OPTIMIZING REFLECTED BROWNIAN MOTION: A NUMERICAL STUDY

A Thesis

Submitted to the Faculty

of

Purdue University

by

Zihe Zhou

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Industrial Engineering

December 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

ACKNOWLEDGMENTS

Thank you very much to Dr. Honnappa for recognizing my potential, trusting in my capability and patiently guiding me through this project. Thank you to Dr. Pasupathy and Dr. Hunter for kindly joining my advisory committee and raising many valuable thoughts and comments on this thesis. I would also like to thank undergraduate student Xiaotian Wang for his devotion to the PDE method part of the thesis. His work included programming, algorithm speed enhancement, numerical integration and other details on the PDE method.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

RBM	reflected Brownian motion
SPSA	simultaneous perturbation stochastic approximation
MCGD	Monte Carlo gradient descent method
PDF	probability density function
CDF	cumulative density function
PDE	partial differential equation
SDE	stochastic differential equation
HJB	Hamilton-Jacobi equation
MSE	mean squared error
FEM	finite element method
GD	gradient descent algorithm

ABSTRACT

Zhou, Zihe MS., Purdue University, December 2019. Optimizing Reflected Brownian Motion: A Numerical Study. Major Professor: Harsha Honnappa.

Reflected Brownian motion is a canonical stochastic process used to model many engineered systems. For example, the state of a queueing system experiencing heavy-traffic is well approximated by a reflected Brownian motion. It has also been used to model chemical reaction networks, as well as financial markets. The optimization and design of such systems can be modeled by stochastic optimization problems defined as additive functionals of reflected Brownian motions over a fixed time horizon. In this thesis, we are interested in a sub-class of problems where the design variable is the drift function of the RBM; in the one-dimensional setting we consider, this is without loss of generality. We also draw further distinctions with the stochastic optimal control problems that are driven by reflected Brownian motion processes, where the objective is to find an adaptive control policy. In contrast, the optimization problem here must be solved once at time zero and hence is not a stochastic optimal control problem. Modulo certain regularity conditions, our problem can be viewed as a deterministic optimal control problem that is (in theory) amenable to a dynamic programming solution. We derive the corresponding Hamilton-Jacobi-Bellman equation. However, this partial differential equation is both non-linear and with non-trivial boundary conditions, necessitating numerical solutions. We demonstrate numerical results from solving the Hamilton-Jacobi-Bellman equation using the finite element method. However, this approach suffers from the "curse of dimensionality." Therefore we develop Monte Carlo simulation optimization methods for solving the stochastic optimization problem, for a time-discretized approximation of the original problem. We avoid the curse of dimensionality by using gradient descent to compute the optimal (time-discretized) drift. However, the gradient of the objective is not known in closed form and must be estimated using the simulation sample paths. We develop a bespoke gradient estimator that exploits the strong Markov property of the reflected Brownian motion process to express the gradient as a nested expectation. We compare the corresponding Monte Carlo estimator with the well-studied simultaneous perturbation stochastic approximation method. Our numerical results show that the Monte Carlo gradient descent method outperforms the simultaneous perturbation stochastic approximation method on our specific problem instance.

1. INTRODUCTION

Reflected Brownian motion (RBM) as a stochastic process with regulation conditions, has long been a natural instrument to model a variety of practical and technical problems under different subjects. Proven by Whitt and Iglehart, [1] the limiting distribution of the workload (e.g. the number of customers waiting to be served) process follows the distribution of an RBM under certain conditions. The RBM could thereby be applied to optimization on queueing systems. For instance, RBM could effectively take part in the hospital patients scheduling problem [2]. In the financial domain, it is used to model interest rate and stock price with daily trading limits or 'circuit breakers'. In biophysics, one of the derivatives of RBM, PRBM (partially reflected Brownian motion) is an efficient model of species' diffusion/transport across an interface like cellular membrane [3]. As RBM is a prevailing model for numerous such problems, a natural topic arises and that is the optimization on RBM, specifically speaking, optimization on cost functions driven by RBM's.

In this thesis, we focus on the optimization of RBM, in particular, focusing on minimizing the expectation of an additive functional of RBM together with terminal conditions. We are interested in the optimal control process, known as the drift function of RBM and we seek different approaches to numerically solving the optimization problem. One way to look at the problem is through a deterministic optimal control point of view. In this perspective, our objective function's dynamic is described by a partial differential equation (PDE) which has a unique solution. The optimization problem can be posed as a Hamilton-Jacobi-Bellman equation. In general, solving the HJB equation is nearly impossible analytically, and one must resort to numerical methods such as the finite element method (FEM). However, FEM suffers from complex diagnostics. This creates the desideratum for an alternative approach to solving the optimization problem. Another approach to the problem is to use Monte Carlo simulation together with gradient descent or stochastic approximation techniques to solve the optimization. This method entails a discretization of the problem, resulting in a significant dimension reduction at the cost of increased variance and approximation error.

In the remainder of this chapter, we present background knowledge regarding RBM's. We discuss properties of RBM including the distribution functions and the role of our optimizer, the drift function. Chapter 2 focuses on the PDE approach. we derive the HJB equation and discuss the FEM approach to numerically solving the HJB equation. Followed by limitations and their solutions regarding the PDE approach, the convexity of the objective function is examined and it leads us to the chapter focused on simulation. In chapter 3, after arguing the necessity of an exact simulation method, we present the standard RBM simulation algorithm of Asmussen, Glynn and Pitman [4]. In chapter 4, we present two simulation-based methods for gradient estimation of stochastic optimization problems driven by RBM's. We present extensive numerical experimentation results in the final chapter. These results reveal various feasibility, efficiency, and other issues, leading to future research opportunities.

1.1 Reflected Brownian Motion

1.1.1 Definition

The RBM is a stochastic process which is the sum of a Brownian motion and a reflection term [5].Denote the RBM with respect to time in a time horizon $t \in [0, T]$ as X_t, the drift function as μ_t, the diffusion coefficient as σ_t and the driving Brownian motion as B_t. The expression for RBM is:

$$X_t = \mu_t + B_t + L_t,$$

and it can be seen as the solution to the following stochastic differential equation:

$$dX_t = \mu_t' + \sigma dB_t + dL_t,$$

where $L_t = \sup_{0 \leq s \leq t}(-\mu_s - \sigma_s B_s)^+$ and μ_t' is the derivative of μ_t if it is differentiable anywhere, and defined as the right derivative of μ_t at t. L_t is called a regulator process and it regulates RBM from being negative. As it is showed in the following graph. Whenever the driving Brownian motion reaches a new infimum, the corresponding RBM is at level 0.

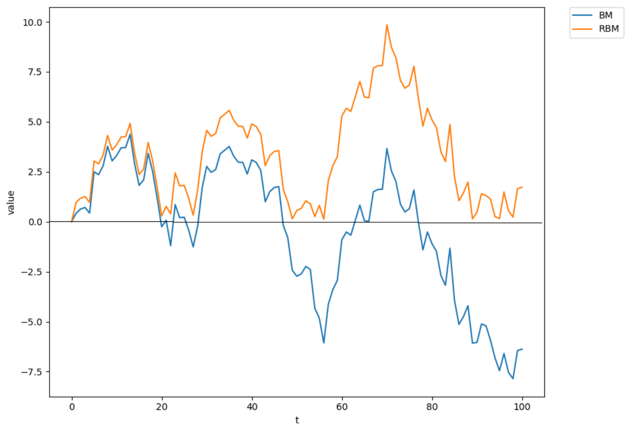


Figure 1.1.: An example of RBM and its driving Brownian motion sample paths

In this thesis, we focus exclusively on the one-dimensional RBM with diffusion coefficient $\sigma_t : t \geq 0$ equal to 1 for all t without loss of generality.

1.1.2 Distribution functions of RBM

The cumulative density function (CDF) of an RBM with constant drift μ, diffusion coefficient σ and starting point $X_0 = y$ is [6]:

$$P_{X_t}(x) = 1 - \Phi(\frac{-x + y + \mu t}{\sigma\sqrt{t}}) - e^{2\mu x/\sigma^2}\Phi(\frac{-x - y - \mu t}{\sigma\sqrt{t}}),$$

where Φ is the standard normal CDF. The probability density function (PDF) can be easily calculated by taking the derivative of CDF with respect to x:

$$p_{X_t}(x) = \frac{1}{\sigma\sqrt{t}}\phi(\frac{-x + y + \mu t}{\sigma\sqrt{t}}) - e^{2\mu x/\sigma^2}\frac{2\mu}{\sigma^2}\Phi(\frac{-x - y - \mu t}{\sigma\sqrt{t}}) + \frac{e^{2\mu x/\sigma^2}}{\sigma\sqrt{t}}\phi(\frac{-x - y - \mu t}{\sigma\sqrt{t}}),$$

where ϕ stands for the standard normal PDF. For simplicity, we assume $y = 0$ and $\sigma = 1$ for the remainder of the thesis. We will also by a small abuse of notation write $p_{X_t}(x)$ to represent the conditional density $p_{X_t|X_0}(x|y)$.

When the drift μ_t is a time-dependent function, the analytical expression for the density function is much harder to compute in general, and typically involves time-ordered integrals. For a piecewise constant drift vector $\vec{\mu} : [\mu_1, \mu_2, ..., \mu_{\lfloor T/\Delta t \rfloor}]$ with a discretization time step of Δt, PDF of an RBM at time t could be calculated through integration of the joint distribution of $x_1, x_2, ..., x_{\lfloor t/\Delta t \rfloor}$:

$$\begin{aligned}
p_{X_t}(x) &= \int_0^\infty ... \int_0^\infty \int_0^\infty p_{X_1, X_2, ..., X_{\lfloor t/\Delta t \rfloor}}(x_1, x_2, ..., x_{\lfloor t/\Delta t \rfloor}) dx_1 dx_2 ... dx_{\lfloor t/\Delta t \rfloor} \\
&= \int_0^\infty ... \int_0^\infty \int_0^\infty p_{X_1}(x_1) p_{X_2|X_1}(x_2|x_1) p_{X_3|X_2,X_1}(x_3|x_2, x_1) \\
&\quad ... p_{X_{\lfloor t/\Delta t \rfloor}|X_{\lfloor t/\Delta t \rfloor -1}, X_{\lfloor t/\Delta t \rfloor -2}, ..., X_1}(x_{\lfloor t/\Delta t \rfloor}|x_{\lfloor t/\Delta t \rfloor -1}, x_{\lfloor t/\Delta t \rfloor -2}, ..., x_1) dx_1 dx_2 ... dx_{\lfloor t/\Delta t \rfloor} \\
&= \int_0^\infty ... \int_0^\infty \int_0^\infty p_{X_1}(x_1) p_{X_2|X_1}(x_2|x_1) ... p_{X_{\lfloor t/\Delta t \rfloor}|X_{\lfloor t/\Delta t \rfloor -1}}(x_{\lfloor t/\Delta t \rfloor}|x_{\lfloor t/\Delta t \rfloor -1}) \\
&\qquad dx_1 dx_2 ... dx_{\lfloor t/\Delta t \rfloor}
\end{aligned}$$

following from the strong Markov property of RBM [6, Proposition 1],

where strong Markov property of the RBM says that, denote the RBM X_t at time $T + t$, X_{T+t} as X_t^*, then a function $k(x) = \mathbb{E}_x[K(X^*)|X_{t,0\leq t}] = k(X_T)$ and does not depend on the history before X_T where $K(\cdot)$ is a $\mathbb{R} \to \mathbb{R}$ function.

2. PROBLEM DESCRIPTION AND PDE METHOD

2.1 Problem Definition

We are interested in solving stochastic optimization problems driven by RBM. The design variable in these settings is the drift function of the RBM. We consider an expected cost function:

$$J(T) = \mathbb{E}[\int_0^T X_t(\mu_t)dt] + G(\mathbb{E}[X_T(\mu_T)]),$$

where $G : \mathbb{R} \to \mathbb{R}$ is a given smooth function. The corresponding optimization problem is:

$$\min_{\mu_t \in \mathcal{A}} \quad \mathbb{E}[\int_0^T X_t(\mu_t)dt] + G(\mathbb{E}[X_T(\mu_T)]). \tag{2.1}$$

Here, $\mu_t : t \geq 0$ is in the function space $\mathcal{A}$. We will assume $\mathcal{A} \subset \mathcal{D}$, the space of functions that are right continuous with left limits (RCLL) or càdlàg. Observe that this is not a stochastic optimal control problem, since μ_t is optimized for at time zero, and is not chosen as a function of the current state of the process. For the rest of the thesis, we focus on the case of linear terminal cost where $G(\mathbb{E}[X_T(\mu_T)]) = C\mathbb{E}[X_T(\mu_T)]$ and $C \in \mathbb{R}$.

