COMPUTATIONAL IMPROVEMENTS FOR STOCHASTIC SIMULATION WITH MULTILEVEL MONTE CARLO

by

Zane Colgin

A Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy in Computational Sciences

Middle Tennessee State University

August 2016

Dissertation Committee:

Dr. Abdul Khaliq, Chair

Dr. Yuri Melnikov

Dr. Zachariah Sinkala

Dr. William Robertson

ACKNOWLEDGMENTS

I would like to thank Dr. John Wallin and Middle Tennessee State University for the opportunity to study as a part of the Computational Science Ph.D. program and for providing an assistantship. I would like to thank Dr. Don Nelson, Dr. Rebecca Calahan, Dr. Julie Murdock, and the Mathematics Department for facilitating my teaching and tutoring assignments. I would like to thank Dr. Guannan Zhang, Dr. Clayton Webster and Oak Ridge National Laboratory for my summer internship and for our collaboration. I would like to thank my many classmates, project teams, and collaborators at Middle Tennessee State University for our fruitful discussions and teamwork. I would like to thank Dr. Yuri Melnikov for allowing me to work for him for my teaching credit.

I would like to give my dissertation committee of Dr. Zachariah Sinkala, Dr. William Robertson, Dr. Yuri Melnikov, and Dr. Abdul Khaliq, my sincere thanks. Most importantly I would like to express my deepest gratitude to Dr. Abdul Khaliq. He has been my professor, my supervisor, my advisor, and a most valued collaborator. Without his support and our many conversations none of this would have been possible.

ABSTRACT

In this work we implement various techniques to improve the multilevel Monte Carlo (MLMC) method as it is applied to a variety of stochastic models. In each case we were able to show a quantifiable computational benefit.

First we explore the use of a parallel antithetic MLMC algorithm to simulate systems of stochastic differential equations (SDEs) with correlated noise. Since Lévy area approximation is unnecessary with antithetic MLMC, it is a natural choice for the solution of systems with non-diagonal, non-commutative noise. The Milstein method used with antithetic MLMC provides a computation complexity of $\mathcal{O}(\varepsilon^{-2})$ root-mean-square error. Furthermore, MLMC uses independent sampling, which is naturally suited for parallel algorithms. We display the advantages of this approach with a case study in stochastic pricing models.

Secondly, we analyze the effects of stiffness on the convergence rate to the solution of a system of SDEs. Similarly to their deterministic counterparts, stochastic differential solvers can be unstable when used with a stiff system. When unstable step sizes are taken on the lower levels of MLMC, convergence is not guaranteed. We examine two approaches to remedy this problem: 1) the use of a semi-implicit method with a larger step-size stability region and 2) simply using a more fine discretization as the initial level for the MLMC simulator. We apply this approach to a case study in biochemical reaction networks.

Lastly, we improve a recently developed MLMC algorithm, which uses an iterative solver for the solution a partial differential equation (PDE) with random input. The innovation of the original algorithm is that each sample utilizes data gathered from all previously computed samples to create a better initial guess for the iterative solver. The drawback of this method is that the computation of a sample is no longer independent in a computational sense. We use a K-dimensional tree to sort the random input initially so that groups of locally distributed samples may be computed in batches at each parallel computing node.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Monte Carlo simulation is an approximation method with a wide range of applications[1]. It should be noted that not all of these applications are stochastic models. For instance, one of the first applications of Monte Carlo that students are taught is in the approximation of deterministic integrals. However, for the purposes of this work, we use Monte Carlo simulation (particularly multilevel Monte Carlo) as an approximating method to stochastic models. For a detailed book on Monte Carlo as it applies to this work see [38].

Monte Carlo sampling methods have proven themselves to be a useful tool in lowering the computational cost for sufficiently large stochastic systems as many techniques suffer from the "curse of dimensionality", where Monte Carlo largely does not. However, traditional Monte Carlo can still be burdensome. Multilevel Monte Carlo (MLMC) was developed to accelerate the process of Monte Carlo sampling by balancing the computational work between statistical error and discretization error [26, 40].

It is widely known that Monte Carlo methods are naturally suited for parallel computation and that this quality is usually extendable to MLMC. Therefore, these methods greatly benefit from the modern architectural trend where processor manufacturers continue to increase the number processing cores on CPUs and GPUs. This multi-processor paradigm extends to high performance computing systems, which utilize large clusters of CPUs and GPUs. Given these hardware realities, researchers have much more reason to develop parallel algorithms. Parallel processing has become so widely available that methods which take advantage of independent sampling stand to benefit more readily over serial algorithms even for some systems of low dimension.

[1] For twenty years of biennial conference proceedings on state-of-the-art Monte Carlo and quasi-Monte Carlo methods see the following publications [18, 20, 23, 46, 54, 57, 55, 56, 58, 60].

Given the importance of stochastic modeling, dimension independent algorithms, and parallel computing, we have chosen to focus on the development MLMC. In the following sections we will outline the preliminary information and definitions that will be necessary for Chapters 2, 3, and 4.

1.1 Numerical Discretization of Stochastic Differential Equations

The stochastic process $\boldsymbol{X}_t = \boldsymbol{X}(t)$, that is described by a system of stochastic differential equations (SDE), takes the general form of the initial value problem (IVP)

$$dX_{i,t} = a_i(\boldsymbol{X}_t, t)dt + \sum_{j=1}^{D} b_{ij}(\boldsymbol{X}_t, t)dW_{j,t}, \;\; \boldsymbol{X}(0) = \boldsymbol{X}_0, \tag{1.1}$$

where $X_{i,t}$ is the i^{th} component of $\boldsymbol{X}(t) \in \mathbb{R}^d$ for $t \in \mathbb{R}_{\geq 0}$, $\boldsymbol{X}_0$ is a given initial condition, $a_i(\boldsymbol{X}_t, t)$, $b_{ij}(\boldsymbol{X}_t, t) : \mathbb{R}^d \times \mathbb{R}_{\geq 0} \to \mathbb{R}$, and $W_{j,t}$ is the j^{th} component of $\boldsymbol{W}_t = \boldsymbol{W}(t) \in \mathbb{R}^D \times \mathbb{R}_{\geq 0}$, a D dimensional Brownian motion defined on a complete probability space $(\Omega, \mathscr{F}, \mathbb{P})$ with filtration $\mathscr{F}_{t \geq 0}$ satisfying the usual conditions [47]. We may refer to $\boldsymbol{a}_t = \Big(a_1(\boldsymbol{X}_t, t),\, a_2(\boldsymbol{X}_t, t)\, ,...,\, a_d(\boldsymbol{X}_t, t)\Big)$ as the drift vector and $B_t = \Big[b_{ij}(\boldsymbol{X}_t, t)\Big]$ for $i = 1, 2, ..., d$, $j = 1, 2, ..., D$ as the diffusion matrix. In practice, we wish to calculate some functional $P : \mathbb{R}^d \to \mathbb{R}$ of $\boldsymbol{X}(t)$ for some $t \in [0, T]$ and calculate characteristics of the resulting distribution, e.g. the mean $\mathbb{E}\Big[P\big(\boldsymbol{X}(t)\big)\Big]$.

Only the most simplistic of these systems have analytical solutions, while more interesting systems require numerical discretization. The following definition is helpful when describing and comparing numerical discretization methods for SDEs.

Definition 1. [47] *For sequence* $\langle t_n \mid n \in \mathbb{N} \cup 0 \rangle$ *where* $t_n \in [0, T]\ \forall\ n$, *the discrete approximation* $\boldsymbol{X}_n \approx \boldsymbol{X}(t_n)$ *converges strongly with order* $q \in (0, \infty)$ *and weakly with order* $p \in (0, \infty)$ *if there exist finite constants* K_q, K_p *and a positive constant* δ_0 *independent of*

$\Delta t_n = t_n - t_{n-1}$ *such that for each* $\Delta t_n \in (0, \delta_0)$

$$\mathbb{E}\left[|\boldsymbol{X}(t_n) - \boldsymbol{X}_n|\right] \leq K_q \Delta t_n^q, \tag{1.2}$$

$$|\mathbb{E}\left[\boldsymbol{X}(t_n)\right] - \mathbb{E}\left[\boldsymbol{X}_n\right]| \leq K_p \Delta t_n^p. \tag{1.3}$$

Let the fixed step-size discretization[2] of the IVP (1.1) on the interval $t = [0, T]$ at $N+1$ points be defined as $\boldsymbol{X}_{\Delta t} = \{\big(t_n, \boldsymbol{X}(t_n)\big) \mid t_n = n\Delta t \text{ for } n = 0, 1, ..., N \text{ and } \Delta t = T/N\}$. This discretization can be estimated by $\hat{\boldsymbol{X}}_{\Delta t} = \{(t_n, \boldsymbol{X}_n) \mid \boldsymbol{X}_n \approx \boldsymbol{X}(t_n)\ \forall\, n\}$, where stochastic finite difference methods are used for the approximation of each $(d+1)$-dimensional point $\boldsymbol{X}_n$. The Euler-Maruyama method (1.4) or the Milstein method (1.5) are the most notable of these methods and have matrix notation of the form [47]

$$\begin{bmatrix} X_{1,n+1} \\ \vdots \\ X_{d,n+1} \end{bmatrix} = \begin{bmatrix} X_{1,n} \\ \vdots \\ X_{d,n} \end{bmatrix} + \begin{bmatrix} a_{1,n} \\ \vdots \\ a_{d,n} \end{bmatrix} \Delta t + \sum_{j=1}^{D} \begin{bmatrix} b_{1j,n} \\ \vdots \\ b_{dj,n} \end{bmatrix} \Delta W_{j,n}, \tag{1.4}$$

and

$$\begin{aligned} \begin{bmatrix} X_{1,n+1} \\ \vdots \\ X_{d,n+1} \end{bmatrix} = \begin{bmatrix} X_{1,n} \\ \vdots \\ X_{d,n} \end{bmatrix} + \begin{bmatrix} a_{1,n} \\ \vdots \\ a_{d,n} \end{bmatrix} \Delta t + \sum_{j=1}^{D} \begin{bmatrix} b_{1j,n} \\ \vdots \\ b_{dj,n} \end{bmatrix} \Delta W_{j,n} \\ + \frac{1}{2} \sum_{j_1=1}^{D} \sum_{j_2=1}^{D} \sum_{k=1}^{d} \frac{\partial}{\partial X_k} \begin{bmatrix} b_{1j_2,n} \\ \vdots \\ b_{dj_2,n} \end{bmatrix} b_{kj_1,n} \left(\Delta W_{j_1,n} \Delta W_{j_2,n} - \delta_{j_1,j_2} \Delta t - A_{j_1,j_2,n}\right), \end{aligned} \tag{1.5}$$

where δ_{j_1,j_2} is the Kronecker delta, each $\Delta W_{j,n}$ is an independent and identically distributed Gaussian random variable with zero mean and Δt variance (i.e. $\Delta W_{j,n} \sim$ i.i.d. $\mathcal{N}(0, \Delta t)$),

[2]Note that (1.4) and (1.5) are defined over a single step (i.e. from $\boldsymbol{X}_n$ to $\boldsymbol{X}_{n+1}$); therefore, fixed step-sizes are not required under this definition. It is merely assumed as an example time discretization.

and

$$A_{j_1,j_2,n} = \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} \left[dW_{j_1}(u) dW_{j_2}(s) - dW_{j_2}(u) dW_{j_1}(s) \right]. \tag{1.6}$$

The double stochastic integral terms (1.6) are known as Lévy areas. These terms are a non-trivial complication, as costly approximation methods are required to compute them. Moreover, for each timestep there could be $\mathscr{O}(D^2)$ Lévy areas to approximate. Discretization error introduced by such approximation methods will be discussed in Section 1.2.1.

