to stochastic programming.* Springer Verlag, 1997.

[4] X.R. Cao. *Stochastic learning and optimization: a sensitivity-based approach*, volume 17. Springer-Verlag New York Inc, 2007.

[5] P. Carpentier, G. Cohen, and A. Dallagi. Particle methods for stochastic optimal control problems. *Arxiv preprint arXiv:0907.4663*, 2009.

[6] C. Claudio, A. Gomez-Exposito, A.J. Conejo, C. Canizares, and G.E. Antonio. *Electric Energy Systems: Analysis and Operation.* CRC, 2008.

[7] M. H. A. Davis. *Markov models and optimization*, volume 49 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 1993.

[8] G.E. Fasshauer. *Meshfree approximation methods with Matlab*. World Scientific Pub Co Inc, 2007.

[9] A. Haurie and C. Van Delft. Turnpike properties for a class of piecewise deterministic systems arising in manufacturing flow control. *Annals of Operations Research*, 29(1):351–373, 1991.

[10] D. Kuhn, P. Parpas, and B. Rustem. Stochastic optimization of investment planning problems in the electric power industry. *Process Systems Engineering: Volume 5: Energy Systems Engineering*, 2008.

[11] Harold J. Kushner and Paul Dupuis. *Numerical methods for stochastic control problems in continuous time*. Springer-Verlag, New York, second edition, 2001.

[12] A. Ludwig. *Random dynamical systems*. Springer Monographs in Mathematics. Springer-Verlag, Berlin, 1998.

[13] B.K. Øksendal and A. Sulem. *Applied stochastic control of jump diffusions*. Springer Verlag, 2005.

[14] P. Parpas and B. Rustem. Computational assessment of nested benders and augmented lagrangian decomposition for mean-variance multistage stochastic problems. *INFORMS Journal on Computing*, 19(2):239–247, 2007.

[15] P. Parpas and M. Webster. A stochastic multiscale model for electricity generation capacity expansion, June 2010. Submitted.

[16] W.B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*. Wiley-Interscience, 2007.

[17] R. Rishel. Control of systems with jump markov disturbances. *Automatic Control, IEEE Transactions on*, 20(2):241 – 244, April 1975.

[18] N. Secomandi and M.X. Wang. A computational approach to the real option management of network contracts for natural gas pipeline transport capacity. *Manufacturing & Service Operations Management*, 2012.

[19] A. Seierstad. *Stochastic Control in Discrete and Continuous Time*. Springer New York, 2009.

[20] C. Weber. *Uncertainty in the electric power industry: methods and models for decision support*. Springer Verlag, 2005.

[21] H. Wendland. *Scattered data approximation*. Cambridge Univ Pr, 2005.

[22] J. J. Ye. Dynamic programming and the maximum principle for control of piecewise deterministic Markov processes. In *Mathematics of stochastic manufacturing systems (Williamsburg, VA, 1996)*, volume 33 of *Lectures in Appl. Math.*, pages 365–383. Amer. Math. Soc., Providence, RI, 1997.

[23] G. Yin and Z. Qing. *Continuous-time Markov chains and applications*, volume 37 of *Applications of Mathematics (New York)*. Springer-Verlag, New York, 1998. A singular perturbation approach.

[24] C. Yuan and X. Mao. Convergence of the euler–maruyama method for stochastic differential equations with markovian switching. *Mathematics and Computers in Simulation*, 64(2):223–235, 2004.