2.2 Convexity Analysis

The space of càdlàg functions is a topological vector space (specifically a Banach space) when one uses the supremum norm to verify the conditions of the vector space. Consequently, addition is a continuous operator under the supremum norm. Using this property, we will check the convexity of the objective function without a terminal cost. By definition, a function f is convex if $\forall \alpha \in [0,1]$, $\forall \mu_{1t}, \mu_{2t} \in \mathcal{A}$: $(1-\alpha)f(\mu_{1t}) + \alpha f(\mu_{2t}) \geq f((1-\alpha)\mu_{1t} + \alpha\mu_{2t})$.

Proposition 1 *A function which takes the form $f(\mu_t) = \mathbb{E}[\int_0^T X_t(\mu_t)dt]$ where X_t is a RBM and μ_t is the drift function is convex on the domain $\mathcal{A}$.*

Proof

$$
\begin{aligned}
&(1-\alpha)f(\mu_{1t}) + \alpha f(\mu_{2t}) - f((1-\alpha)\mu_{1t} + \alpha\mu_{2t}) \\
&= (1-\alpha)\mathbb{E}[\int_0^T (\mu_{1t} + B_t + L_t(\mu_{1t}))dt] + \alpha\mathbb{E}[\int_0^T (\mu_{2t} + B_t + L_t(\mu_{2t}))dt] \\
&- \mathbb{E}[\int_0^T ((1-\alpha)\mu_{1t} + \alpha\mu_{2t} + B_t + L_t((1-\alpha)\mu_{1t} + \alpha\mu_{2t}))dt] \\
&= \mathbb{E}[\int_0^T ((1-\alpha)L_t(\mu_{1t}) + \alpha L_t(\mu_{2t}) - L_t((1-\alpha)\mu_{1t} + \alpha\mu_{2t}))dt] \\
&(\textit{because only } L_t \textit{ is dependent on } \mu_t) \\
&= \mathbb{E}[\int_0^T (\sup_{0\le s\le t}(-(1-\alpha)(\mu_{1s} + B_s))^+ + \sup_{0\le s\le t}(-\alpha(\mu_{2s} + B_s))^+ \\
&- \sup_{0\le s\le t}(-(1-\alpha)\mu_{1s} - \alpha\mu_{2s} - B_s)^+)dt] \\
&(\textit{because } L_t = \sup_{0\le s\le t}(-\mu_s + \sigma B_s)^+) \\
&= \mathbb{E}[\int_0^T \max(\sup_{0\le s\le t}(-(1-\alpha)(\mu_{1s} + B_s)), 0) + \max(\sup_{0\le s\le t}(-\alpha(\mu_{2s} + B_s)), 0) \\
&- \max(\sup_{0\le s\le t}(-(1-\alpha)\mu_{1s} - \alpha\mu_{2s} - B_s), 0)dt] \\
&\ge \mathbb{E}[\int_0^T \max(\sup_{0\le s\le t}(-(1-\alpha)(\mu_{1s} + B_s)) + \sup_{0\le s\le t}(-\alpha(\mu_{2s} + B_s)), 0) \\
&- \max(\sup_{0\le s\le t}(-(1-\alpha)\mu_{1s} - \alpha\mu_{2s} - B_s), 0)dt] \\
&\ge \mathbb{E}[\int_0^T \max(\sup_{0\le s\le t}(-(1-\alpha)(\mu_{1s} + B_s) - \alpha(\mu_{2s} + B_s)), 0) \\
&- \max(\sup_{0\le s\le t}(-(1-\alpha)\mu_{1s} - \alpha\mu_{2s} - B_s), 0)dt] \\
&= 0
\end{aligned}
$$

■

The convexity of the objective function with a linear terminal cost $f(\mu) = \mathbb{E}[\int_0^T(\mu t + B_t + L_t)dt] + C\mathbb{E}[\mu_T + B_T + L_T]$ where C is a positive constant can be proven similarly as the objective function without a terminal cost. On the other hand, it is much harder to prove convexity when C is negative. Observe that,

$$
\begin{aligned}
&(1-\alpha)f(\mu_{1t}) + \alpha f(\mu_{2t}) - f((1-\alpha)\mu_1 + \alpha\mu_2)\\
&= (1-\alpha)\mathbb{E}[\int_0^T(\mu_{1t} + B_t + L_t(\mu_{1t}))dt] + \alpha\mathbb{E}[\int_0^T(\mu_{2t} + B_t + L_t(\mu_{2t}))dt]\\
&- \mathbb{E}[\int_0^T((1-\alpha)\mu_{1t} + \alpha\mu_{2t} + B_t + L_t((1-\alpha)\mu_{1t} + \alpha\mu_{2t}))dt]\\
&+ C\mathbb{E}[((1-\alpha)\mu_{1t} + \alpha\mu_{2t} + B_T + L_T((1-\alpha)\mu_{1t} + \alpha\mu_{2t}))\\
&- C(1-\alpha)(\mu_{1t} + B_T + L_T(\mu_{1t})) - C\alpha(\mu_{2t} + B_T + L_T(\mu_{2t}))]\\
&= \mathbb{E}[\int_0^T((1-\alpha)L_t(\mu_{1t}) + \alpha L_t(\mu_{2t}) - L_t((1-\alpha)\mu_{1t} + \alpha\mu_{2t}))dt]\\
&+ C\mathbb{E}[L_T((1-\alpha)\mu_{1t} + \alpha\mu_{2t}) - (1-\alpha)L_T(\mu_{1t}) - \alpha L_T(\mu_{2t})]\\
&= \mathbb{E}[\int_0^T((1-\alpha)L_t(\mu_{1t}) + \alpha L_t(\mu_{2t}) - L_t((1-\alpha)\mu_{1t} + \alpha\mu_{2t})\\
&- \frac{C}{T}((1-\alpha)L_T(\mu_{1T}) + \alpha L_t(\mu_{2T}) - L_T((1-\alpha)\mu_{1T} + \alpha\mu_{2T})))dt]\\
&\geq \mathbb{E}[\int_0^T((1-\alpha)L_t(\mu_{1t}) + \alpha L_t(\mu_{2t}) - \frac{C}{T}((1-\alpha)L_T(\mu_{1T}) + \alpha L_t(\mu_{2T}))\\
&+ (\frac{C}{T} - 1)L_t((1-\alpha)\mu_{1t} + \alpha\mu_{2t}))dt].
\end{aligned}
$$

Now, a sufficient condition for $f(\mu)$ to be convex in a convex domain $\mathcal{A}$ is that for $\forall\alpha \in [0,1]$, $\forall\mu_{1t}, \mu_{2t} \in \mathcal{A}$,

$$\begin{aligned}
&\mathbb{E}[(1-\alpha)L_t(\mu_{1t}) + \alpha L_t(\mu_{2t}) - \frac{C}{T}((1-\alpha)L_T(\mu_{1T}) + \alpha L_t(\mu_{2T}))\\
&+ (\frac{C}{T} - 1)L_t((1-\alpha)\mu_{1t} + \alpha\mu_{2t})]\\
&\geq 0\\
&\text{equivalently,}\\
&\frac{\mathbb{E}[(1-\alpha)L_t(\mu_{1t}) + \alpha L_t(\mu_{2t}) + (\frac{C}{T} - 1)L_t((1-\alpha)\mu_{1t} + \alpha\mu_{2t})]}{\mathbb{E}[(1-\alpha)L_T(\mu_{1T}) + \alpha L_t(\mu_{2T})]}\\
&\geq \frac{C}{T}
\end{aligned}$$

Given this sufficient condition of our objective function's convexity, an actual domain for μ_t where the condition is satisfied is rather hard to find. Indeed, it is straightforward to see that this condition is not satisfied even when the domain is convex, as the following example shows: take $\mu_{1t} = -100t$ and $\mu_{2t} = -200t$, $T = 1$, $\alpha = 0.5$ and $\frac{C}{T} = 3$, then,

$$\begin{aligned}
&(1-\alpha)f(\mu_{1t}) + \alpha f(\mu_{2t}) - f((1-\alpha)\mu_1 + \alpha\mu_2)\\
&= \mathbb{E}[\int_0^T (1-\alpha)X_t(\mu_{1t}) + \alpha X_t(\mu_{2t}) - X_t((1-\alpha)\mu_{1t} + \alpha\mu_{2t})\\
&- 3((1-\alpha)X_T(\mu_{1t}) + \alpha X_T(\mu_{2t}) - X_T((1-\alpha)\mu_{1t} + \alpha\mu_{2t}))dt]
\end{aligned}$$

The plot for the integrand: $(1-\alpha)X_t(\mu_{1t}) + \alpha X_t(\mu_{2t}) - X_t((1-\alpha)\mu_{1t} + \alpha\mu_{2t}) - 3((1-\alpha)X_T(\mu_{1t}) + \alpha X_T(\mu_{2t}) - X_T((1-\alpha)\mu_{1t} + \alpha\mu_{2t}))$ is as following.

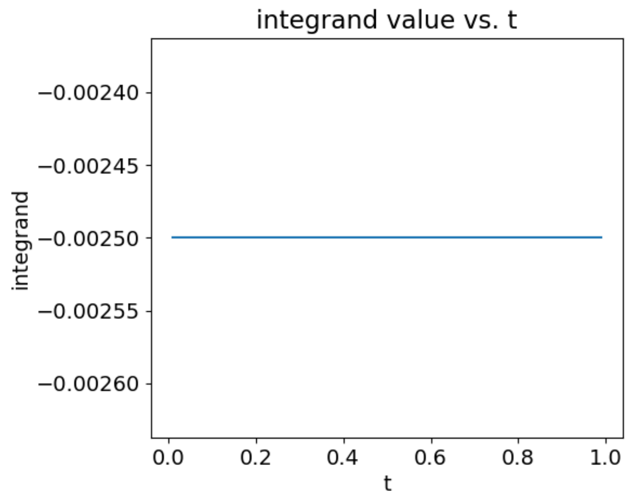


Figure 2.1.: integrand value vs. time

As it can be observed from the plot, the integrand is always negative within the integration limit $t \in [0, 1]$ so that the integral is also negative. We conclude that:

$$(1-\alpha)f(\mu_{1t}) + \alpha f(\mu_{2t}) - f((1-\alpha)\mu_1 + \alpha\mu_2) < 0$$

and $f(\mu_t)$ is not a convex function.

In general, the objective function with a negative linear terminal cost is not necessarily convex within the domain of interest and our algorithms may converge to a local optimum or even a saddle point.

2.3 Deriving PDE Formulation

The optimization objective (2.1) can be viewed as a deterministic optimal control problem where $\mathbb{E}X_t$ is the space variable, t is the time variable and the drift function $\mu(\cdot)$ is the control variable. In the following derivation, we denote $\mathbb{E}X_t$ as X_t. We define the optimal cost function $U(x,t)$ as

$$U(x,t) = \inf_{\mu(\cdot)\in\mathcal{A}} \{ \int_t^T r(X_s, \mu_s) ds + g(X_T) \},$$

where $r(X_t, \mu_t) = X_t^{\mu(\cdot)}$ and $g(X_T) = G(X_T)$, which satisfies the optimality condition in [7, Theorem 1]:

$$U(x,t) = \inf_{\mu(\cdot)\in\mathcal{A}} \{ \int_t^{t+\Delta t} r(X_s, \mu_s) ds + U(x, t+\Delta t) \}.$$

By [7, Theorem 2], dynamic of $U(x,t)$ can be represented by a Hamilton-Jacobi-Bellman (HJB) equation in the form,

$$\frac{\partial U(x,t)}{\partial t} + \min_{\mu \in A} \{ f(x,\mu) \nabla_x U(x,t) + r(x,\mu) \} = 0.$$

with a terminal condition:

$$U(x,T) = g(x).$$

Now we informally derive the actual HJB equation for our cost function, observe that,

$$U(x,t) = \inf_{\mu(\cdot)\in\mathcal{A}} \{ \int_t^{t+\Delta t} r(X_s, \mu_s) ds + U(x, t+\Delta t) \},$$

and assuming that there exist such a optimal drift function $\mu^*(\cdot) \in \mathcal{A}$ that achieves the infimum, it is easy to see that,

$$U(x,t) = \inf_{\mu(\cdot)\in\mathcal{A}} \{ \int_t^{t+\Delta t} r(X_s, \mu_s) ds \} + U(x, t+\Delta t)$$

and:

$$\frac{U(x,t+\Delta t) - U(x,t)}{\Delta t} = -\frac{\inf_{\mu(\cdot)\in\mathcal{A}} \{ \int_t^{t+\Delta t} r(X_s, \mu_s) ds \}}{\Delta t}.$$

Let $\Delta t \to 0$:

$$\frac{\partial U(x,t)}{\partial t} + \frac{\partial U(x,t)}{\partial x} \frac{dX}{dt} = -\min_{\mu \in \mathbb{R}} \{ r(x,\mu) \}$$