There are particular situations where the Milstein method is simplified and no Lévy approximations are necessary [47]. The best case for simplification is called *diagonal noise*, that is

$$b_{ij}(\boldsymbol{X}_t, t) \equiv 0 \text{ and } \frac{\partial}{\partial X_i} b_{jj}(\boldsymbol{X}_t, t) \equiv 0, \tag{1.7}$$

for each $(\boldsymbol{X}_t, t) \in \mathbb{R}_{\geq 0} \times \mathbb{R}^+$, where $i, j = 1, ..., D$, $d = D$, and $i \neq j$. This provides a major simplification for Milstein which takes the form

$$\begin{aligned} X_{i,n+1} = X_{i,n} + a_i(\boldsymbol{X}_n, t_n)\Delta t + b_{ii}(\boldsymbol{X}_n, t_n)\Delta W_{i,n}, \\ + \frac{1}{2} b_{ii}(\boldsymbol{X}_n, t_n) \frac{\partial}{\partial X_i} b_{ii}(\boldsymbol{X}_n, t_n)(\Delta W_{i,n}^2 - \Delta t). \end{aligned} \tag{1.8}$$

Another special case is when the system has *commutative noise*, that is for

$$f_{ijk}(\boldsymbol{X}_t, t) = \frac{1}{2} \sum_{\ell=1}^{d} b_{\ell k}(\boldsymbol{X}_t, t) \frac{\partial}{\partial X_\ell} b_{ij}(\boldsymbol{X}_t, t), \tag{1.9}$$

that

$$f_{ijk}(\boldsymbol{X}_t, t) = f_{jik}(\boldsymbol{X}_t, t), \tag{1.10}$$

for all $i, j = 1, .., D$, $k = 1, ..., d$, and $(\boldsymbol{X}_t, t) \in \mathbb{R}^d \times \mathbb{R}_{\geq 0}$ [47]. This gives a simplification of the form

$$\begin{aligned} X_{i,n+1} = X_{i,n} + a_i(\boldsymbol{X}_n, t_n)\Delta t + \sum_{j=1}^{D} b_{ij}(\boldsymbol{X}_n, t_n)\Delta W_{j,n}, \\ + \sum_{j,k=1}^{D} f_{ijk,n}(\Delta W_{j,n}\Delta W_{k,n} - \delta_{j,k}\Delta t). \end{aligned} \tag{1.11}$$

If Lévy area calculation if unavoidable, several approximation methods have been developed (as well as for the approximation of multiple stochastic integrals in general). A rather straightforward process involving a truncated summation, where each term in the summation requires four independently generated standard Normal random numbers, was introduced by [48]. This was enhanced by [71] by adding a tail approximation, which improved the order of convergence with little computational cost. A method was developed by [63] with the same order of convergence as [71], but it requires a large sum of independent Laplace random variables (see [51]). An exact acceptance-rejection method was developed by [24]. An interesting method was developed by [51] that involved a series of Logistic random variables. The methods developed by [25] and [19] could be of interest for future research for use with multilevel Monte Carlo; however, the antithetic multilevel Monte Carlo (AMLMC) method developed in [27] proved to be the most beneficial to our purposes, as it provides an implementation of the Milstein method that does not require Lévy area approximation and retains the same weak and strong orders of convergence. This method will be covered in detail in Chapter 2.

1.2 Multilevel Monte Carlo

1.2.1 Monte Carlo Estimator

The solution of any random process has statistical properties, e.g. mean and variance. However, stochastic finite difference methods only provide single realizations of the es-

timator for each random input. Each realization can be though of as a possible outcome for the system given the bias of the estimator. In the case of SDE (1.1), stochastic finite difference methods such as (1.4) and (1.5) have a random input of an independently generated D-dimensional Wiener process $\boldsymbol{W}(t)$. The Wiener process is the formal mathematical formulation of Brownian motion; therefore it is fitting that a single independently generated $\boldsymbol{W}(t)$ is sometimes referred to as a *path* or *sample path*. However, it is not incorrect to refer to many types of samples derived from single independent random inputs as paths.

Single realizations can have little statistical significance as they individually give little information about the shape of the distribution. However, by the law of large numbers, as the sample size becomes more numerous, the distribution of samples converges to the distribution of the solution. Since we are using samples that are approximations, it is important to note that in this case, as the sample size becomes more numerous, the distribution of samples converges to the distribution of the estimator. The difference between the solution and the estimator is called estimator bias or just bias.

The Monte Carlo method is a classical tool that uses sampling for the estimation of expectations. The functional $P\big(\boldsymbol{X}(t)\big)$ on the solution $\boldsymbol{X}(t)$ to SDE (1.1) with probability space $(\Omega, \mathscr{F}, \mathbb{P})$ has an expected value that can be approximated by the finite sum

$$\mathbb{E}\Big[P\big(\boldsymbol{X}(t)\big)\Big] \approx P^{MC} = \frac{1}{M}\sum_{m=1}^{M} P\big(\hat{\boldsymbol{X}}(t,\omega^{[m]})\big), \tag{1.12}$$

where ω represents uncertain inputs of the model and $\omega^{[m]}$ is the m^{th} realization of that uncertainty which shapes the distribution of $\hat{\boldsymbol{X}}$ to that defined by the probability measure $\mathbb{P}$ as $M \to \infty$.

The accuracy of (1.12) is controlled by the weak convergence of numerical discretization for the SDE and variance of the approximate solution. To see this we consider mean square

error of numerical solution

$$\begin{aligned} MSE &= \mathbb{E}\left[\left(P_{\Delta t}^{MC} - \mathbb{E}\left[P\big(\boldsymbol{X}(t)\big)\right]\right)^2\right] \\ &= \underbrace{\left(\mathbb{E}\left[P_{\Delta t}^{MC} - P\big(\boldsymbol{X}(t)\big)\right]\right)^2}_{\text{I := Discretization bias}} + \underbrace{\mathbb{E}\left[\left(P_{\Delta t}^{MC} - \mathbb{E}\left[P_{\Delta t}^{MC}\right]\right)^2\right]}_{\text{II := Monte Carlo variance}} \\ &= \mathscr{O}\left(\Delta t^2\right) + \mathscr{O}\left(\frac{1}{M}\right), \end{aligned} \tag{1.13}$$

where Δt is the time step of the discretization, and we have assumed a numerical method with first order weak convergence, a quality shared by both Euler-Maruyama and Milstein schemes [42].

To achieve the desired error tolerance ε, it is necessary to take $M = \mathscr{O}(\varepsilon^{-2})$ and $\Delta t = \mathscr{O}(\varepsilon)$. It follows that the computational complexity of the Monte Carlo estimator is proportional to the number of Monte Carlo samples M and the number of time steps in each sample path $\mathscr{O}\left(\Delta t^{-1}\right)$ yielding a scaling cost $\mathscr{O}(\varepsilon^{-3})$ [26].

1.2.2 Multilevel Monte Carlo Estimator

In this section we discuss the multilevel Monte Carlo (MLMC) method. This method is largely attributed to the work of Heinrich [40]; however, many major developments have been made by Giles et. al. [26, 27, 28, 29][3]. MLMC has been applied to a wide variety of problems including continuous time Markov chain models (see e.g. [3]), stochastic partial differential equations (see e.g. [7]), and jump diffusion SDEs (see e.g. [72]). When paired with a numerical method of a suitable rate of strong convergence (e.g. (1.5)) and given certain continuity conditions, MLMC can achieve a computational complexity of $\mathscr{O}(\varepsilon^{-2})$

[3]These citations are by no means a comprehensive list of the work that Giles and his collaborators have done in the development of this method; however, in the author's opinion these are the most important. The listed publications also sufficiently give credit and reference to any information used in this work.

root-mean-square error [29]. Compared to the complexity of classical Monte Carlo $\mathcal{O}(\varepsilon^{-3})$, this is a considerable improvement.

In [44], Higham describes the differences between Monte Carlo and MLMC with the "black box" analogy. With Monte Carlo we have a black box that gives us a sample solution with a particular error tolerance. For SDEs the black box could be the Milstein method or the Euler-Maruyama method. We can use the black box many times to get many sample solutions that approximate the distribution of the true solution. However, this black box has a fixed error tolerance and a fixed cost associated with each use. With MLMC the black box has a dial on the side that controls the error tolerance. For fixed-step size methods for the solution of SDEs, like the Milstein or the Euler-Maruyama method, the dial would control the size of the time step Δt and, therefore, the error tolerance. Large step sizes are obviously less expensive than small step sizes. MLMC is able to exploit the dial by combining many cheaply produced samples with a few expensive samples to obtain the same statistical result as Monte Carlo at a fraction of the cost.

The multilevel technique allows for better scaling of computational cost by exploiting strong convergence properties of numerical discretization. Consider the nested family of discretizations of the time interval with steps $\Delta t = T2^{-\ell}$, $\ell = 0, 1, \ldots$ and let P_ℓ denote the value of the functional computed at level ℓ. Taking advantage of the linearity of the expectation we obtain

$$\mathbb{E}\left[P_L\right] = \mathbb{E}\left[P_0\right] + \sum_{\ell=1}^{L} \mathbb{E}\left[P_\ell^f - P_{\ell-1}^c\right], \tag{1.14}$$

where P_ℓ^f and $P_{\ell-1}^c$ are fine-path and coarse-path approximations of the payoff function respectively. Obviously, we require that $\mathbb{E}\left[P_\ell^f\right] = \mathbb{E}\left[P_\ell^c\right]$ for the consistency of this approximation.