Set the domain of the PDE to be $t \in (0,T)$ and $X \in (0,+\infty)$. For the reason that t does not depend on $\mu(t)$, the term $\frac{\partial U(x,t)}{\partial t}$ could be left outside the optimization. The resulting HJB equation is:

$$\min_{\mu \in \mathbb{R}} \left\{ \frac{\partial U(x,t)}{\partial t} + \frac{\partial U(x,t)}{\partial x} \frac{dX}{dt} + r(x,\mu) \right\} = 0$$

with the terminal condition:

$$U(x,T) = G(x).$$

However, solving this HJB equation requires the knowledge of system dynamic:$\frac{d\mathbb{E}X.}{dt}$. In this case, the system dynamic is calculated as: say that the discretization of $(0 \leq t \leq T)$ is $[\tau_1, \tau_2, ..., \tau_T]$, and for $t \in [\tau_{i-1}, \tau_i]$,

$$\frac{d\mathbb{E}[X_t|X_0 = z]}{dt} = \int_{x_{\tau_{i-1}}} (\int_{x_{\tau_t}} x_{\tau_i} \frac{d}{dt} P_{\tau_{i-1},t}(x_{\tau_{i-1}}, dx_{\tau_i})) P_{0,\tau_{i-1}}(z, dx_{\tau_{i-1}})$$

where

$$\begin{aligned}
\frac{d}{dt} P_{\tau_{i-1},t}(y,x) &= \frac{d}{dt}\frac{d}{dx} P(Z_t \leq x | Z_{\tau_{i-1}}) \\
&= \frac{d}{dt}\frac{d}{dx}\{\Phi(\frac{x - \mu_{\tau_{i-1}}(t-\tau_{i-1})}{\sigma\sqrt{t-\tau_{i-1}}}) - e^{\frac{2\mu_{\tau_{i-1}}x}{\sigma^2}} \Phi(\frac{-x - \mu_{\tau_{i-1}}(t-\tau_{i-1})}{\sigma\sqrt{t-\tau_{i-1}}})\} \\
&= \frac{d}{dx}\{\phi(\frac{x - \mu_{\tau_{i-1}}(t-\tau_{i-1})}{\sigma\sqrt{t-\tau_{i-1}}}) \\
&\frac{-\mu_{\tau_{i-1}}\sigma\sqrt{t-\tau_{i-1}} - (x - \mu_{\tau_{i-1}}(t-\tau_{i-1}))\frac{\sigma}{2}(t-\tau_{i-1})^{-1/2}}{\sigma^2(t-\tau_{i-1})} \\
&- e^{\frac{2\mu_{\tau_{i-1}}x}{\sigma^2}} \phi(\frac{-x - \mu_{\tau_{i-1}}(t-\tau_{i-1})}{\sigma\sqrt{t-\tau_{i-1}}}) \\
&\frac{-\mu_{\tau_{i-1}}\sigma\sqrt{t-\tau_{i-1}} - (-x - \mu_{\tau_{i-1}}(t-\tau_{i-1}))\frac{\sigma}{2}(t-\tau_{i-1})^{-1/2}}{\sigma^2(t-\tau_{i-1})} \\
&= \phi''(\frac{x - \mu_{\tau_{i-1}}(t-\tau_{i-1})}{\sigma\sqrt{t-\tau_{i-1}}}) I(t,x) \frac{1}{\sigma\sqrt{t-\tau_{i-1}}} \\
&+ \phi(\frac{x - \mu_{\tau_{i-1}}(t-\tau_{i-1})}{\sigma\sqrt{t-\tau_{i-1}}}) \frac{d}{dx} I(t,x) \\
&- e^{\frac{2\mu_{\tau t-1}x}{\sigma^2}} \frac{2\mu_{taut-1}}{\sigma^2} \phi(\frac{-x - \mu_{\tau_{i-1}}(t-\tau_{i-1})}{\sigma\sqrt{t-\tau_{i-1}}}) I(t,x) \\
&- e^{\frac{2\mu_{\tau t-1}x}{\sigma^2}} \phi'(\frac{-x - \mu_{\tau_{i-1}}(t-\tau_{i-1})}{\sigma\sqrt{t-\tau_{i-1}}}) I(t,x) \frac{1}{\sigma\sqrt{t-\tau_{i-1}}} \\
&- e^{\frac{2\mu_{\tau t-1}x}{\sigma^2}} \phi(\frac{-x - \mu_{\tau_{i-1}}(t-\tau_{i-1})}{\sigma\sqrt{t-\tau_{i-1}}}) \frac{d}{dx} I(t,x)\}.
\end{aligned}$$

The closed form of the system dynamic as derived above is near-impossible to compute analytically. The HJB equation cannot be solved without the expression of the system dynamic. This issue leads us to develop an alternative Monte Carlo simulation based gradient descent method which We present in the next chapter.

3. RBM SIMULATION

In this chapter, we briefly summarize the RBM simulation algorithm we used in this thesis. Recall our objective function:

$$J(T) = \mathbb{E}[\int_0^T X_t(\mu_t)dt] + G(\mathbb{E}[X_T(\mu_T)]).$$

As noted before, the expectation of the integral is almost impossible to calculate when $(\mu_t : t \geq 0)$ is a time-varying function, in general. The PDE method discussed in the previous chapter requires the creation of a "mesh of grids," and the resulting calculations suffer from the curse of dimensionality. An alternative approach, that avoids the creation of the mesh is to use Monte Carlo sampling and exploit the connection between parametric PDE's and the Feynman-Kac Theorem. Choosing an arbitrary time step as Δt, drift function $\mu(t)$ is discretized into a vector $\vec{\mu}$: $[\mu_1, \mu_2, ..., \mu_{\lfloor T/\Delta t \rfloor}]$. Now, expectations of the RBM can be estimated using Monte Carlo simulation of sample paths, and the objective function therefore becomes:

$$\tilde{J}(T) = \hat{\mathbb{E}}[\sum_{i=0}^{\lfloor T/\Delta t \rfloor} X_i(\mu_i)\Delta t)] + G(\hat{\mathbb{E}}[(X_T(\mu_T)])$$

where $\hat{\mathbb{E}}$ is the Monte Carlo estimation of the expectation. Denote N as the Monte Carlo sample size, $\{Y_1, Y_2, ..., Y_N\}$ as N Monte Carlo samples of a random variable X, $f(\cdot)$ as any function, the estimator $\hat{\mathbb{E}}f(X)$ for $\mathbb{E}f(X)$ is:

$$\hat{\mathbb{E}}f(X) = \frac{1}{N}\sum_{i=1}^{N} f(Y_i)$$

3.1 Simulation method

Note that assuming the piece-wise constant drift vector $\vec{\mu} : [\mu_1, \mu_2, ..., \mu_{\lfloor T/\Delta t \rfloor}]$, we know the probability distribution of a RBM at time $(t : 0 \leq t \leq T)$ can be calculated using the iterated integral:

$$p_{X_t}(x) = \int_0^\infty ... \int_0^\infty \int_0^\infty p_{X_1}(x_1) p_{X_2|X_1}(x_2|x_1) ... p_{X_{\lfloor t/\Delta t \rfloor}|X_{\lfloor t/\Delta t \rfloor - 1}}(x_{\lfloor t/\Delta t \rfloor}|x_{\lfloor t/\Delta t \rfloor - 1}) \\ dx_1 dx_2 ... dx_{\lfloor t/\Delta t \rfloor}$$

as we derived in section 1.1.2. A common approach to simulating a sample path of a process Y_t with discretization time step Δt is to simulate $Y_{\Delta t}$, $Y_{2\Delta t}$, $\cdots$ as random variates drawn from the corresponding probability distributions. However, classic approaches to simulate from distribution are hard (if not impossible) to implement for the RBM. For instance, with inversion sampling, the inverse of cumulative density function at time t, ${F_{X}}_t^{-1}(x)$, is required to simulate the RBM sample X_t. However, the analytical expression of this inverse function is nearly impossible to compute. Alternatively, with rejection sampling, a probability density function $g_{Xt}(x)$ that satisfies $\frac{p_{Xt}(x)}{g_{Xt}(x)} \leq C \; \forall 0 \leq X_t \leq \infty$, $C > 1$ is required. Finding an appropriate $g_{Xt}(x)$ is complicated in the case of RBM. Further, if $g_{Xt}(x)$ is chosen so that $C >> 1$, the method suffers from a high rejection probability and low efficiency. Usually, a stochastic process which is the solution of a stochastic differential equation can be simulated using the Euler-Maruyama method. However, if we simulate RBM sample paths using the Euler-Maruyama, a systematic error will be incurred at the discretization time grid. Asmussen, Glynn and Pitman, on the other hand, introduced an exact simulation method for RBM [4]. They fully utilized the Markov properties of RBM's driving Brownian motion and RBM's running max to obtain this algorithm. We summarize this method below.

Recall that the expression for a RBM at time t is

$$X_t = W_t + L_t,$$

where $W_t = \mu_t + B_t$, $L_t = \sup_{0\leq s\leq t}(-\mu_s - B_s)^+$, μ_t is the drift function and B_s is the driving Brownian motion. Note that $\sup_{0\leq s\leq t}(-\mu_s - B_s)^+$ has the same distribution as $\sup_{0\leq s\leq t}(-\mu_s + B_s)^+$ because of symmetry and $\sup_{0\leq s\leq t}(-\mu_s + B_s)^+$ is the running maximum of RBM's driving Brownian motion with drift $-\mu_t$. By [4, Lemma 3]:

$$P(\max_{0\leq t\leq T} B(t) - y \leq x | B(T) = y) = 1 - e^{-2x(y-x)/T}.$$

Given this CDF, we can simulate the reflection term $L_t = \sup_{0\leq s\leq t}(-\mu_s - B_s)^+ \overset{D}{=} \sup_{0\leq s\leq t}(-\mu_s + B_s)^+$ simply by inversion sampling. The driving Brownian motion W_t can be simulated from the normal distribution $\mathcal{N}(\mu_t, \sqrt{t})$. These two samples added is the RBM sample at time t. Proven in [4], the approximation error vanishes at the discretization point and the convergence rate estimating $\mathbb{E}f(X(t))$ is of order $c^{-1/2}$ with this algorithm. Here is an example of this exact simulation method to the contrast of the Euler-Maruyama method. It is obvious that the Euler-Maruyama

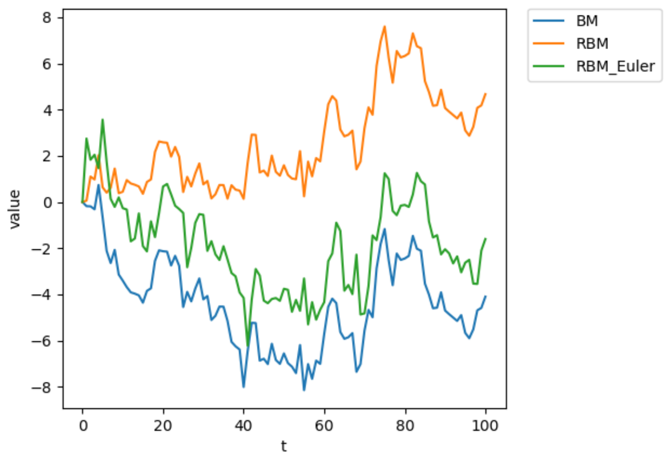


Figure 3.1.: An example of RBM sample paths simulated using exact simulation and the Euler-Maruyama.

method incurs an error and the resulting RBM sample path can even reach a negative level. Below summarizes this sampling algorithm.

RBM Simulation Algorithm

Denote the current Brownian motion process value as B, RBM value as X, the current time as t, time step as Δt and simulation time horizon as T.