The MLMC estimator can be derived from (1.14) by the approximation of $L+1$ expec-

tations on the right hand side of

$$\mathbb{E}\left[P_L\right] \approx P_L^{MLMC} = \sum_{\ell=0}^{L} M_\ell^{-1} \sum_{m_\ell=1}^{M_\ell} \left(P_\ell^f - P_{\ell-1}^c\right). \tag{1.15}$$

By analogy with the Monte Carlo estimator, we can establish mean square error of MLMC [17]. That is

$$\begin{aligned} MSE &= \mathbb{E}\left[\left(P_L^{MLMC} - \mathbb{E}\left[P\big(\boldsymbol{X}(t)\big)\right]\right)^2\right] \\ &= \underbrace{\left(\mathbb{E}\left[P_L^{MLMC} - P\big(\boldsymbol{X}(t)\big)\right]\right)^2}_{\text{I := Discretization bias}} + \underbrace{\mathbb{E}\left[\left(P_L^{MLMC} - \mathbb{E}\left[P_L^{MLMC}\right]\right)^2\right]}_{\text{II := MLMC variance}} \\ &= (c_1 \Delta t_L)^{2\alpha} + \sum_{\ell=0}^{L} \frac{\sigma_\ell}{M_\ell} = \varepsilon_I^2 + \varepsilon_{II}^2 = \varepsilon^2, \end{aligned} \tag{1.16}$$

where $\sigma_\ell = \mathbb{V}(P_\ell^f - P_{\ell-1}^c)$ [17].

From error estimate (1.16), it follows that the accuracy of MLMC is controlled by two parameters:

1. the number of levels $L = \left\lceil \log_2\left(T(c_1 \varepsilon_I^{-1})^{1/\alpha}\right)\right\rceil$ and
2. the number of samples per level $M_\ell = \left\lceil \dfrac{\sigma_\ell}{a_\ell \varepsilon_{II}^2} \right\rceil$,

where coefficients a_ℓ are the weights assigning certain parts of the sampling error to each level. Computational cost of MLMC is proportional to the number of samples and the number of time steps in each sample path. Therefore

$$C^{MLMC} \propto \sum_{\ell=0}^{L} \frac{M_\ell}{\Delta t_\ell}. \tag{1.17}$$

By treating a_ℓ as continuous variables, the above cost function can be minimized by using

the method of Lagrange multipliers subject to the constraint

$$\sum_{\ell=0}^{L} a_\ell = 1. \tag{1.18}$$

In this case, the Lagrangian function takes the form

$$\Lambda(a_0,...,a_L,\lambda) = \varepsilon_{II}^{-2} \sum_{\ell=0}^{L} \frac{\sigma_\ell}{a_\ell \Delta t_\ell} + \lambda \left(\sum_{\ell=0}^{L} a_\ell - 1 \right), \tag{1.19}$$

and the minimization problem has the following solution

$$a_\ell = \frac{\sqrt{\sigma_\ell \Delta t_\ell^{-1}}}{\sum_{k=0}^{L} \sqrt{\sigma_k \Delta t_k^{-1}}}. \tag{1.20}$$

Therefore, the optimal computational cost can be defined as

$$C^{MLMC} \propto \varepsilon_{II}^{-2} \left(\sum_{\ell=0}^{L} \sqrt{\sigma_\ell \Delta t_\ell^{-1}} \right)^2, \tag{1.21}$$

which gives rise to the following theorem.

Theorem 1. [26] *If there exist independent estimators P_ℓ^{MC} based on M_ℓ Monte Carlo samples, each with expected cost C_ℓ and variance σ_ℓ, and positive constants α, β, γ, c_1, c_2, c_3 such that $\alpha \geq \frac{1}{2}\min(\beta,\gamma)$ and*

1. $\mathbb{E}\left[P_\ell - P\big(\boldsymbol{X}(t)\big)\right] \leq c_1 \Delta t_\ell^{-\alpha}$,
2. $\sigma_\ell \leq c_2 \Delta t_\ell^{\beta}$,
3. $C_\ell \leq c_3 \Delta t_\ell^{-\gamma}$,

then there exists a positive constant c_4 such that for any $\varepsilon < e^{-1}$ there are values L and M_ℓ

for which the multilevel estimator

$$P_L^{MLMC} = \sum_{\ell=0}^{L} M_\ell^{-1} \sum_{m_\ell=1}^{M_\ell} \left(P_\ell^f - P_{\ell-1}^c \right). \tag{1.22}$$

has a mean-square-error with bound

$$MSE = \mathbb{E}\left[\left(P_L^{MLMC} - \mathbb{E}\left[P\big(\boldsymbol{X}(t)\big) \right] \right)^2 \right] < \varepsilon^2, \tag{1.23}$$

with a computational complexity C with bound

$$C^{MLMC} \leq c_4 \begin{cases} \varepsilon^{-2} & \text{if } \gamma - \beta < 0, \\ \varepsilon^{-2} \left| \ln \varepsilon \right|^2 & \text{if } \gamma - \beta = 0, \\ \varepsilon^{-2 - \frac{\gamma - \beta}{\alpha}} & \text{if } \gamma - \beta > 0. \end{cases} \tag{1.24}$$

Proof. See [26] □

Variances σ_ℓ are controlled by the strong convergence of the discretization scheme. If quantities P_ℓ^f and $P_{\ell-1}^c$ are Lipschitz continuous and are computed using the same Brownian path, we obtain

$$\begin{aligned} \sigma_\ell &= \mathbb{V}(P_\ell^f - P_{\ell-1}^c) = \mathbb{E}\left[(P_\ell^f - P_{\ell-1}^c)^2 \right] - \left(\mathbb{E}\left[P_\ell^f - P_{\ell-1}^c \right] \right)^2 \\ &= \mathscr{O}(\Delta t_\ell^{2q}) + \mathscr{O}(\Delta t_\ell^{2p}), \end{aligned} \tag{1.25}$$

where q and p are the strong and weak orders of the numerical discretization scheme according to Definition 1.

It is the property of the Wiener process that makes $q < p$ for difference schemes applied to SDEs with sufficiently smooth coefficients. Thus, $\sigma_\ell = \mathscr{O}(\Delta t_\ell^{2q})$ which, in light of Theorem 1, implies that strong order $q > \frac{\gamma}{2}$ is required to achieve optimal performance of

the MLMC method. In practice, $\gamma = 1$ for explicit schemes yielding in $q > \frac{1}{2}$.

Among difference methods based on the stochastic Taylor series expansion, the Milstein method is the simplest scheme possessing order of strong convergence $q = 1$. Improved $\mathcal{O}(\varepsilon^{-2})$ complexity of the MLMC based on the Milstein scheme is the direct consequence of its strong order of convergence. This result is obvious for a Lipschitz continuous payoff function which is considered in Chapter 2.

CHAPTER 2

ANTITHETIC MLMC: A CASE STUDY IN STOCHASTIC PRICING MODELS

The field of financial engineering is devoted to the development and application of financial theory toward the allocation of assets and liabilities over time. It is useful to separate the science of these allocations into two main categories: 1) the construction of descriptive, predictive models and 2) the development of accurate, efficient methods that solve or simulate these models. In this chapter we will examine the latter as it applies to a two asset exchange option valuation and the stochastic interest rate volatility model option valuation.

In their landmark paper, Black and Scholes [9] produced a theoretical valuation formula for options. Classical Black–Scholes theory states that the price of an asset may be modeled by geometric Brownian motion[1]. A system of stochastic differential equations may be constructed if correlation exist between any pair of assets. Our first example in this chapter will be of this type in the form of a system of two correlated assets with an exchange option payoff function (to be defined in Section 2.2).

The second model for consideration removes the assumption of a geometric Brownian motion driven diffusion term for the asset price SDE. Although this assumption is still of use today, it neglects variable volatility and interest rate. In [41], the Black–Scholes model was generalized to incorporate volatility as an additional stochastic process. This addition not only provided a mechanism by which variable volatility could be included in the model but also allowed for correlation between option price and volatility. A third equation can also be added to the model to incorporate interest rate as a stochastic process. A description

[1]The corresponding partial differential equation that describes the payoff of the option is known as the Black–Scholes equation.

of this system can be found in [52] and [39]. Clearly, the three-factor model allows for correlation between each pair of the three equations, however empirical data suggests that option pricing has no correlation between prices and interest rates [6]. We consider a model of this form and refer to it as the *stochastic interest rate volatility model*. We will provide a formal description of this model in Section 2.3.

We wish to approximated these systems of correlated random variables with stochastic finite difference methods such as (1.4) or (1.5). However, the definition of these methods call for uncorrelated Brownian increments (recall $\Delta W_{j,n} \sim$ i.i.d. $\mathcal{N}(0,\Delta t)$ for $j = 1,2,...,D$). To differentiate between correlated and uncorrelated Brownian motions, let us define $\boldsymbol{W}_t = \boldsymbol{W}(t)$ as a correlated D-dimensional Brownian motion with correlation matrix Π and $\boldsymbol{Z}_t = \boldsymbol{Z}(t)$ as an uncorrelated D-dimensional Brownian motion (i.e. with the identity matrix as its correlation matrix). By using a Cholesky decomposition of the correlation matrix for $\boldsymbol{W}_t$ (i.e. $\Pi = UU^T$), we can then use the identity (2.1) to relate correlated random variables as a linear combination of uncorrelated random variables [39]. That is, by using

$$d\boldsymbol{W}_t = U d\boldsymbol{Z}_t, \tag{2.1}$$

we can adjust an SDE to have uncorrelated noise with

$$d\boldsymbol{X}_t = \boldsymbol{a}_t dt + B_t d\boldsymbol{W}_t = \boldsymbol{a}_t dt + B_t(U d\boldsymbol{Z}_t), \tag{2.2}$$

while retaining the proper correlation in random variable $\boldsymbol{X}_t$.

2.1 Antithetic Multilevel Monte Carlo

As stated in Chapter 1, MLMC using a Milstein scheme may achieve a better order of complexity than Euler-Maruyama. However, the use of the Milstein method comes with complications. If working with an SDE system with non-diagonal, non-commutative

noise it is necessary to approximate the double stochastic integral terms (1.6). No efficient method is known for this except for the case $D \leq 2$ [62]. Moreover, it has been shown that it is not possible to achieve a better strong order of convergence than the Euler-Maruyama method $(q = 1/2)$ when using only Wiener increments of the Brownian path [14, 53]. Thus, the asymptotic complexity of the Milstein method will be the same as that of the Euler-Maruyama if we neglect simulation of the iterated stochastic integrals.

On the other hand, while the strong order $q = 1$ of the discretization scheme is sufficient to guarantee optimal computational cost of the MLMC method it is not necessary. What really matters is the order of the variance decay β which must be greater than 1, i.e. condition (iii) of the Theorem 1 must be satisfied. Recently [27] showed that it is possible to construct estimator for which $\beta = 2$ with the Lévy areas of the Milstein methoed set to zero. One can achieve this by exploiting flexibility in choosing the fine-path P^f and coarse-path P^c approximations of the payoff function.

In the classical MLMC method $P^f = P^c$. However, if we can construct appropriate approximations P^f and P^c such that iterated integrals cancel for their difference $P^f - P^c$ then one can disregard simulation of the Lévy areas. This is the main idea of the antithetic MLMC estimator which is based on the well-known method of antithetic variates [38]. An example of this approach is provided in [27]. By taking $P^c_{\ell-1}$ to be the usual payoff based on the $\ell - 1$ coarse simulation X^c and defined P^f_ℓ as the average of the antithetic payoffs $P(X^f)$ and $P(X^a)$. In this case, the Milstein approximation takes the form

$$\begin{aligned} X^c_{i,n+1} = X^c_{i,n} + a_i(\boldsymbol{X}^c_n, t_n)\Delta t + \sum_{j=1}^{D} b_{ij}(\boldsymbol{X}^c_n, t_n)\Delta W_{j,n} \\ + \sum_{j_1,j_2=1}^{D} f_{ij_1j_2,n}(\Delta W_{j_1,n}\Delta W_{j_2,n} - \delta_{j_1j_2}\Delta t), \end{aligned} \tag{2.3}$$

for the coarse approximation and $\boldsymbol{X}^f_{n+1} = \frac{1}{2}\left(\tilde{\boldsymbol{X}}^f_{n+1} + \tilde{\boldsymbol{X}}^a_{n+1}\right)$ for the fine approximation

$$
\begin{aligned}
\tilde{X}^f_{i,n+1/2} &= X^f_{i,n} + a_i(\boldsymbol{X}^f_n, t_n)\Delta t/2 + \sum_{j=1}^{D} b_{ij}(\boldsymbol{X}^f_n, t_n)\delta W_{j,n} \\
&\quad + \sum_{j_1,j_2=1}^{D} f_{ij_1j_2,n}(\delta W_{j_1,n}\delta W_{j_2,n} - \delta_{j_1j_2}\Delta t/2), \\
\tilde{X}^f_{i,n+1} &= \tilde{X}^f_{i,n+1/2} + a_i(\tilde{\boldsymbol{X}}^f_{n+1/2}, t_{n+1/2})\Delta t/2 + \sum_{j=1}^{D} b_{ij}(\tilde{\boldsymbol{X}}^f_{n+1/2}, t_{n+1/2})\delta W_{j,n+1/2} \\
&\quad + \sum_{j_1,j_2=1}^{D} f_{ij_1j_2,n+1/2}(\delta W_{j_1,n+1/2}\delta W_{j_2,n+1/2} - \delta_{j_1k_2}\Delta t/2),
\end{aligned}
\tag{2.4}
$$

where

$$
f_{ij_1j_2} = \frac{1}{2}\sum_{\ell=1}^{d} b_{\ell j_2}\frac{\partial b_{j_1j_2}}{\partial x_\ell},
$$

$$
\delta \boldsymbol{W}_n \equiv \boldsymbol{W}(t_{n+1/2}) - \boldsymbol{W}(t_n),
$$

$$
\delta \boldsymbol{W}_{n+1/2} \equiv \boldsymbol{W}(t_{n+1}) - \boldsymbol{W}(t_{n+1/2}).
$$

The approximation $\tilde{\boldsymbol{X}}^a_{n+1}$ is the same as $\tilde{\boldsymbol{X}}^f_{n+1}$ except that Brownian increments $\delta \boldsymbol{W}_n$ and $\delta \boldsymbol{W}_{n+1/2}$ are interchanged. That is

$$
\begin{aligned}
\tilde{X}^a_{i,n+1/2} &= X^f_{i,n} + a_i(\boldsymbol{X}^f_n, t_n)\Delta t/2 + \sum_{j=1}^{D} b_{ij}(\boldsymbol{X}^f_n, t_n)\delta W_{j,n+1/2} \\
&\quad + \sum_{j_1,j_2=1}^{D} f_{ij_1j_2,n}(\delta W_{j_1,n+1/2}\delta W_{j_2,n+1/2} - \delta_{j_1j_2}\Delta t/2), \\
\tilde{X}^a_{i,n+1} &= \tilde{X}^a_{i,n+1/2} + a_i(\tilde{\boldsymbol{X}}^a_{n+1/2}, t_{n+1/2})\Delta t/2 + \sum_{j=1}^{D} b_{ij}(\tilde{\boldsymbol{X}}^a_{n+1/2}, t_{n+1/2})\delta W_{j,n} \\
&\quad + \sum_{j_1,j_2=1}^{D} f_{ij_1j_2,n+1/2}(\delta W_{j_1,n}\delta W_{j_2,n} - \delta_{j_1k_2}\Delta t/2).
\end{aligned}
\tag{2.5}
$$

In this case, we obtain $\left|\mathbf{X}^f - \mathbf{X}^c\right| = \mathscr{O}(\Delta t)$ and $\mathbb{V}\left[\frac{1}{2}\left(\tilde{P}^f_\ell + \tilde{P}^a_{\ell-1}\right) - P^c_{\ell-1}\right] = \mathscr{O}(\Delta t^2)$ as required.

2.2 Exchange Asset Model

We have chosen a system from [1] as our first example problem. The exchange option pricing model is a system of two correlated asset prices evaluated on the interval $[0,T]$, where the exchange option payoff function is exercised at time T. That is

$$\begin{aligned} dS_{1,t} &= r_1 S_{1,t} dt + \sigma_1 S_{1,t} dW_{1,t}, \\ dS_{2,t} &= r_2 S_{2,t} dt + \sigma_2 S_{2,t} dW_{2,t}, \\ \left\langle dW_{1,t}\, dW_{2,t} \right\rangle &= \pi_{12} dt, \end{aligned} \tag{2.6}$$

and

$$P\big(\boldsymbol{S}(T)\big) = \max\{S_1 - S_2, 0\}, \tag{2.7}$$

where coefficients r_1, r_2 are constant interest rates, σ_1, σ_2 are constant volatilities, and π_{12} is the correlation coefficient for $dW_{1,t}$ and $dW_{2,t}$. Hereafter let $\pi = \pi_{12}$ for simplification of notation.

First we wish to use the identity (2.2) to relate correlated random variables $d\boldsymbol{W}_t = (dW_{1,t}\ dW_{2,t})^T$ with uncorrelated random variables $d\boldsymbol{Z}_t = (dZ_{1,t}\ dZ_{2,t})^T$. So for correlation matrix

$$\Pi = \begin{pmatrix} 1 & \pi \\ \pi & 1 \end{pmatrix}, \tag{2.8}$$

which has a Cholesky factorization of

$$U = \begin{pmatrix} 1 & 0 \\ \pi & \sqrt{1-\pi^2} \end{pmatrix}, \tag{2.9}$$

the identity (2.1) produces

$$d\boldsymbol{W}_t = U d\boldsymbol{Z}_t = \begin{pmatrix} 1 & 0 \\ \pi & c_\pi \end{pmatrix} d\boldsymbol{Z}_t = \begin{pmatrix} dZ_{1,t} \\ \pi dZ_{1,t} + c_\pi dZ_{2,t} \end{pmatrix}, \tag{2.10}$$

where $c_\pi = \sqrt{1-\pi^2}$ and $0 \leq \pi \leq 1$. This produces the new adjusted model with uncorrelated noise $d\boldsymbol{Z}_t$

$$\begin{aligned} dS_{1,t} &= r_1 S_{1,t} dt + \sigma_1 S_{1,t} dZ_{1,t}, \\ dS_{2,t} &= r_2 S_{2,t} dt + \pi\sigma_2 S_{2,t} dZ_{1,t} + c_\pi \sigma_2 S_{2,t} dZ_{2,t}. \end{aligned} \tag{2.11}$$

The Euler-Maruyama (1.4) and Milstein (1.5) formulations of (2.11) require the following terms

$$\begin{aligned} a_1(\boldsymbol{X}_t, t) &= r_1 S_{1,t}, \\ a_2(\boldsymbol{X}_t, t) &= r_2 S_{2,t}, \\ b_{1,1}(\boldsymbol{X}_t, t) &= \sigma_1 S_{1,t}, \\ b_{2,1}(\boldsymbol{X}_t, t) &= \pi\sigma_2 S_{2,t}, \\ b_{2,2}(\boldsymbol{X}_t, t) &= c_\pi \sigma_2 S_{2,t}, \\ \frac{\partial}{\partial X_1} b_{1,1}(\boldsymbol{X}_t, t) &= \frac{\partial}{\partial S_1} \sigma_1 S_{1,t} = \sigma_1, \\ \frac{\partial}{\partial X_2} b_{2,1}(\boldsymbol{X}_t, t) &= \frac{\partial}{\partial S_2} \pi\sigma_2 S_{2,t} = \pi\sigma_2, \\ \frac{\partial}{\partial X_2} b_{2,2}(\boldsymbol{X}_t, t) &= \frac{\partial}{\partial S_2} c_\pi \sigma_2 S_{2,t} = c_\pi \sigma_2, \end{aligned} \tag{2.12}$$

where the remaining terms are set to zero. This produces the Euler-Maruyama formulation

of (2.11)

$$\begin{aligned} S_{1,n+1} &= S_{1,n} + r_1 S_{1,n}\Delta t + \sigma_1 S_{1,n}\Delta W_{1,n}, \\ S_{2,n+1} &= S_{2,n} + r_2 S_{2,n}\Delta t + \pi\sigma_2 S_{2,n}\Delta W_{1,n} + c_\pi \sigma_2 S_{2,n}\Delta W_{2,n}, \end{aligned} \tag{2.13}$$

and the Milstein[2] formulation

$$\begin{aligned} S_{1,n+1} &= S_{1,n} + r_1 S_{1,n}\Delta t + \sigma_1 S_{1,n}\Delta W_{1,n} + \frac{1}{2}\sigma_1^2 S_{1,n}(\Delta W_{1,n}^2 - \Delta t), \\ S_{2,n+1} &= S_{2,n} + r_2 S_{2,n}\Delta t + \pi\sigma_2 S_{2,n}\Delta W_{1,n} + c_\pi \sigma_2 S_{2,n}\Delta W_{2,n} \\ &\quad + \frac{1}{2}\sigma_2^2 S_{2,n}\left[\pi^2(\Delta W_{1,n}^2 - \Delta t) + 2\pi c_\pi(\Delta W_{1,n}\Delta W_{2,n}) + c_\pi^2(\Delta W_{2,n}^2 - \Delta t)\right]. \end{aligned} \tag{2.14}$$

We use the sets of parameters and initial conditions from [1] to simulate model (2.11). These values are listed in Table 1. First we will use the Euler-Maruyama formulation (2.13) as our numerical discretization with MLMC as our sampling method. We will then compare these results with various metrics with the Milstein formulation with AMLMC.