Algorithm 1: RBM sample paths simulation algorithm [4]

```
1 Initialize $t \leftarrow 0$, $W \leftarrow 0$, $X \leftarrow 0$, $M \leftarrow 0$, $\vec{X} \leftarrow$ an empty array
2 Generate $(T_1, T_2)$ with $\tau = \Delta t$
3 Let $t \leftarrow t + \Delta t$, $M \leftarrow max(M, W + T_2)$, $W \leftarrow W + T_1$,$X \leftarrow M - W$, append
    $X$ to $\vec{X}$
4 if $t < T$ then return to step 3;
5 else stop algorithm and return $\vec{X}$;
```

Where

$$(T_1, T_2) = (W(\tau), \max_{0 \le t \le \tau} W(t))$$

and for $U \sim uniform(0, 1)$:

$$\max_{0 \le t \le \tau} W(t)) \leftarrow \frac{W(\tau)}{2} + \frac{\sqrt{W(\tau)^2 - 2(U)}}{2}.$$

4. GRADIENT DESCENT METHOD

In the previous chapter, we discretized time and formed a new objective function:

$$\tilde{J}(T) = \hat{\mathbb{E}}[\sum_{i=0}^{\lfloor T/\Delta t \rfloor} X_i(\mu_i)\Delta t)] + \hat{\mathbb{E}}[G(X_T(\mu_T))]$$

computed using the Monte Carlo simulation of RBM sample paths. In this project, we focus on when $G(\cdot)$ is a linear function so that

$$\tilde{J}(T) = \hat{\mathbb{E}}[\sum_{i=0}^{\lfloor T/\Delta t \rfloor} X_i(\mu_i)\Delta t)] + \hat{\mathbb{E}}[CX_T(\mu_T)],$$

where C is a constant. The sample average approximation (SAA) optimization problem is then:

$$\min_{\vec{\mu} \in \mathbb{R}^{\lfloor T/\Delta t \rfloor}} \tilde{J}(T).$$

Now we consider the problem of numerically optimizing this new objective. In this thesis, we focus on first-order methods for numerically computing the optimal drift. Specifically, we consider the gradient descent (GD) and stochastic approximation (SA) methods [8] [9]. In this chapter, we present our algorithms applying gradient descent to this new objective function, and particularly, focus on the gradient estimation schemes. We start with a summary of the gradient descent method.

4.1 Gradient Descent Method

The GD method iteratively computes an estimate of the optimal drift by moving in the greatest decrease direction of the gradient at each iteration of the algorithm. It is well known [9] that the GD/SA method is guaranteed to converge to the optimal drift when the objective is convex and differentiable. When these conditions are violated, the GD/SA method is only guaranteed to converge to a first-order critical point. As demonstrated before, our objective is, in general, non-convex if the terminal condition is negative. A further complication that arises in our setting is the fact that the gradient is not known in closed form. Recall the gradient of the objective function with respect to the drift vector is:

$$\nabla_{\vec{\mu}}\tilde{J}(T) = [\frac{d\tilde{J}(T)}{d\mu_1}, \frac{d\tilde{J}(T)}{d\mu_2}, ..., \frac{d\tilde{J}(T)}{d\mu_{\lfloor T/\Delta t\rfloor}}],$$

This gradient is unknown analytically, in general. In the remainder of this chapter, we present two gradient approximation methods.

4.2 SPSA Method

4.2.1 Introduction

A classic approach to approximating the gradient from sample paths is the Kiefer-Wolfowitz algorithm [9]. The estimator for the *ith* gradient entry using the Kiefer-Wolfowitz algorithm is:

$$\hat{\nabla}_{\vec{\mu_i}}\tilde{J}_i \approx \frac{\tilde{J}(\vec{\mu_i} + c_i\Delta\vec{e_i}) - \tilde{J}(\vec{\mu_i} - c_i\Delta\vec{e_i})}{2c_i},$$

where c_i is a positive scalar and $c_i \to 0$ as the iteration number goes to ∞ and $\vec{\Delta e_i}$ is a random vector whose *ith* entry is non-zero and all other entries are 0. However, because of the fact that for each entry of the gradient estimator $\Delta\vec{\mu_i}$ is only perturbed in one dimension, this estimator turns out to have comparatively large error. The simultaneous perturbation stochastic approximation (SPSA) method is proven to converge faster than the standard Kiefer-Wolfowitz method by adding perturbations in each dimension of every entry of the gradient estimator [10] [11] [12]. The SPSA estimator is simply:

$$\hat{\nabla}_{\vec{\mu_i}}\tilde{J}_i \approx \frac{\tilde{J}(\vec{\mu_i} + c_i\Delta\vec{\mu_i}) - \tilde{J}(\vec{\mu_i} - c_i\Delta\vec{\mu_i})}{2c_i\Delta\vec{\mu_i}},$$

where as before, c_i is a positive scalar which goes to 0 as the iteration number goes to ∞ and $\Delta\vec{\mu_i}$ is a random vector which is non-zero for all dimensions.

4.2.2 Validation for the SPSA Estimator

The SPSA gradient estimator for the *nth* iteration in the gradient descent method is:

$$\hat{\nabla}_{\vec{\mu_n}} \tilde{J} \approx \frac{\tilde{J}(\vec{\mu_n} + c_n \Delta\vec{\mu_n}) - \tilde{J}(\vec{\mu_n} - c_n \Delta\vec{\mu_n})}{2c_n \Delta\vec{\mu_n}}$$

where $\vec{c_n}$ is a positive scalar which goes to 0 as $n \to \infty$ and $\Delta\vec{\mu_n}$ is a random directional vector drawn from a uniform distribution. The convergence of the estimator is discussed by Spall in [10]. The bias of the estimator is proven to be of the order of $O(c_n^2)$, and as $c_n \to 0$ and the bias also goes to 0. In [10, Proposition 1], the strong convergence of the estimator and the estimator bias is proved.

4.2.3 Algorithms

Gradient Approximation

Combined with the RBM simulation algorithm, $\hat{\nabla}_{\vec{\mu}}\tilde{J}$ or $\hat{\nabla}_{\vec{\mu}}\hat{J}$ can be estimated using Algorithm 2.

Algorithm 2: Approximating gradient with SPSA

```
1 Generate random $\Delta\vec{\mu}$ with each entry $\sim Uniform(0,1)$
2 Generate $c$ as from any sequence that goes to zero as the gradient descent
    iteration number goes to $\infty$
3 Generate $N$ RBM sample paths with **Algorithm 1** with $\vec{\mu} + c\Delta\vec{\mu}$
4 Generate $N$ RBM sample paths with **Algorithm 1** with $\vec{\mu} - c\Delta\vec{\mu}$
5 for $t \in [\Delta t, 2\Delta t, ..., T]$ do
6     $C_1 \leftarrow \frac{1}{N}\sum_{i=1}^{N} \hat{X}_t(\vec{\mu} + c\Delta\vec{\mu})$, State $C_2 \leftarrow \frac{1}{N}\sum_{i=1}^{N} \hat{X}_t(\vec{\mu} - c\Delta\vec{\mu})$
7     $\hat{\nabla}_{\vec{\mu}}\mathbb{E}[X_t] \leftarrow \frac{C_1 - C_2}{2c\Delta\vec{\mu}}$
8 end
9 $\hat{\nabla}_{\vec{\mu}}\tilde{J} \leftarrow \sum_{t=1}^{T} \hat{\nabla}_{\vec{\mu}}\mathbb{E}[X_t]$ and return $\hat{\nabla}_{\vec{\mu}}\tilde{J}$
```

Optimization

The gradient estimation from Algorithm 2 is now combined with GD to optimize $\tilde{J}(\cdot)$.

Algorithm 3: Gradient descent optimization with SPSA

```
1 Input an arbitrary drift, $\vec{\mu}_{arbitrary}$ and step size, $\alpha$
2 Initialize a starting drift: $\vec{\mu_0} \leftarrow \vec{\mu}_{arbitrary}$
3 while the stopping criterion is not met do
4     For the nth iteration, estimate $\hat{\nabla}_{\vec{\mu_n}} \tilde{J}$ using Algorithm 2
5     $\vec{\mu_{n+1}} \leftarrow \vec{\mu_n} - \alpha \hat{\nabla}_{\vec{\mu_n}} \tilde{J}$
6     $n \leftarrow n + 1$
7 end
8 Return the optimal drift vector $\vec{\mu_n}$
```

4.3 MCGD Method

4.3.1 Introduction

Besides the classic SPSA method, there is another way to estimate the gradient of the objective function $\tilde{J}(\cdot)$. We denote the RBM at the jth time epoch as X_j and the μ_i as the ith drift vector entry. Observe that, for $i < \lfloor \frac{T}{\Delta t} \rfloor$, the ith entry of the gradient vector can be written as

$$\frac{d\tilde{J}(T)}{d\mu_i} = \sum_{j=1}^{\lfloor T/\Delta t \rfloor} \frac{d\mathbb{E}X_j}{d\mu_i}\Delta t + (C - \Delta t)\frac{d\mathbb{E}X_{\lfloor T/\Delta t \rfloor}}{d\mu_i},$$

where $\frac{d\mathbb{E}X_j}{d\mu_i}$ is the integral

$$\frac{d\mathbb{E}X_j}{d\mu_i} = \frac{d}{d\mu_i}(\int_0^\infty p_{X_j}(x)x dx).$$

Here, $p_{X_j}(x)x$ is a Lebesgue-integrable function of x for each drift at the ith time step $\mu_i \in \mathbb{R}$, and the derivative $\frac{d}{d\mu_i}(p_{X_j}(x)x)$ exists for all $\mu_i \in \mathbb{R}$. However, in order to interchange the integral and the derivative, by Leibniz's integral rule, we must show that the derivative of the integrand is bounded above by an integrable function. Say the starting point of the RBM is $X_0 = 0$, the time discretization $\Delta t = 1$, denote the standard normal PDF as ϕ and the standard normal CDF as Φ. Then, the derivative of the RBM density function for the first time step with respect to μ_1 is:

$$\begin{aligned}
\frac{d}{d\mu_1}(p_{X_1}(x)) &= \frac{d}{d\mu_1}(\phi(-x+\mu_1) - e^{2\mu_1 x}2\mu_1\Phi(-x-\mu_1) + e^{2\mu_1 x}\phi(-x-\mu_1)) \\
&= \phi'(-x+\mu_1) - e^{2\mu_1 x}4x\mu_1\Phi(-x-\mu_1) - e^{2\mu_1 x}2\Phi(-x-\mu_1) \\
&\quad\quad + e^{2\mu_1 x}2\mu_1\phi(-x-\mu_1) + e^{2\mu_1 x}2x\phi(-x-\mu_1) - e^{2\mu_1 x}\phi'(-x-\mu_1) \\
&\le \phi'(-x+\mu_1) - e^{2\mu_1 x}\phi'(-x-\mu_1) + 2e^{2\mu_1 x}\phi(-x-\mu_1)(\mu_1+x) \\
&\quad - 4\mu_1 x e^{2\mu_1 x}\Phi(-x-\mu_1) \\
&= \frac{(x-\mu_1)}{\sqrt{2\pi}}e^{\frac{-(x-\mu_1)^2}{2}} + \frac{3}{\sqrt{2\pi}}(x+\mu_1)e^{\frac{-(x^2+\mu_1^2)}{2}} - 4\mu_1 x e^{2\mu_1 x}\Phi(-x-\mu_1) \\
&=: I(x,\mu_1) - 4\mu_1 x e^{2\mu_1 x}\Phi(-x-\mu_1) \\
&\le I(x,\mu_1) + \max(0, -4\mu_1 x e^{2\mu_1 x}\Phi(-x-\mu_1))
\end{aligned}$$

Observe that,

$$\begin{aligned}
\int_{x\ge 0} x|I(x,\mu_1)|dx &\le \frac{1}{\sqrt{2\pi}}\int_{x\ge 0} x|x-\mu_1|e^{\frac{-(x-\mu_1)^2}{2}}dx + \frac{3}{\sqrt{2\pi}}\int_{x\ge 0} x|x+\mu_1|e^{\frac{-(x^2+\mu_1^2)}{2}}dx \\
&\le \frac{1}{\sqrt{2\pi}}(\int_{x\ge 0} x^2 e^{\frac{-(x-\mu_1)^2}{2}}dx + |\mu_1|\int_{x\ge 0} x e^{\frac{-(x-\mu_1)^2}{2}}dx) \\
&\quad + \frac{3}{\sqrt{2\pi}}(\int_{x\ge 0} x^2 e^{\frac{-(x^2+\mu_1^2)}{2}}dx + |\mu_1|\int_{x\ge 0} x e^{\frac{-(x^2+\mu_1^2)}{2}}dx) \\
&\le (1+\mu_1^2) + \mu_1|\mu_1| + 3e^{-\mu_1^2/2}
\end{aligned}$$

On the other hand

$$\int_{x\ge 0} x\max(0, -4\mu_1 x e^{2\mu_1 x}\Phi(-x-\mu_1))dx \le \begin{cases} 0 & \text{if } \mu_1 \ge 0, \text{ and} \\ 4|\mu_1|\int_{x\ge 0} x^2 e^{2\mu_1 x}dx & \text{if } \mu_1 < 0, \end{cases}$$

where we have used the fact that $\Phi(\cdot) \leq 1$. Using the fact that $\int_{x\geq 0} x^2 e^{2\mu_1 x} dx = \frac{1}{4\mu_1^2}$ it follows that

$$\int_{x\geq 0} x \; |I(x,\mu_1) + \max(0, -4\mu_1 x e^{2\mu_1 x}\Phi(-x-\mu_1))|dx$$
$$\leq (1+\mu_1^2) + \mu_1|\mu_1| + 3e^{-\mu_1^2/2} + \frac{|\mu_1|}{\mu_1}\frac{1}{\mu_1} < \infty.$$

Thus, as long as $\mu_1 \in [-\infty, b)$ where $b < \infty$, we can claim integrability of the function $x(I(x,\mu_1) + \max(0, -4\mu_1 x e^{2\mu_1 x}\Phi(-x-\mu_1)))$. Therefore, the derivative $\frac{d}{d\mu_1}(xp_{X_1}(x))$ is bounded by an integrable function. We can straightforwardly generalize this result to all i and j's and interchange the derivative and the integral,

$$\frac{d\mathbb{E}X_j}{d\mu_i} = \int_0^\infty \frac{d}{d\mu_i}(p_{X_j}(x)x)dx,$$

where $p_{X_j}(x)$ is the probability density function (PDF) of the RBM at time $j\Delta t$.

This PDF, however, has a non-trivial analytical form. Recall that the PDF of an RBM is only known in closed form for constant drift coefficients, or homogeneous RBM's. Assuming constant drift coefficients for each sub-interval, the actual $p_{X_j}(x)$ function can be computed through a nested integral:

$$p_{X_j}(x) = \int_0^\infty \int_0^\infty ... \int_0^\infty p_{X_1}(x_1)p_{X_2|X_1}(x_2|x_1)...p_{X_j|X_{j-1}}(x_j|x_{j-1})dx_1dx_2...dx_{j-1}.$$

The expression for the derivative of the expectation depends on the relative ordering of i and j. It will be shown that the resulting expressions are nested integrals. We estimate these nested integrals using Monte Carlo sampling of sample paths of the RBM. Consequently, we term the GD algorithm using this estimator as the Monte Carlo-based gradient descent (MCGD) method.

4.3.2 MCGD Estimators

There are several cases to consider.

Case 1: $i > j$: Since future drifts do not affect the current process, the expectation satisfies,

$$\frac{d\mathbb{E}X_j}{d\mu_i} = 0.$$

Case 2: $1 = i = j$: Observe that

$$\begin{aligned}\frac{d\mathbb{E}X_j}{d\mu_i} &= \int_0^\infty \frac{d\log\ p_{X_1}(x_1)}{d\mu_1} p_{X_1}(x_1)x dx \\ &= \mathbb{E}[\frac{d\log\ p_{X_1}(x_1)}{d\mu_1} X_1],\end{aligned}$$

with the corresponding estimator,

$$\frac{d\hat{\mathbb{E}}X_j}{d\mu_i} = \frac{1}{N}\sum_{n=1}^{N} \frac{d\log\ p_{X_1}(x_{1_n})}{d\mu_1} x_{1_n}.$$

Case 3: $1 = i < j$:

$$\begin{aligned}\frac{d\mathbb{E}X_j}{d\mu_i} &= \int_0^\infty ... \int_0^\infty \frac{d\big(\log\ p_{X_1}(x_1)\big)}{d\mu_i} p_{X_1}(x_1) \prod_{n=1}^{j-1} p_{X_{n+1}|X_n}(x_{n+1}|x_n) x_j dx_1 dx_2 ... dx_j \\ &= \mathbb{E}_{X_1}[\frac{d\log\ p_{X_1}(x_1)}{d\mu_1} \mathbb{E}[X_j|X_1]],\end{aligned}$$

with the corresponding estimator,

$$\frac{d\hat{\mathbb{E}}X_j}{d\mu_i} = \frac{1}{N}\sum_{m=1}^{N} \Big(\frac{d\big(\log\ p_{X_1}(x_{1_m})\big)}{d\mu_1} \big(\frac{1}{N}\sum_{n=1}^{N} x_{j_{m,n}}\big)\Big).$$

Case 4: $1 < i = j$: The gradient entry is,

$$\begin{aligned}\frac{d\mathbb{E}X_j}{d\mu_i} &= \int_0^\infty ... \int_0^\infty \frac{d\big(\log\ p_{X_i|X_{i-1}}(x_i|x_{i-1})\big)}{d\mu_i} p_{X_1}(x_1) \prod_{n=1}^{i-1} p_{X_{n+1}|X_n}(x_{n+1}|x_n) x_i dx_1 dx_2 ... dx_i \\ &= \mathbb{E}[\frac{d\log\ p_{X_i|X_{i-1}}(x_i|x_{i-1})}{d\mu_i} X_i],\end{aligned}$$

with the corresponding estimator,

$$\frac{d\hat{\mathbb{E}}X_j}{d\mu_i} = \frac{1}{N}\sum_{n=1}^{N} \Big(\frac{d\log\ p_{X_i|X_{i-1}}(x_{i_n}|x_{i-1_n})}{d\mu_i} x_{i_n}\Big).$$

Case 5: for$1 < i < j$: Finally we have the gradient,

$$\begin{aligned}\frac{d\mathbb{E}X_j}{d\mu_i} &= \int_0^\infty ... \int_0^\infty \frac{d\big(\log\ p_{X_i|X_{i-1}}(x_i|x_{i-1})\big)}{d\mu_i} p_{X_1}(x_1) \prod_{n=1}^{j-1} p_{X_{n+1}|X_n}(x_{n+1}|x_n) x_j dx_1 dx_2 ... dx_j \\ &= \mathbb{E}_{X_{i-1}}\mathbb{E}_{X_i}[\frac{d\log\ p_{X_i|X_{i-1}}(x_i|x_{i-1})}{d\mu_i} \mathbb{E}[X_j|X_i]]],\end{aligned}$$

and the corresponding estimator,

$$\frac{d\hat{\mathbb{E}}X_j}{d\mu_i} = \frac{1}{N}\sum_{m=1}^{N} \Big(\frac{d\log\ p_{X_i|X_{i-1}}(x_{i_m}|x_{i-1_m})}{d\mu_i} \big(\frac{1}{N}\sum_{n=1}^{N} x_{j_{m,n}}\big)\Big).$$

4.3.3 Validation for MCGD Estimator

Observe that the MCGD estimator is a nested expectation estimator. Consider the most common case $1 < i < j$ as an example; the corresponding element is:

$$\frac{d\hat{\mathbb{E}X_j}}{d\mu_i} = \frac{1}{N}\sum_{m=1}^{N}\Big(\frac{d\log\ p_{X_i|X_{i-1}}(x_{i_m}|x_{i-1_m})}{d\mu_i}\big(\frac{1}{N}\sum_{n=1}^{N} x_{j_{m,n}}\big)\Big).$$

This nested Monte Carlo estimator structure is discussed by Rainforth et al. [13] As shown in [13, Theorem 2], for a Monte Carlo estimator with the form

$$\hat{\theta} = \frac{1}{M}\sum_{m=1}^{M}\Big(f(\hat{I}_m)\Big) = \frac{1}{M}\sum_{m=1}^{M}\Big(f\big(\frac{1}{N}\sum_{n=1}^{N} x_{j_{m,n}}\big)\Big),$$

where f is a given function, $I_m = \mathbb{E}[X]$ for all $m \in \{1, 2, ...M\}$ and $\hat{I}_m = \frac{1}{N}\sum_{n=1}^{N} x_{j_{m,n}}$, the sufficient condition for the estimator to converge almost surely to the true parameter is that $|f(\hat{I}_m) - f(I_m)| \to 0$ as $N \to \infty$. It is clear that $\frac{1}{N}\sum_{n=1}^{N} x_{j_{m,n}}$ is an unbiased estimator for I_m. We still need to prove that, in our case, $f(\cdot)$ is continuous on its domain to assert that the estimator $\hat{\theta}$ will converge to θ. We check this condition in the following part.

$$\begin{aligned} f(x) &= \frac{d\log\ p_{X|Y}(x|y)}{d\mu_i} \\ &= ((-(-x+y+\mu) * \phi(-x+y+\mu) - (2e^{2\mu x} + 4\mu x e^{2\mu x})\Phi(-x-y-\mu) \\ &- e^{2\mu x}(-(-x-y-\mu))\phi(-x-y-\mu) + 2\mu e^{2\mu x} + 2xe^{2\mu x}\phi(-x-y-\mu))) \\ &/(\phi(-x+y+\mu) - 2\mu e^{2\mu x}\Phi(-x-y-\mu) + e^{2\mu x}\phi(-x-y-\mu)) \end{aligned}$$

This function is essentially a quotient of two continuous functions.

$$f(x) = \frac{g(x)}{\lambda(x)}$$

where $g(x)$ and $\lambda(x)$ are compositions of the product, the sum and the difference of continuous functions. Because that these operations on continuous functions will maintain the continuity, $g(x)$ and $\lambda(x)$ can be easily proved to be continuous. The denominator $\lambda(x)$ is the probability density function of the RBM process. By definition, $\lambda(x) \neq 0$ for all x in its domain. Therefore, $f(x)$ is a continuous function and $\hat{\theta} \to \theta$ as the Monte Carlo sample size increases.

4.3.4 Simulation for MCGD Method

Observe that each element would require N initial sample paths suppose there are two layers of averaging in the estimator, the number of sample path branches will be N^2 when the second layer of averaging happens. Consider case 5 as an example; the estimator is $\frac{d\hat{\mathbb{E}X_j}}{d\mu_i} = \frac{1}{N}\sum_{m=1}^{N}\left(\frac{d\log\, p_{X_i|X_{i-1}}(x_{i_m}|x_{i-1_m})}{d\mu_i}\left(\frac{1}{N}\sum_{n=1}^{N} x_{j_{m,n}}\right)\right)$. There will initially be N sample paths. From time j, each sample paths will have N sample path branches so that the total number of samples for each time step is N^2. An advantage of the MCGD estimator is that it avoids the curse the dimensionality, unlike dynamic programming. Specifically, estimating $\frac{d\hat{\mathbb{E}X_j}}{d\mu_i}$ does not require an increase in the sample size at each time step. Instead, the sample size is only squared when a new layer of expectation emerges so that the number of sample paths does not blow up exponentially.

4.3.5 Algorithms

Gradient Approximation Algorithm for MCGD Method

Combined with the RBM simulation algorithm, $\hat{\nabla}_{\vec{\mu}}\tilde{J}$ or $\hat{\nabla}_{\vec{\mu}}\hat{J}$ can be estimated with the algorithm below.

Algorithm 4: Approximating gradient with MCGD

```
1  Initialize an empty list grad_vec to store the gradient vector
2  for $i \in [\Delta t, 2\Delta t, ..., T]$ do
3      Generate the RBM sample paths with Algorithm 1, initialize
        $partial\_sum \leftarrow 0$
4      for $j \in [\Delta t, 2\Delta t, ..., T]$ do
5          if $1 = i = j$ then $increment \leftarrow \frac{1}{N}\sum_{n=1}^{N}\frac{d\log\ p_{X_1}(x_1)}{d\mu_1}x_n$,
            $partial\_sum \leftarrow partial\_sum + increment$;
6          else if $1 = i < j$ then
            $increment \leftarrow \frac{1}{N}\sum_{m=1}^{N}\left(\frac{d\left(\log\ p_{X_1}(x_1)\right)}{d\mu_i}\left(\frac{1}{N}\sum_{n=1}^{N}x_{j_{m,n}}\right)\right)$
            $partial\_sum \leftarrow partial\_sum + increment$;
7          else if $1 < i = j$ then
            $increment \leftarrow \frac{1}{N}\sum_{n=1}^{N}\left(\frac{d\log\ p_{X_i|X_{i-1}}(x_{i_n}|x_{i-1_n})}{d\mu_i}x_{j_n}\right)$
            $partial\_sum \leftarrow partial\_sum + increment$;
8          else if $1 < i < j$ then
            $increment \leftarrow \frac{1}{N}\sum_{m=1}^{N}\left(\frac{d\log\ p_{X_i|X_{i-1}}(x_{i_m}|x_{i-1_m})}{d\mu_i}\left(\frac{1}{N}\sum_{n=1}^{N}x_{j_{m,n}}\right)\right)$
            $partial\_sum \leftarrow partial\_sum + increment$;
9          else if $i > j$ then
10         $increment \leftarrow 0$;
11     end
12     Append partial_sum to grad_vec
13 end
14 Return grad_vec as the gradient vector myalg
```

Gradient Descent Algorithm for MCGD Method

Algorithm 5: Gradient descent optimization

```
1 Input an arbitrary drift, $\vec{\mu}_{arbitrary}$ and step size, $\alpha$
2 Initialize an arbitrary drift: $\vec{\mu_0} \leftarrow \vec{\mu}_{arbitrary}$
3 while the stopping criterion is not met do
4     For the nth iteration, estimate $\hat{\nabla}_{\vec{\mu_n}}\tilde{J}$ using Algorithm 5
5     $\vec{\mu_{n+1}} \leftarrow \vec{\mu_n} - \alpha\hat{\nabla}_{\vec{\mu_n}}\tilde{J}$
6     $n \leftarrow n+1$
7 end
8 Return the optimal drift vector $\vec{\mu_n}$
```

4.4 Step Size Determination

For both SPSA and MCGD method, α, the step size of the gradient descent method, is crucial. An overly large step size could increase the Euclidean distance between drifts from two sequential iterations and the optimal drift may not be found. A step size that is too small could potentially decelerate the convergence to a local optimal drift. Therefore, the step size should be chosen carefully.