Table 2 shows the average simulation cost of the 39 tests and savings compared to traditional Monte Carlo using the Euler-Maruyama method. Here cost is measured in the total number of steps taken. Note that the savings dramatically increase with ε. Table 3 shows the cost savings results using the Milstein AMLMC method. Note that the average savings are consistently better; however, there was higher variance in these saving for the 39 sample tests. Figure 1 plots the average cost of the 39 tests against ε. Here the improved rates of convergence are clearly shown with MLMC methods having a major improvement over Monte Carlo. Figure 2 plots the sample paths per level used for each test for various values of ε. Comparing the Euler-Maruyama MLMC method on the left to the Milstein AMLMC method on the right, we see a steep decline in the number of path necessary with Milstein AMLMC. Note that many tests require fewer levels to converge as well. The dotted

[2]Note that this formulation requires AMLMC to cancel the double stochastic integrals. If not using AMLMC this is not the Milstein formulation for this model.

line represents the minimum number of samples necessary for the simulation. Figure 3 shows the variance decay and weak convergence of the Euler Maruyama MLMC method for each of the tests. When compared with Figure 4, which uses Milstein AMLMC, there is clearly a much lower level of variance in the correction levels when using Milstein AMLMC. These convergence tests were made using 100 sample paths per level. Table 4 shows the estimated values of variance decay and weak convergence found in the convergence tests. Consistently and on average, Euler-Maruyama MLMC produced ~ 1 as the rate of variance decay and Milstein AMLMC produced ~ 2 as expected[3].

Table 1: Exchange Option Simulation Tests

The 39 parameter sets used to test the exchange option pricing model with $S_1(0) = 100$.

#	$S_2(0)$	r_1, r_2	T	σ_1	σ_2	π	#	$S_2(0)$	r_1, r_2	T	σ_1	σ_2	π
1	80	0	1	0.1	0.1	0.5	**21**	90	0.04	6	0.2	0.1	0.8
2	80	0	1	0.2	0.2	0.5	**22**	100	0	1	0.2	0.1	0.5
3	80	0	1	0.1	0.1	0.8	**23**	100	0	1	0.1	0.1	0.5
4	80	0	1	0.2	0.1	0.8	**24**	100	0	6	0.2	0.1	0.5
5	80	0	6	0.1	0.1	0.5	**25**	100	0	6	0.1	0.1	0.5
6	80	0	6	0.2	0.2	0.5	**26**	100	0.04	1	0.2	0.2	0.5
7	80	0	6	0.1	0.1	0.8	**27**	100	0.04	6	0.2	0.1	0.5
8	80	0.04	1	0.1	0.1	0.5	**28**	100	0.04	6	0.1	0.1	0.5
9	80	0.04	1	0.2	0.1	0.5	**29**	110	0	1	0.1	0.1	0.5
10	80	0.04	6	0.1	0.1	0.8	**30**	110	0	1	0.2	0.1	0.5
11	80	0.04	6	0.2	0.1	0.8	**31**	110	0	1	0.2	0.2	0.5
12	90	0	1	0.1	0.1	0.5	**32**	110	0	1	0.2	0.2	0.8
13	90	0	1	0.2	0.2	0.5	**33**	110	0	1	0.2	0.1	0.8
14	90	0	1	0.2	0.1	0.8	**34**	110	0.04	1	0.1	0.1	0.5
15	90	0	6	0.2	0.1	0.8	**35**	110	0.04	1	0.2	0.1	0.5
16	90	0	6	0.1	0.1	0.8	**36**	110	0.04	1	0.2	0.2	0.5
17	90	0.04	1	0.1	0.1	0.5	**37**	110	0.04	1	0.2	0.2	0.8
18	90	0.04	1	0.2	0.1	0.5	**38**	110	0.04	6	0.2	0.1	0.5
19	90	0.04	6	0.2	0.1	0.5	**39**	110	0.04	6	0.1	0.1	0.5
20	90	0.04	6	0.2	0.2	0.5							

[3]The programs for this section were written in the C++ programming language. The metrics used are platform independent and should only vary slightly with random number generation.

Table 2: Exchange Option Simulation Euler-Maruyama MLMC

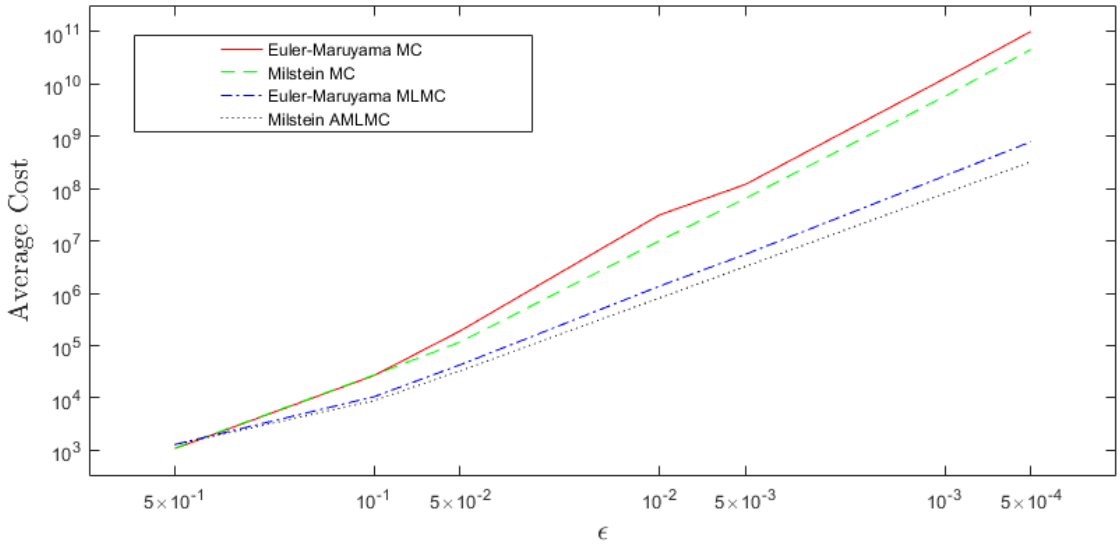
The average simulation cost of the 39 tests and savings compared to traditional Monte Carlo.

ε	average MLMC cost	average MC cost	average savings	variance savings
0.5	1.286×10^{3}	1.072×10^{3}	0.6471	0.4229
0.1	1.054×10^{4}	2.679×10^{4}	2.004	1.503
0.05	4.277×10^{4}	1.902×10^{5}	2.780	4.874
0.01	1.369×10^{6}	3.152×10^{7}	7.527	180.5
0.005	5.550×10^{6}	1.208×10^{8}	8.862	153.9
0.001	1.755×10^{8}	1.286×10^{10}	23.94	1595
0.0005	7.890×10^{8}	1.014×10^{11}	47.03	7324

Table 3: Exchange Option Simulation Milstein AMLMC

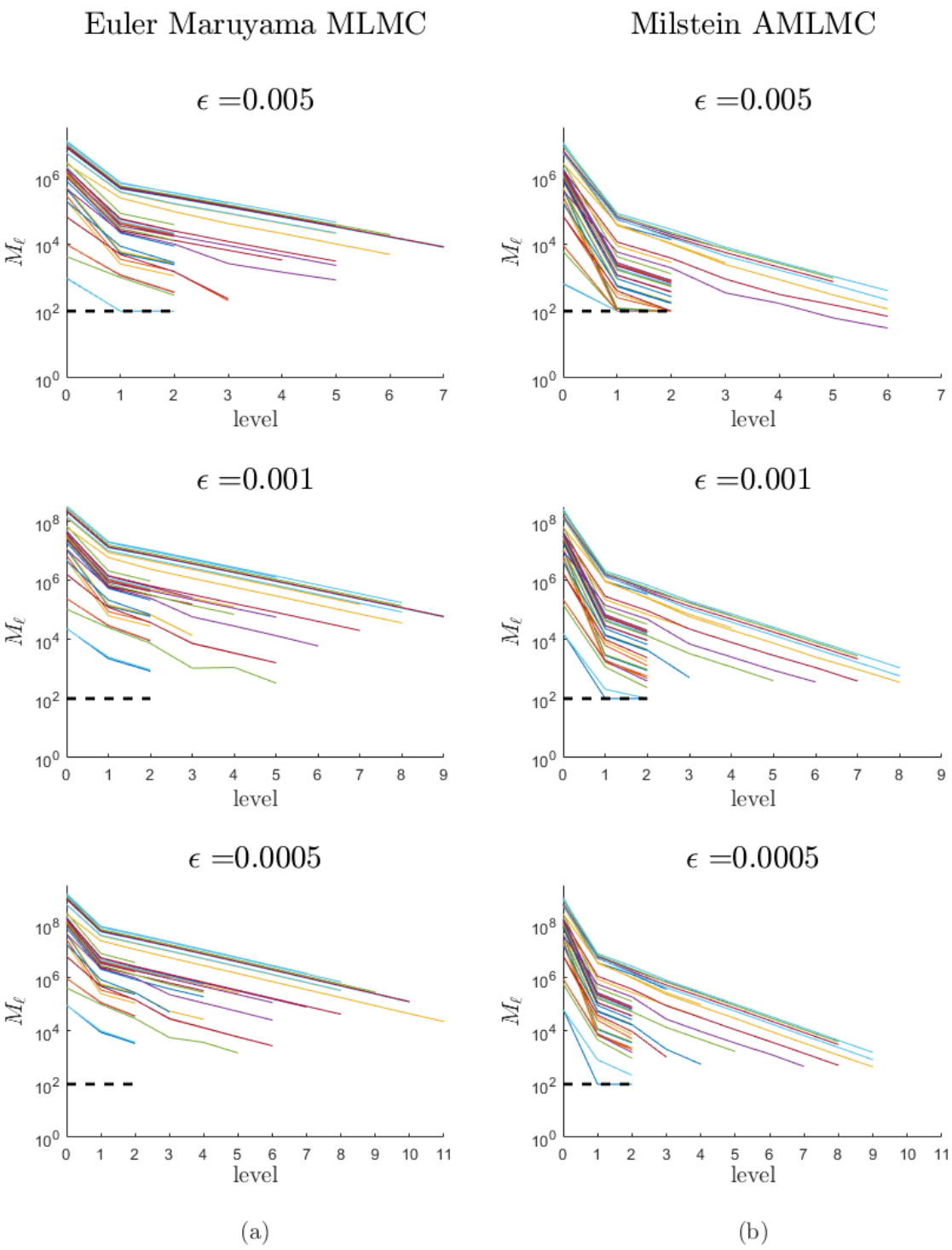
The average simulation cost of the 39 tests and savings compared to traditional Monte Carlo.

ε	average MLMC cost	average MC cost	average savings	variance savings
0.5	1.267×10^{3}	1.101×10^{3}	0.6794	0.4876
0.1	8.828×10^{3}	2.752×10^{4}	2.255	1.897
0.05	3.286×10^{4}	1.169×10^{5}	2.682	2.484
0.01	8.099×10^{5}	1.006×10^{7}	6.249	89.11
0.005	3.246×10^{6}	6.546×10^{7}	9.405	216.2
0.001	8.130×10^{7}	5.817×10^{9}	27.04	3132
0.0005	3.258×10^{8}	4.594×10^{10}	51.39	1.270×10^{4}



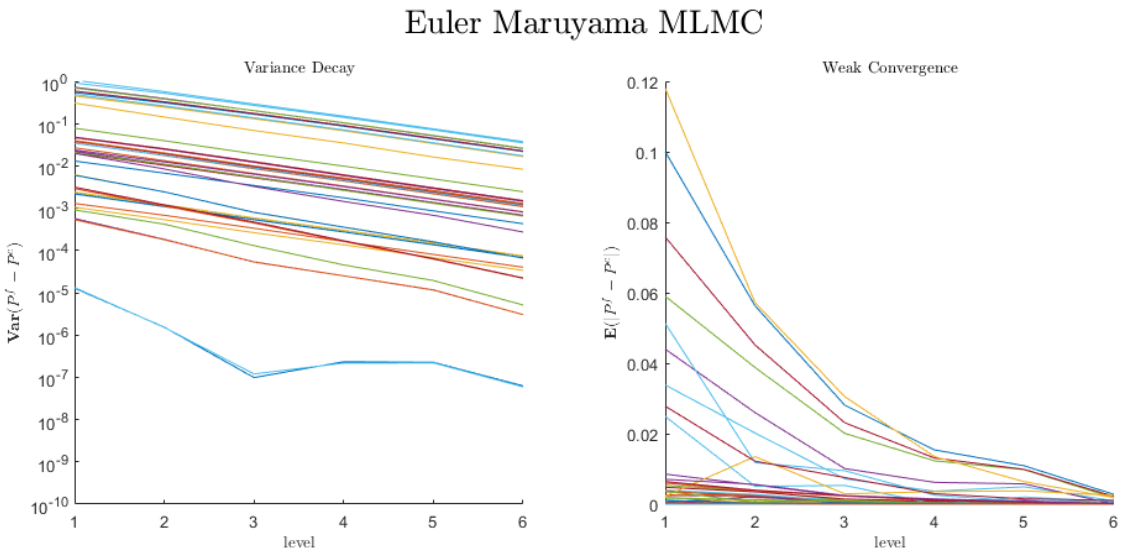
The average simulation cost for the 39 tests using the four simulation methods for decreasing values of error tolerance ε.

Figure 1: Exchange Option Simulation Cost



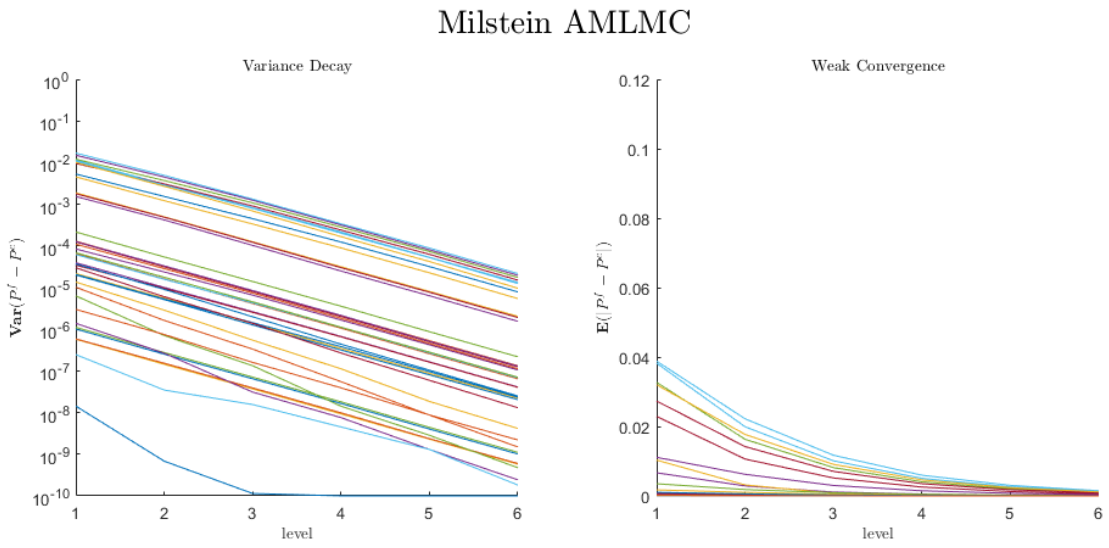
The number of sample paths needed to converge to error tolerance ε for each of the 39 tests. The minimum number of paths requred by the simulation is marked by a dotted line. *(a)* Euler Maruyama MLMC Simulation. *(b)* Milstein AMLMC Simulation.

Figure 2: Exchange Option Simulation Paths Per Level



Variance decay and weak convergence for the 39 tests simulated by 100 paths/level.

Figure 3: Exchange Option Simulation Euler-Maruyama MLMC



Variance decay and weak convergence for the 39 tests simulated by 100 paths/level.

Figure 4: Exchange Option Simulation Milstein AMLMC

Table 4: Exchange Option Simulation Rates of Convergence

The rate of weak convergence α and variance decay β for each of the 39 tests using Euler-Maruyama MLMC and Milstein AMLMC methods.

#	α^{EM}	β^{EM}	α^{Mil}	β^{Mil}	#	α^{EM}	β^{EM}	α^{Mil}	β^{Mil}
1	0.72003	1.0007	1.3769	1.9948	**21**	0.99286	0.96654	1.0328	1.9211
2	0.71048	1.0002	1.2831	1.9932	**22**	0.4684	0.995	1.6733	1.9561
3	0.58347	0.99296	0.89475	1.9949	**23**	0.8662	1.0216	2.2494	2.0035
4	0.85423	1.0028	1.5894	1.9886	**24**	0.45417	0.97115	1.7172	1.9513
5	0.70105	0.99973	1.3133	1.9935	**25**	0.79658	1.0218	2.6704	2.0034
6	0.9419	0.97986	0.70935	1.9623	**26**	0.59063	1.0212	0.98738	1.9959
7	0.58278	0.99271	0.86309	1.9946	**27**	1.0729	0.97972	0.97621	1.9648
8	0.70315	1.0017	1.1844	1.9938	**28**	0.935	1.0216	1.0017	1.9974
9	0.87479	1.0042	0.65982	1.9963	**29**	0.0016933	0.80188	1.4776	0.57284
10	0.26838	0.99546	1.2832	1.9831	**30**	1.0122	1.4183	1.9496	2.5435
11	2.363	0.98365	1.3819	1.9638	**31**	0.3112	1.2729	1.1683	2.3762
12	0.68805	1.0001	1.3075	1.9935	**32**	0.45864	1.404	1.1768	2.4779
13	0.73846	0.98667	0.29947	1.9702	**33**	0.98486	1.5459	2.0713	2.664
14	1.1277	0.99162	1.3081	1.9748	**34**	0.44047	0.84764	0.8807	1.8737
15	1.0043	0.96047	1.4413	1.9056	**35**	1.0096	1.4369	1.0957	2.2258
16	0.57646	0.98824	0.61454	1.9872	**36**	0.43645	1.2716	1.0456	2.1246
17	0.68744	1.0009	0.36339	1.9925	**37**	0.69385	1.3986	1.0155	2.1011
18	1.5733	0.98758	1.263	1.9606	**38**	1.2136	1.0307	0.98593	2.0296
19	0.99604	0.96603	1.0035	1.9334	**39**	1.2389	1.2187	1.028	2.0187
20	0.70176	0.97135	0.99021	1.9574	**AVG**	**0.80448**	**1.0629**	**1.2137**	**2.0086**

2.3 Stochastic Interest Rate Volatility Model

The second model we present is the stochastic interest rate volatility model. This model presents complications as an SDE such as non-diagonal, non-commutative noise. It also contains non-geometric Brownian motion, i.e. the assumptions of the Black-Scholes model that interest rate and volatility are constant are removed. We will also provide the result of Colgin et. al. [17], where a GPU algorithm is implemented for MLMC that provided an impressive speed-up to the simulation.

2.3.1 Parallel MLMC Sampling on GPUs

Graphics processing units (GPUs) play an important role in modern scientific computing. The architecture of these devices readily lends itself to parallel computation of threads executed with independent data sets. A GPU's architectural configuration is known as SIMD or Single Instruction Multiple Data. This is well suited for many sampling methods including Monte Carlo methods, especially those in which large groups of samples require the same number of steps to compute. The individual samples of the simulation will be computed in parallel over all the processing cores of the GPU, where each core is computing a different sample. In this context *multiple data* refers to the independent data sets that describe the random input for each individual sample (i.e. the Brownian path of the sample). For more information on the use of GPUs for scientific computing we refer the reader to [13].

The following algorithm may be used to parallelize a MLMC or an AMLMC simulation on a GPU [17]:

1. Define variables
   (a) Let G be the number of GPU threads to be used
   (b) Let ε be the desired error tolerance

(c) Let L be the number of levels in the MLMC

(d) Let L_0 be the initial level of the MLMC

(e) Let M^* be the number initial samples to determine the variance for each level

(f) Let R_ℓ be the collection of samples at level ℓ

2. For each level $L_0 \leq \ell \leq (L_0 + L - 1)$

(a) Assign each thread $\lceil M^*/G \rceil$ samples

(b) If $\ell = L_0$ then

i. Take 2^ℓ time steps to calculate the price

ii. Add the price to R_ℓ

(c) If $\ell > L_0$ then

i. Take 2^ℓ time steps to calculate the price P_ℓ

ii. Using the same normal random variables from the previous step, take 2^ℓ time steps to calculate the price P_ℓ^a. However, each pair of random variables is swapped for the antithetic estimator.

iii. Take $2^{\ell-1}$ time steps to calculate the price $P_{\ell-1}$

iv. Add the price correction $(P_\ell + P_\ell^a)/2 - P_{\ell-1}$ to R_ℓ

(d) Aggregate the results from all threads

(e) Estimate the variance V_ℓ at each level ℓ using R_ℓ

(f) Calculate the number of samples at level ℓ using the formula
$N_\ell = \lceil 2\varepsilon^{-2} \sqrt{V_\ell \delta t_\ell} \sum_{\ell=L_0}^{L_0+L-1} \sqrt{V_\ell / \delta t_\ell} \rceil$

3. For each level $L_0 \leq \ell \leq (L_0 + L - 1)$

(a) Assign each thread $\lceil (N_\ell - M^*)/G \rceil$ samples

(b) If $\ell = L_0$ then

i. Take 2^ℓ time steps to calculate the price

ii. Add the price to R_ℓ

(c) If $\ell > L_0$ then

i. Take 2^ℓ time steps to calculate the price P_ℓ

ii. Using the same normal random variables from the previous step, take 2^ℓ time steps to calculate the price P_ℓ^a. However, each pair of random variables is swapped for the antithetic estimator.

iii. Take $2^{\ell-1}$ time steps to calculate the price $P_{\ell-1}$

iv. Add the price correction $(P_\ell + P_\ell^a)/2 - P_{\ell-1}$ to R_ℓ

(d) Aggregate the results from all threads

2.3.2 Model Formulation

The model of interest describes an asset price with interest and volatility as separate stochastic processes. It takes the form

$$\begin{aligned} dS_t &= r_t S_t dt + \sqrt{v_t} S_t dW_{1,t}, \\ dv_t &= a(v_t)dt + b(v_t)dW_{2,t}, \\ dr_t &= \alpha(r_t,t)dt + \beta(r_t)dW_{3,t}, \\ \left\langle dW_{i,t} dW_{j,t} \right\rangle &= \pi_{i,j} dt, \end{aligned} \tag{2.15}$$

for $i, j = 1,2,3$, where S is asset price, r is interest rate, v is volatility, and $\pi_{i,j}$ is the correlation between $dW_{i,t}$ and $dW_{j,t}$.