4.4.1 Combining the Wolfe Conditions

The Wolfe conditions [14] consists of two standards for evaluating the step size:

$$f(x + \alpha d) \leq f(x) + c_1 \alpha f'(x; d)$$

$$f'(x + \alpha d; d) \geq c_2 f'(x; d),$$

where the function $f(\cdot)$ is smooth. The first condition is called the Armijo rule [15], also known as the 'sufficient decrease' condition. This condition guarantees that, with proper step size, the decrease of the objective function will be larger than a benchmark. The benchmark depends on the choice of c_1 but it is at the same scale as the slope multiplied by the step size. This condition is imposed so that the step size will not be too large causing the local minimum to be skipped between the iterations. The second condition is called the curvature condition. It ensures that the directional derivative of the objective function is larger (less steep) on the new location versus the old location. If the opposite situation happens, it follows that the objective function could have decreased even more along the descent direction and the current step size is not sufficiently large to exert this potential. The curvature condition prevents the step size from being too small. The Wolfe conditions are an efficient way to check if the current step size is feasible.

4.4.2 Determining the Step Size

Below is our algorithm used to determine the step size in the gradient descent method. Checking the Wolfe conditions requires the knowledge of directional derivatives of the objective. Say for any function $f(x)$, the directional derivative of that function on any given direction d can be written as

$$f'(x; d) = \nabla f \cdot \frac{d}{|d|}.$$

Since the directional derivative is not directly computable in our settings, we estimate is as

$$\hat{f}'(x; d) = \hat{\nabla} f \cdot \frac{d}{|d|},$$

in the algorithm.

Algorithm 6: Wolfe condition step size selection algorithm

```
Input any objective function as $f$
Initialize two arbitrary constant $c_1$ and $c_2$ so that $0 < c_1 < c_2 < 1$
Initialize $a \leftarrow 0$, $\alpha \leftarrow 1$ and $b \leftarrow \infty$
Set $stopping\_label \leftarrow 0$
while $stopping\_label = 0$ do
    Compute the old directional derivative as $\hat{f}'(x; d)$ as the dot product of
      the gradient and the descent direction
    Compute the new function value with the current step size $f(x + \alpha d)$
    Compute the new directional derivative as $\hat{f}'(x + \alpha d; d)$ as the dot
      product of the gradient and the descent direction
    if $f(x + \alpha d) > f(x) + c_1 \alpha \hat{f}'(x; d)$ then
        Set $b \leftarrow \alpha$ and $\alpha \leftarrow \frac{1}{2}(a + b)$
    end
    else if $\hat{f}'(x + \alpha d; d) < c_2 \hat{f}'(x; d)$ then
        Set $a \leftarrow \alpha$, set $\alpha \leftarrow 2a$ if $b = +\infty$ and otherwise, set $\alpha \leftarrow \frac{1}{2}(a + b)$
    end
    else
        $stopping\_label \leftarrow 1$
    end
    if either the total iteration number or the resulting step size exceeds the
      thresholds then
        Set $\alpha \leftarrow$ 0 to be conservative
        $stopping\_label \leftarrow 1$
    end
end
Return the optimal step size $\alpha$
```

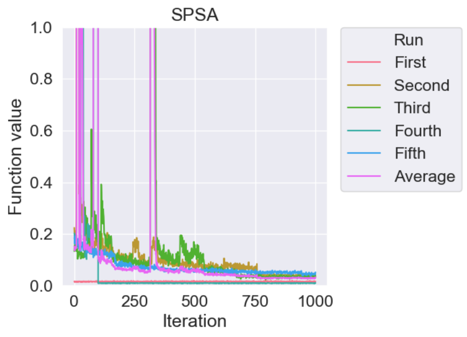
5. RESULTS

We now present numerical experiments illustrating both SPSA and MCGD methods, as well as comparisons with numerical solutions to the Hamilton Jacobi PDE. We use the following problem setting:

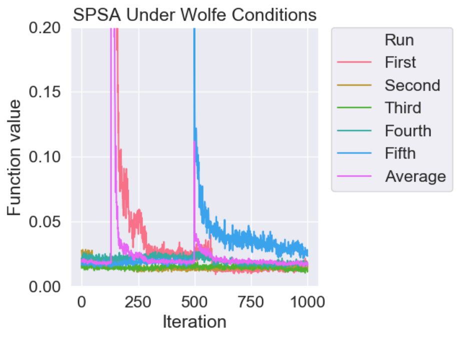
$$\begin{aligned} T &= 5, \\ h &= 1, \\ \mu_0 &= [1,1,1,1,1], \\ G(x) &= -3\mathbb{E}(X_T), \\ N &= 50, \end{aligned}$$

where T is the time horizon, h is the time discretization, μ_0 is the initial drift vector and $G(x)$ is the final condition/cost term if there is one, N is the Monte Carlo simulation sample size. We also set 10 to be the upper limit for each drift vector entry and 30 to be maximum step size for Wolfe conditions. All methods are run 30 times with different seeds.

5.1 SPSA Method Results

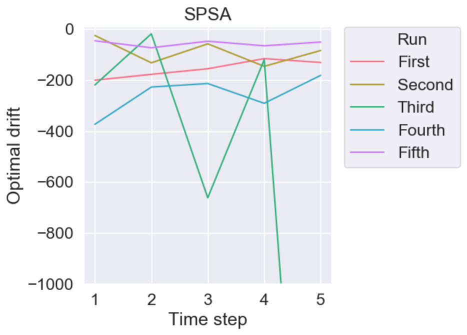


(a) With step size 1

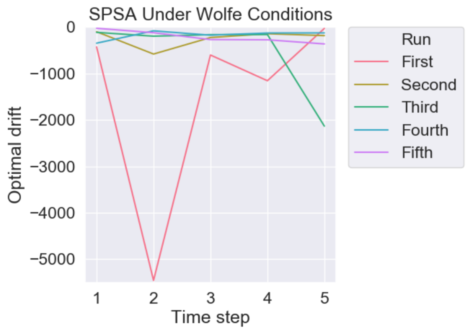


(b) With Wolfe conditions

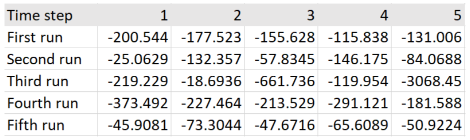
Figure 5.1.: SPSA method without final condition: objective function value vs. iteration.



(a) With step size 1

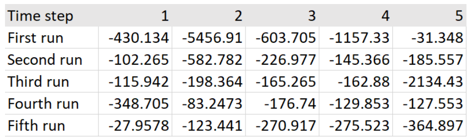


(b) With Wolfe conditions



Time step	1	2	3	4	5
First run	-200.544	-177.523	-155.628	-115.838	-131.006
Second run	-25.0629	-132.357	-57.8345	-146.175	-84.0688
Third run	-219.229	-18.6936	-661.736	-119.954	-3068.45
Fourth run	-373.492	-227.464	-213.529	-291.121	-181.588
Fifth run	-45.9081	-73.3044	-47.6716	-65.6089	-50.9224

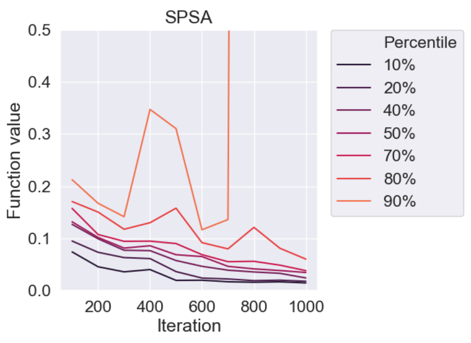
(c) With step size 1



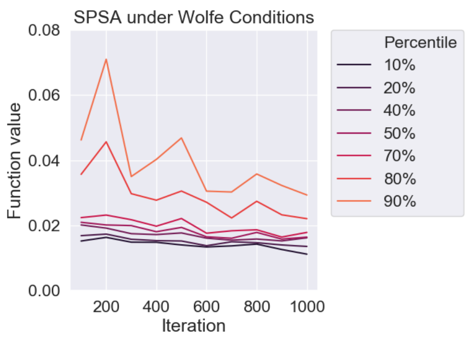
Time step	1	2	3	4	5
First run	-430.134	-5456.91	-603.705	-1157.33	-31.348
Second run	-102.265	-582.782	-226.977	-145.366	-185.557
Third run	-115.942	-198.364	-165.265	-162.88	-2134.43
Fourth run	-348.705	-83.2473	-176.74	-129.853	-127.553
Fifth run	-27.9578	-123.441	-270.917	-275.523	-364.897

(d) With Wolfe conditions

Figure 5.2.: SPSA method without final condition: optimal drift.



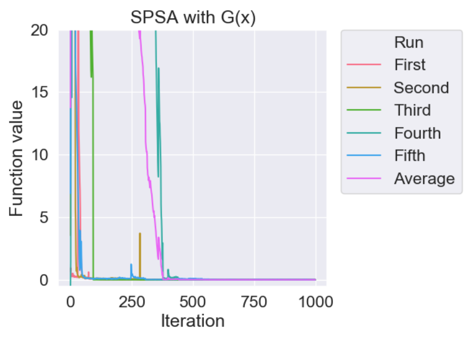
(a) With step size 1



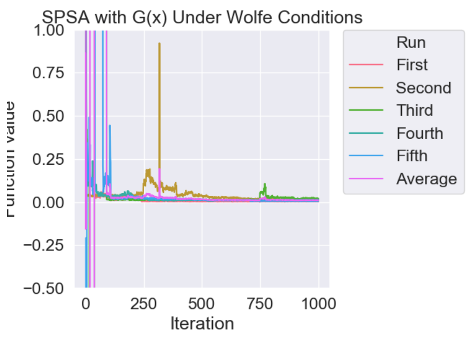
(b) With Wolfe conditions

Figure 5.3.: SPSA method without final condition: objective function value percentiles.

Figure 5.1 plots are objective function values against the number of iteration on 5 trial runs, the corresponding optimal drift are plotted in Figure 5.2 and percentiles of function values against the number of iteration calculated from 30 runs of the algorithms in Figure 5.3. Both left figures are the results using 1 as a constant step size and the right figures are the results using Wolfe conditions to determine the optimal step size. In both methods, the objective function values are not monotonically decreasing. Instead, there are fluctuations in different scales. The instability of the decreasing rate can also be observed. As it can be seen from Figure 5.1, the SPSA method under Wolfe conditions usually achieves a small objective function value after the first iteration and appears to be more effective than the fixed step size. The efficiency of Wolfe conditions is more prominent in Figure 5.3 as all the function value percentiles are lower than those of the fixed step size. The 90th function value percentile using constant step size deviates from the decreasing trend and rapidly increases. The spread between the lowest and the highest percentile is significantly smaller after the Wolfe conditions are applied, implying the variance of the estimator is decreased.

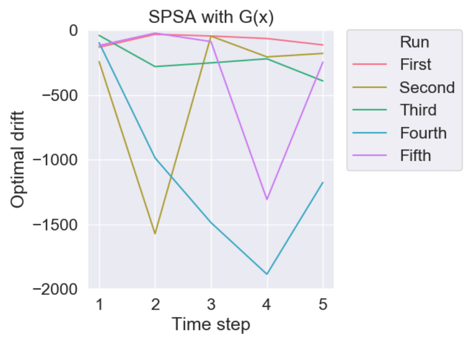


(a) With step size 1

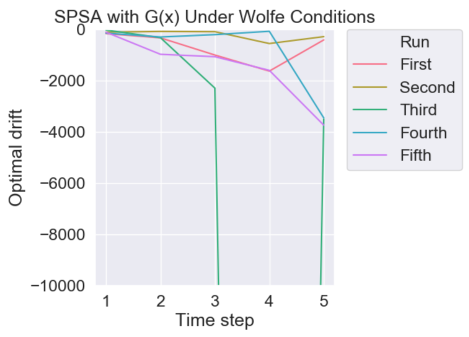


(b) With Wolfe conditions

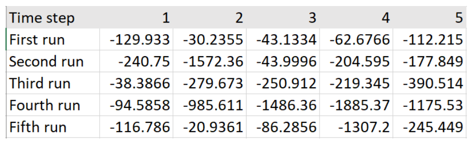
Figure 5.4.: SPSA method with final condition: objective function value vs. iteration.