To use this model with our scheme, it is adjusted to use uncorrelated random variables $dZ_{i,t}$ for $i = 1,2,3$ by using a Cholesky decomposition of the correlation matrix [39]. The

correlation matrix

$$\Pi = \begin{pmatrix} 1 & \pi_{12} & \pi_{13} \\ \pi_{12} & 1 & \pi_{23} \\ \pi_{13} & \pi_{23} & 1 \end{pmatrix}, \tag{2.16}$$

has a Cholesky factorization ($\Pi = UU^T$) of

$$U = \begin{pmatrix} 1 & 0 & 0 \\ \pi_{12} & c_\pi & 0 \\ \pi_{13} & \frac{\pi_{23}-\pi_{13}\pi_{12}}{c_\pi} & \sqrt{1-\pi_{13}^2-\left(\frac{\pi_{23}-\pi_{13}\pi_{12}}{c_\pi}\right)^2} \end{pmatrix}, \tag{2.17}$$

where $c_\pi = \sqrt{1-\pi_{12}^2}$.

However, according to [6], empirical data suggests that option pricing has no correlation between prices and interest rates; thererore, $\pi_{13} = \pi_{23} = 0$ and let $\pi_{12} = \pi$ for simplification of notation.

We then may use the identity (2.1) with the Cholesky factorization to relate uncorrelated random variables $d\boldsymbol{Z} = (dZ_{1,t}\ dZ_{2,t}\ dZ_{3,t})^T$ to correlated random variables $d\boldsymbol{W} = (dW_{1,t}\ dW_{2,t}\ dW_{3,t})^T$

$$d\boldsymbol{W}_t = Ud\boldsymbol{Z}_t = \begin{pmatrix} 1 & 0 & 0 \\ \pi & c_\pi & 0 \\ 0 & 0 & 1 \end{pmatrix} d\boldsymbol{Z}_t = \begin{pmatrix} dZ_t \\ \pi dZ_{1,t} + c_\pi dZ_{2,t} \\ dZ_{3,t} \end{pmatrix}, \tag{2.18}$$

Adjusting model (2.15) for uncorrelated random variables $d\boldsymbol{Z}_t$ gives

$$\begin{aligned} dS_t &= r_t S_t dt + \sqrt{v_t} S_t dZ_{1,t}, \\ dv_t &= a(v_t)dt + \pi b(v_t) dZ_{1,t} + c_\pi b(v_t) Z_{2,t}, \\ dr_t &= \alpha(r_t,t)dt + \beta(r_t) dZ_{3,t}. \end{aligned} \tag{2.19}$$

Which gives an Euler-Maruyama formulation of

$$\begin{aligned} S_{n+1} &= S_n(1 + r_n\Delta t + \sqrt{v_n}\Delta W_{1,n}), \\ v_{n+1} &= v_n + a(v_t)\Delta t + \pi b(v_t)\Delta W_{1,n} + c_\pi b(v_t)\Delta W_{2,n}, \\ r_{n+1} &= r_n + \alpha(r_n,t_n)\Delta t + \beta(r_n)\Delta W_{3,n}, \end{aligned} \tag{2.20}$$

and a Milstein formulation for AMLMC of

$$\begin{aligned} S_{n+1} &= S_n(1 + r_n\Delta t + \sqrt{v_n}\Delta W_{1,n}) \\ &\quad + \frac{1}{2}S_n\left[\left(v_n + \frac{\pi b(v_n)}{2\sqrt{v_n}}\right)(\Delta W_{1,n}^2 - \Delta t) + \frac{c_\pi b(v_n)}{2\sqrt{v_n}}(\Delta W_{1,n}\Delta W_{2,n})\right], \\ v_{n+1} &= v_n + a(v_t)\Delta t + \pi b(v_t)\Delta W_{1,n} + c_\pi b(v_t)\Delta W_{2,n} \\ &\quad + \frac{1}{2}b(v_n)\frac{\partial}{\partial v}b(v_n)\left[\pi^2(\Delta W_{1,n}^2 - \Delta t) + 2\pi c_\pi(\Delta W_{1,n}\Delta W_{2,n}) + c_\pi^2(\Delta W_{2,n}^2 - \Delta t)\right], \\ r_{n+1} &= r_n + \alpha(r_n,t_n)\Delta t + \beta(r_n)\Delta W_{3,n} + \frac{1}{2}\left[\left(\beta(r_n)\cdot\frac{\partial}{\partial r}\beta(r_n)\right)(\Delta W_{3,n}^2 - \Delta t)\right]. \end{aligned} \tag{2.21}$$

2.3.3 Numerical Experiments

A model from [52] was used for experimentation. This model used a European style option payoff. This example takes the form

$$
\begin{aligned}
dS_t &= r_t S_t dt + \sqrt{v_t} S_t dW_{1,t}, \\
dv_t &= \kappa_v(\bar{v} - v_t)dt + \sigma_v \sqrt{v_t} dW_{2,t}, \\
dr_t &= \kappa_r(\bar{r} - r_t)dt + \sigma_r \sqrt{r_t} dW_{3,t},
\end{aligned}
\tag{2.22}
$$

with $r(0) = r_0 = \bar{r}$, $v(0) = v_0 = \bar{v}$, and where interest rate r is uncorrelated with volatility v and price S.

Adding the drift and diffusion functions from (2.22) to (2.20)

$$
\begin{aligned}
S_{n+1} &= S_n(1 + r_n \Delta t + \sqrt{v_n}\Delta W_{1,n}), \\
v_{n+1} &= v_n + \kappa_v(\bar{v} - v_n)\Delta t + \sigma_v \sqrt{v_n}(\pi \Delta W_{1,n} + c_\pi \Delta W_{2,n}), \\
r_{n+1} &= r_n + \kappa_r(\bar{r} - r_n)\Delta t + \sigma_r \sqrt{r_n}\Delta W_{3,n},
\end{aligned}
\tag{2.23}
$$

and to (2.21)

$$
\begin{aligned}
S_{n+1} &= S_n(1 + r_n \Delta t + \sqrt{v_n}\Delta W_{1,n}) \\
&\quad + \frac{1}{2} S_n \left[\left(v_n + \frac{\pi \sigma_v}{2} \right)(\Delta W_{1,n}^2 - \Delta t) + \frac{c_\pi \sigma_v}{2}(\Delta W_{1,n} \Delta W_{2,n}) \right], \\
v_{n+1} &= v_n + \kappa_v(\bar{v} - v_n)\Delta t + \sigma_v \sqrt{v_n}(\pi \Delta W_{1,n} + c_\pi \Delta W_{2,n}) \\
&\quad + \frac{\sigma_v^2}{4} \left[\pi^2(\Delta W_{1,n}^2 - \Delta t) + 2\pi c_\pi (\Delta W_{1,n} \Delta W_{2,n}) + c_\pi^2(\Delta W_{2,n}^2 - \Delta t) \right], \\
r_{n+1} &= r_n + \kappa_r(\bar{r} - r_n)\Delta t + \sigma_r \sqrt{r_n}\Delta W_{3,n} + \frac{\sigma_r^2}{4}(\Delta W_{3,n}^2 - \Delta t).
\end{aligned}
\tag{2.24}
$$

For this example model, we wish to approximate the expected value of asset S at the terminal time T; therefore, our payoff function takes the form $P = S(T)$, which is an European style option.

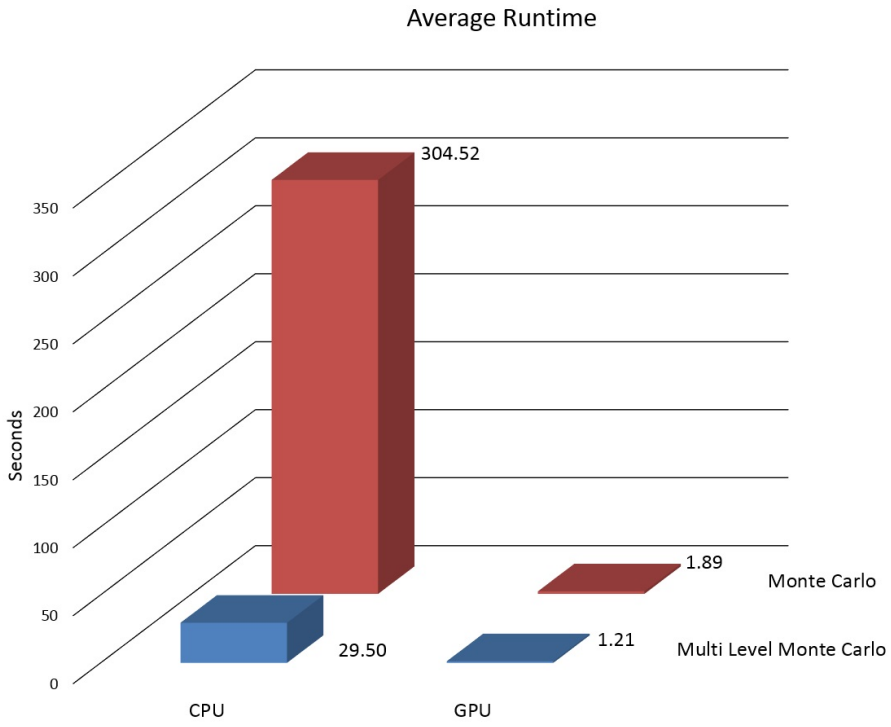
Using the parameter sets indicated in Table 5, a considerable speedup was achieved on

the test GPU[4].

Table 5: Stochastic Interest Rate Volatility Model Simulation Parameters

Parameter sets used for numerical testing with $S(0) = 100$, $\kappa_r = 0.3$, $\bar{r} = 0.04$, $\bar{v} = 0.02$, and $\sigma_r = 0.1$ [17].