(a) With step size 1

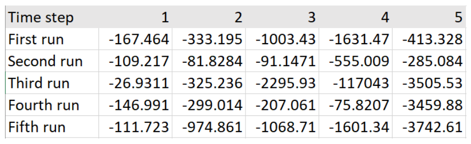


(b) With Wolfe conditions



Time step	1	2	3	4	5
First run	-129.933	-30.2355	-43.1334	-62.6766	-112.215
Second run	-240.75	-1572.36	-43.9996	-204.595	-177.849
Third run	-38.3866	-279.673	-250.912	-219.345	-390.514
Fourth run	-94.5858	-985.611	-1486.36	-1885.37	-1175.53
Fifth run	-116.786	-20.9361	-86.2856	-1307.2	-245.449

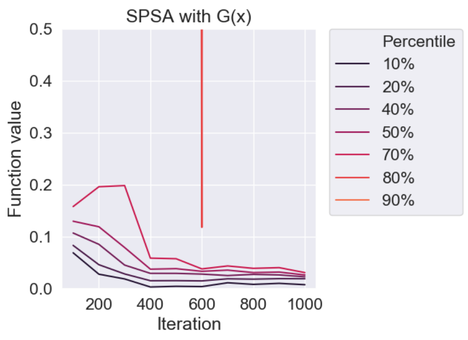
(c) With step size 1



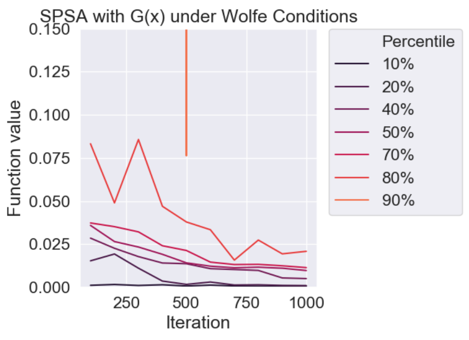
Time step	1	2	3	4	5
First run	-167.464	-333.195	-1003.43	-1631.47	-413.328
Second run	-109.217	-81.8284	-91.1471	-555.009	-285.084
Third run	-26.9311	-325.236	-2295.93	-117043	-3505.53
Fourth run	-146.991	-299.014	-207.061	-75.8207	-3459.88
Fifth run	-111.723	-974.861	-1068.71	-1601.34	-3742.61

(d) With Wolfe conditions

Figure 5.5.: SPSA method with final condition: optimal drift.



(a) With step size 1

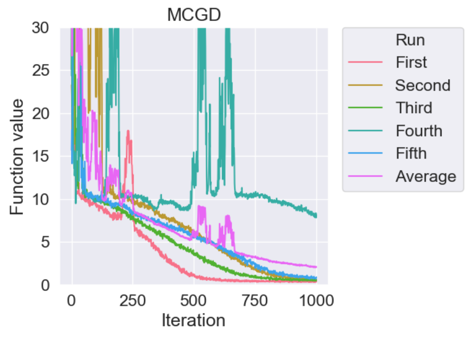


(b) With Wolfe conditions

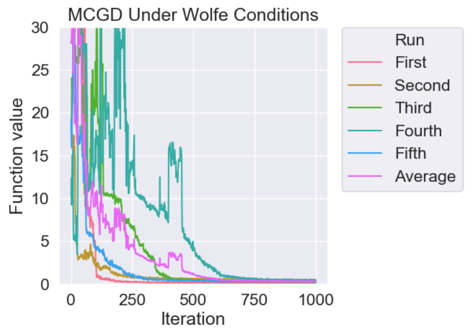
Figure 5.6.: SPSA method with final condition: objective function value percentiles.

The above figures are results generated using the SPSA method on the objective function with a terminal cost. With a terminal cost $-3G(\mathbb{E}X_T)$, the objective value could potentially be negative if $\mathbb{E}X_T$ is sufficiently large. For some trials from either step size selection methods, the function value did visit a negative level in the beginning according to Figure 5.4. As can be seen from Figure 5.2, The 90th percentile for both selection methods increases significantly and does not converge after a thousand iterations. Function values of approximately 80% of the runs converge towards 0 as the number of iterations approaches a thousand. A possible explanation for this phenomenon is that the SPSA method fails to approximate the gradient when the function value is close to zero so the potential negative level of the objective function is never reached. An alternative possibility is that 0 could be a local minimum or a saddle point of the objective function. However, analytically checking this seems almost impossible, since closed forms for the objectives are not available. In terms of the convergence rate, the Wolfe conditions slightly accelerate the convergence towards 0 and the spread between the lower and higher percentiles is smaller using Wolfe conditions.

5.2 MCGD Method Results

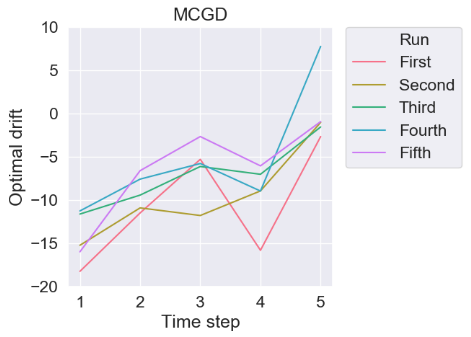


(a) With step size 1

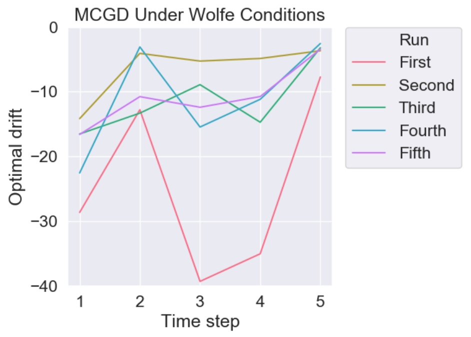


(b) With Wolfe conditions

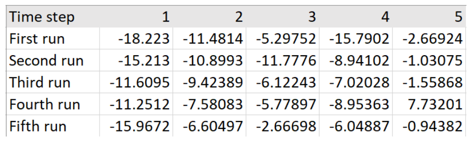
Figure 5.7.: MCGD method without final condition: value vs. iteration.



(a) With step size 1

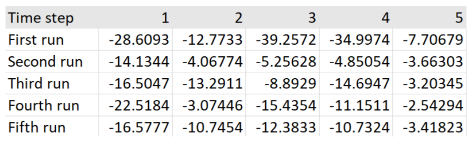


(b) With Wolfe conditions



Time step	1	2	3	4	5
First run	-18.223	-11.4814	-5.29752	-15.7902	-2.66924
Second run	-15.213	-10.8993	-11.7776	-8.94102	-1.03075
Third run	-11.6095	-9.42389	-6.12243	-7.02028	-1.55868
Fourth run	-11.2512	-7.58083	-5.77897	-8.95363	7.73201
Fifth run	-15.9672	-6.60497	-2.66698	-6.04887	-0.94382

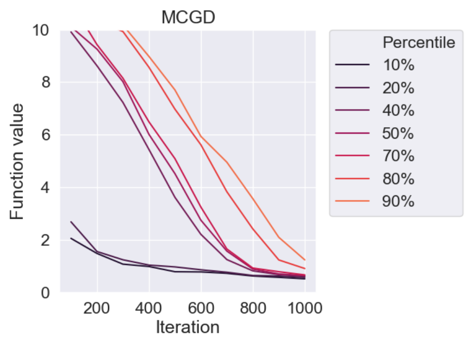
(c) With step size 1



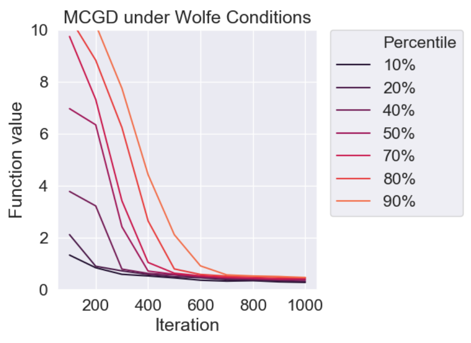
Time step	1	2	3	4	5
First run	-28.6093	-12.7733	-39.2572	-34.9974	-7.70679
Second run	-14.1344	-4.06774	-5.25628	-4.85054	-3.66303
Third run	-16.5047	-13.2911	-8.8929	-14.6947	-3.20345
Fourth run	-22.5184	-3.07446	-15.4354	-11.1511	-2.54294
Fifth run	-16.5777	-10.7454	-12.3833	-10.7324	-3.41823

(d) With Wolfe conditions

Figure 5.8.: MCGD method without final condition: optimal drift.



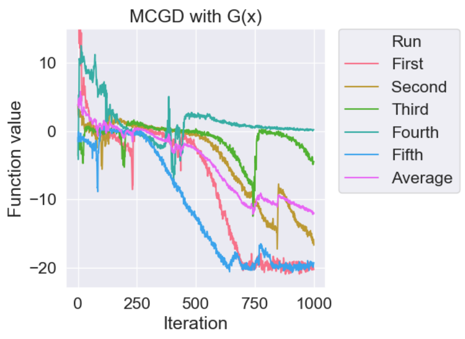
(a) With step size 1.



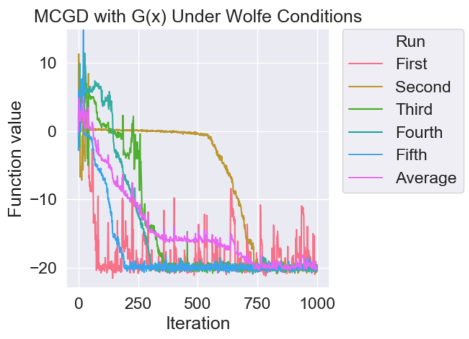
(b) With Wolfe conditions.

Figure 5.9.: MCGD method without final condition: objective function value percentiles.

For both step size selection methods, the function values are not monotonically decreasing as the iterations number increases. There are obvious transient fluctuations especially in the early phase of the iterations. Compared to SPSA, the function values using MCGD did not drastically decrease within the first few iterations, but gradually decreased trend despite the fluctuations. The trials with a fixed step size visibly converge slower than those with Wolfe condition. Some trial runs with step size = 1 experienced more fluctuations and did not converge after 1000 iterations. Overall, MCGD method combined with the Wolfe conditions converges faster. This can also be confirmed by Figure 5.9(b) as all percentiles converge to 0 at the end of the horizon. According to the percentile plots, the higher percentiles for both step size selection methods no longer diverge. The spread across different percentiles is visibly more consistent in either case. In contrast to the SPSA method, choosing the step properly does not affect the convergence rate that significantly for the MCGD method.

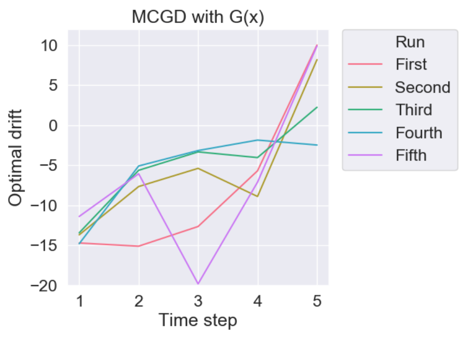


(a) With step size 1

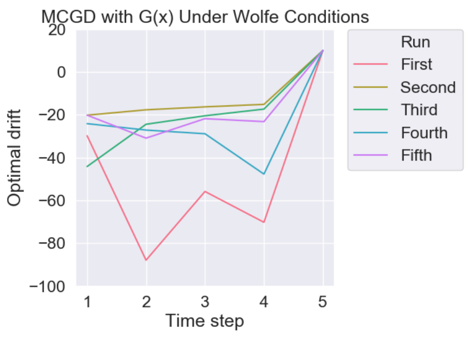


(b) With Wolfe conditions

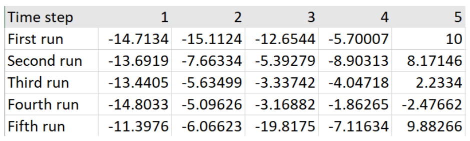
Figure 5.10.: MCGD method with final condition: value vs. iteration.



(a) With step size 1

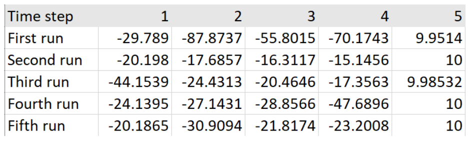


(b) With Wolfe conditions



Time step	1	2	3	4	5
First run	-14.7134	-15.1124	-12.6544	-5.70007	10
Second run	-13.6919	-7.66334	-5.39279	-8.90313	8.17146
Third run	-13.4405	-5.63499	-3.33742	-4.04718	2.2334
Fourth run	-14.8033	-5.09626	-3.16882	-1.86265	-2.47662
Fifth run	-11.3976	-6.06623	-19.8175	-7.11634	9.88266

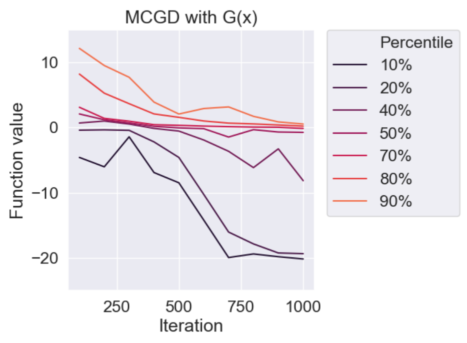
(c) With step size 1



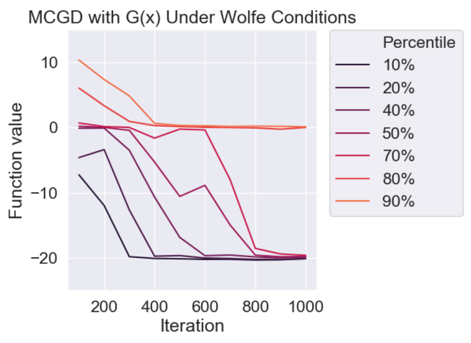
Time step	1	2	3	4	5
First run	-29.789	-87.8737	-55.8015	-70.1743	9.9514
Second run	-20.198	-17.6857	-16.3117	-15.1456	10
Third run	-44.1539	-24.4313	-20.4646	-17.3563	9.98532
Fourth run	-24.1395	-27.1431	-28.8566	-47.6896	10
Fifth run	-20.1865	-30.9094	-21.8174	-23.2008	10

(d) With Wolfe conditions

Figure 5.11.: MCGD method with final condition: optimal drift.



(a) With step size 1.



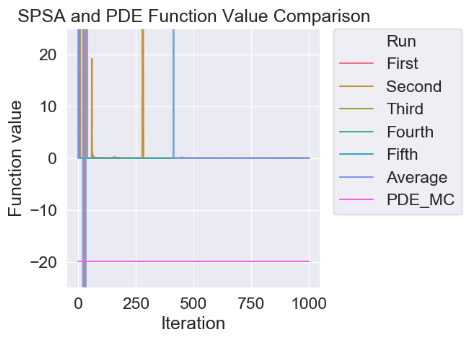
(b) With Wolfe conditions.

Figure 5.12.: MCGD method with final condition: objective function value percentiles.

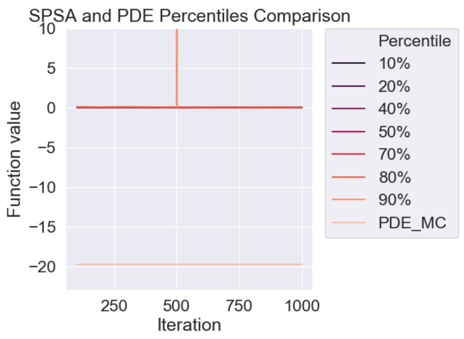
Figure 5.10-5.12 plot the outcome from running the MCGD method on an objective function with a terminal cost $-3G(\mathbb{E}X_T)$. For both step size selection methods, there are fluctuations of the objective value throughout the 1000 iterations. For the fixed step size case, the objective value did not stabilize after 1000 iterations for at least 80% of the trial runs according to Figure 5.12. However, 80% of the trial runs using Wolfe conditions converges after a thousand iterations. Unlike the SPSA method, the MCGD method can reach a negative level and sample runs appear to approach the -20 level. The MCGD method appears to be more effective especially when the function value is around 0. As for the optimal drift, a common trait is that drifts vectors for the five sample trial runs are initially small and tilted back towards the upper bound for the last time step.

5.3 Comparison of PDE and Gradient Descent Methods

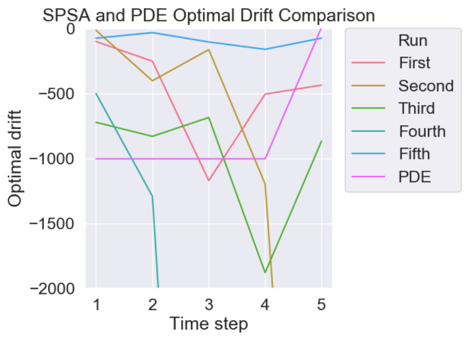
5.3.1 Objective Function with a Single Terminal Cost



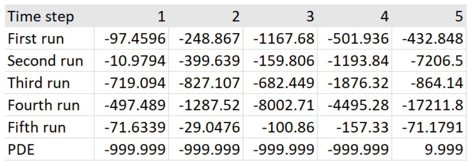
(a) SPSA function value vs. iteration and PDE optimal function value



(b) SPSA function value percentiles



(c) SPSA and PDE optimal drift graph

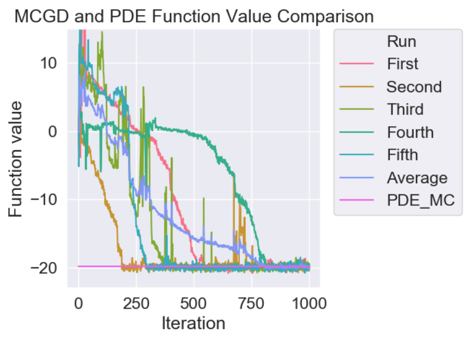


Time step	1	2	3	4	5
First run	-97.4596	-248.867	-1167.68	-501.936	-432.848
Second run	-10.9794	-399.639	-159.806	-1193.84	-7206.5
Third run	-719.094	-827.107	-682.449	-1876.32	-864.14
Fourth run	-497.489	-1287.52	-8002.71	-4495.28	-17211.8
Fifth run	-71.6339	-29.0476	-100.86	-157.33	-71.1791
PDE	-999.999	-999.999	-999.999	-999.999	9.999

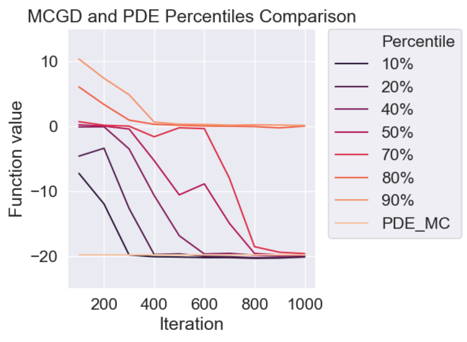
(d) SPSA and PDE optimal drift table

Figure 5.13.: Comparison of SPSA and PDE on single terminal cost set-up.

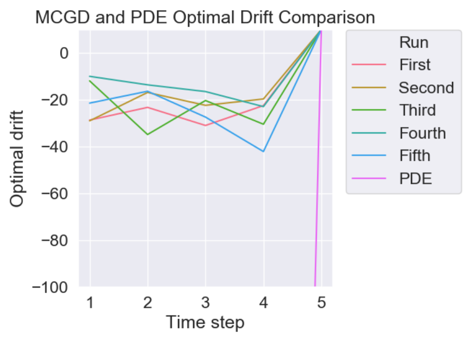
Figure 5.13 compares to SPSA and PDE methods on the objective function with a single terminal cost $-3\mathbb{E}G(X_T)$. As can be seen from Figure 5.13(a) where the evolution of five trial runs' are plotted, the function values mostly decayed or stabilized around 0 within the first 100 iterations. There are drastic fluctuations after 100 iterations but as Figure 5.13(b) shows, only the $90th$ percentile of the function values deviate from 0 and most of the trial runs end around 0. The resulting optimal drift vectors are noisy without obvious trends. Generally speaking, SPSA does not generate nearly optimal drifts regarding the objective function with a single terminal cost.



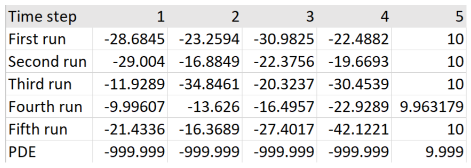
(a) MCGD function value vs. iteration and PDE optimal function value



(b) MCGD function value percentiles



(c) MCDG and PDE optimal drift graph



Time step	1	2	3	4	5
First run	-28.6845	-23.2594	-30.9825	-22.4882	10
Second run	-29.004	-16.8849	-22.3756	-19.6693	10
Third run	-11.9289	-34.8461	-20.3237	-30.4539	10
Fourth run	-9.99607	-13.626	-16.4957	-22.9289	9.963179
Fifth run	-21.4336	-16.3689	-27.4017	-42.1221	10
PDE	-999.999	-999.999	-999.999	-999.999	9.999

(d) MCGD and PDE optimal drift table

Figure 5.14.: Comparison of MCGD and PDE on single terminal cost set-up.

Figure 5.14 compares the MCGD and PDE methods on the same objective. According to Figure 5.14(b), up to 70% percentiles of the function values from all 30 trial runs converge to a level similar to the PDE method results. The remaining trial runs converge around 0 at the end of the 1000 iterations. Regarding the optimal drift vector, the PDE method gives a clean result where the first few drift entries are the lower limits for drift vectors and the last entry is the upper limit. Using the MCGD method, the trend of being negative initially and increasing to the upper bound of the drift vector is consistent with PDE method results. Then the MCGD method appears to be a better choice with terminal costs.

6. CONCLUSION

From queueing theory, finance to biology and biophysics, RBM is a common model in many scientific domains. Optimization of cost function driven by RBM's arise in many applications and there are many open problems. In this thesis, we formulated a generic optimization problem driven by RBM that can be specialized to specific applications. We focused on numerical methods for solving this generic problem.

First, we viewed the optimization problem as a deterministic optimal control problem and using the dynamic programming principle and some manipulations on the objective function, an HJB equation was derived. Utilizing the finite element method (FEM) to numerically approximate the solution over all sub-domains, we obtained approximate optimal cost as well as the optimal control, i.e. the optimal drift vector. We analyzed the convexity of the objective function and demonstrated that the generic objective is not always convex.

Based on the fact that analytical evaluation of the objective function is a near-impossible task, we developed a Monte Carlo approach to solving this generic problem. this includes the estimation of the gradient of the objective and developing a gradient descent algorithm. Approximating gradients using Monte Carlo methods is not unprecedented, and in this thesis, we used the existing and well established SPSA method, as well as a new MCGD method that we derived.

Our numerical results show that the PDE method and gradient descent method produced optimal drifts of similar trends. Carrying out a Monte Carlo estimation on the objective function using the resulting drifts from both methods generated a very close objective function value. Overall, the MCGD method out-performs the SPSA method, in terms of the convergence rate and the final function value. By using the Wolfe conditions that are commonly used to determine the gradient descent step size in deterministic optimization, we demonstrated a significant improvement in convergence rate. However, we still do not have highly consistent results using different methods. The optimal drifts computed from gradient descent methods are rather noisy despite the convergence of the algorithm.

The work in this thesis leads to several interesting and important open problems. Understanding the properties (e.g. curvature) of the objective function requires further analytical work and experimentation. Furthermore, we have not analyzed the convergence of the MCGD gradient estimator, or the bias and variance trade-off inherent in the time-discretization. Further algorithmic questions include the design of adaptive sampling schemes for the nested estimator. Going back to the optimal control perspective, the critical issue with that method is that the system dynamics (i.e., the dynamics of the expectation of RBM) are unknown in closed form. This turns out to be critical for solving the HJB equation numerically. However, this can also now be viewed as a reinforcement learning and system identification problem. What is the connection between our simulation optimization gradient descent method and policy gradient and policy search methods? It appears that these are closely connected. These questions, however, are outside the scope of this thesis but promise fruitful future research opportunities.

REFERENCES

REFERENCES

[1] Donald L Iglehart and Ward Whitt. Multiple channel queues in heavy traffic. i. *Advances in Applied Probability*, 2(1):150–177, 1970.

[2] Harsha Honnappa, Rahul Jain, and Amy R Ward. A queueing model with independent arrivals, and its fluid and diffusion limits. *Queueing Systems*, 80(1-2):71–103, 2015.

[3] Denis S Grebenkov. Partially reflected brownian motion: a stochastic approach to transport phenomena. *Focus on probability theory*, pages 135–169, 2006.

[4] Soren Asmussen, Peter Glynn, Jim Pitman, et al. Discretization error in simulation of one-dimensional reflecting brownian motion. *The Annals of Applied Probability*, 5(4):875–896, 1995.

[5] J Michael Harrison. *Brownian motion and stochastic flow systems*. Wiley New York, 1985.

[6] J Michael Harrison. *Brownian models of performance and control*. Cambridge University Press, 2013.

[7] Lawrence Evans. *Partial Differential Equations*. American Mathematical Society, 2010.

[8] Augustin Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[9] Jack Kiefer, Jacob Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[10] James C Spall et al. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 1992.

[11] James C Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE transactions on automatic control*, 45(10):1839–1853, 2000.

[12] Shalabh Bhatnagar, HL Prasad, and LA Prashanth. *Stochastic recursive algorithms for optimization: simultaneous perturbation methods*, volume 434. Springer, 2012.

[13] Tom Rainforth, Robert Cornish, Hongseok Yang, Andrew Warrington, and Frank Wood. On nesting monte carlo estimators. *arXiv preprint arXiv:1709.06181*, 2017.

[14] Philip Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

[15] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.