#	1	2	3	4	5	6	7	8
K	100	100	110	110	100	100	110	110
T	0.25	0.50	0.25	0.50	0.25	0.50	0.25	0.50
v	0.01	0.01	0.04	0.04	0.04	0.04	0.04	0.04
K_v	1.50	1.50	0.75	0.75	1.50	1.50	1.50	1.50
σ_v	0.15	0.15	0.30	0.30	0.30	0.30	0.15	0.15
π	0.1	0.1	0.1	0.1	0.1	0.1	-0.5	-0.5

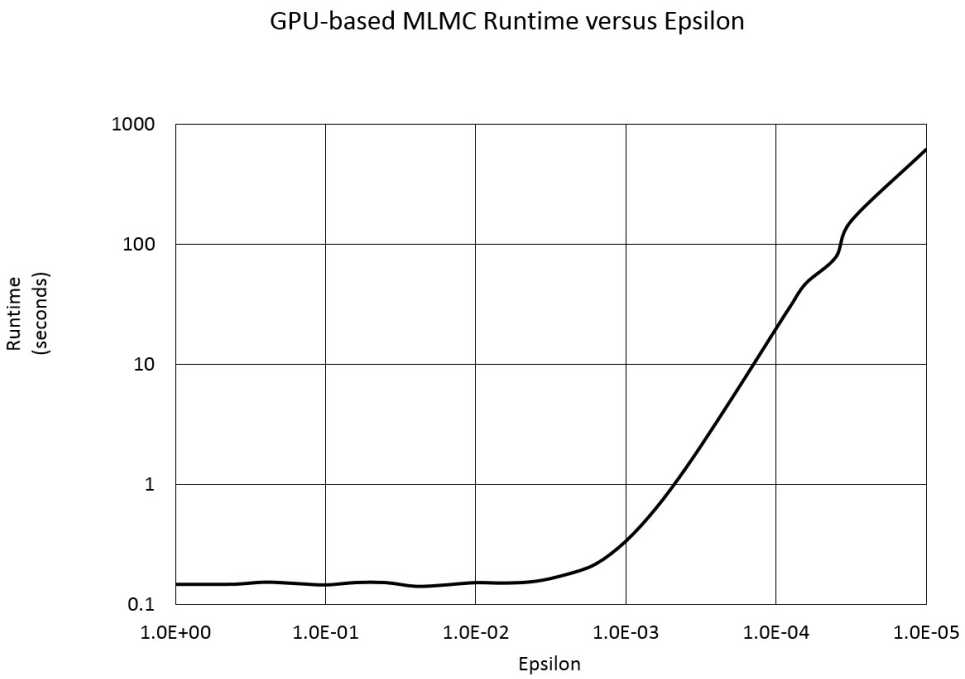


Performance of different methods versus platform [17].

Figure 5: Stochastic Interest Rate Volatility Model Simulation Platform Runtime

Figure 5 provides a graphical representation of how the Milstein antithetic method performed using the Monte Carlo and Multilevel Monte Carlo algorithm on the CPU and

[4]The testing environment was an M6800 with an Intel i7-4710MQ quad core CPU running at 2.50 GHz and a NVIDIA Quadro K3100M GPU with 768 cores and 4 GB of RAM.



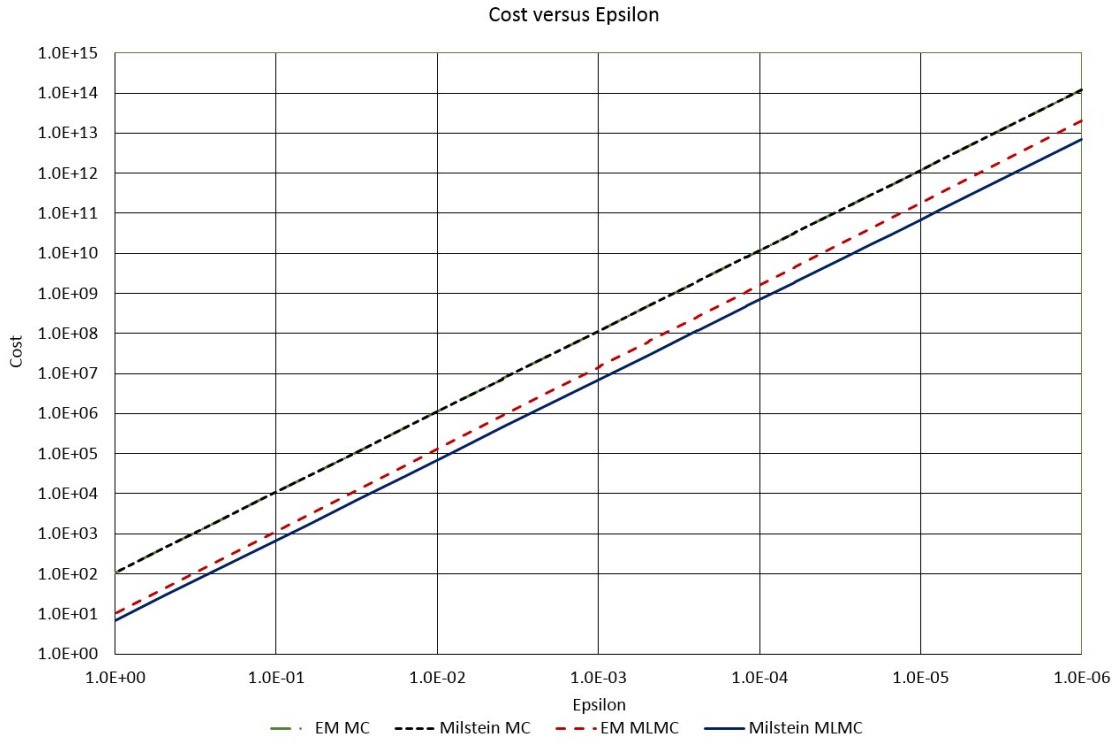
Impact of error tolerance (ε) on MLMC performance [17].

Figure 6: Stochastic Interest Rate Volatility Model Simulation ε Runtime

GPU. As expected Multilevel Monte Carlo is far more efficient than Monte Carlo, as it reduces the number of the most costly samples; the MLMC execution time is 9.7% of the Monte Carlo. The improvement to the execution time when moving to the GPU is just as dramatic. The GPU-based MLMC is a mere 4.1% of the equivalent CPU code. When compared to the original CPU-based Monte Carlo, the GPU-based MLMC is about 0.4%.

The selection of an appropriate error tolerance is critical. It impacts not only the accuracy of the result but also the performance of the Multilevel Monte Carlo. Figure 6 shows the tradeoff between accuracy and performance. An ε of 10^{-3} provides an execution time of about one third of a second while an ε of 10^{-4} results in a twenty second runtime. The GPU takes about 600 seconds for $\varepsilon = 10^{-5}$.

In addition to runtime, the methods can be evaluated in terms of computational cost.

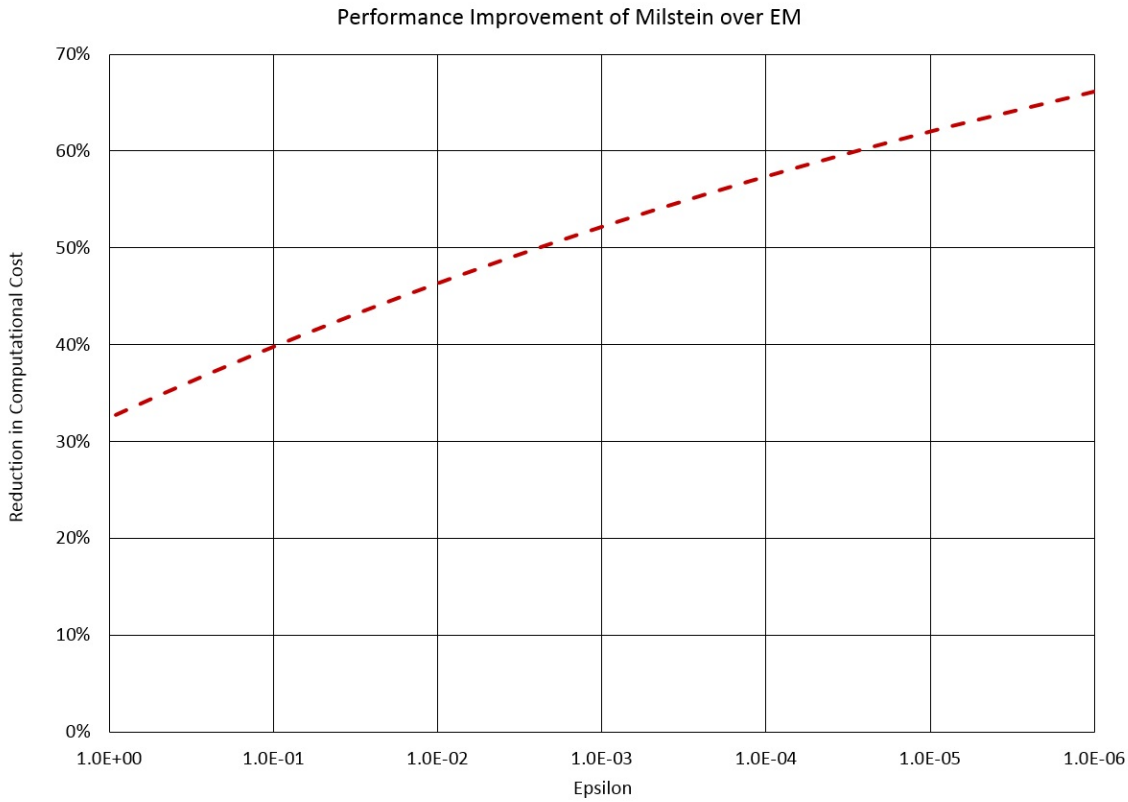


Error tolerance (ε) versus computational cost [17].

Figure 7: Stochastic Interest Rate Volatility Model Simulation ε Cost

This eliminates the differences in hardware and allows the various methods to be compared using a platform agnostic metric. Figure 7 shows the computational cost of the four methods explored. Computational cost is defined as the total number of time steps taken. The two AMLMC methods are significantly more efficient over the standard Monte Carlo methods. The Milstein discretization adds a marked improvement to the MLMC method. Figure 8 quantifies the reduction in computational cost for the AMLMC when moving from Euler-Maruyama to Milstein.

These results display the major advantage that GPUs have in Monte Carlo-style problems, as they can execute a large number of independent threads simultaneously. This is an ideal application since each simulation is independent from others and does not require any interprocess communication. Multilevel Monte Carlo simulations can be further parallelized



Improvement of Milstein over Euler-Maruyama for MLMC [17].

Figure 8: Stochastic Interest Rate Volatility Model Simulation Savings

as each level within a single simulation can be executed simultaneously and thus benefiting applications where the calculation of the level is computationally expensive.

The GPU implementation of AMLMC is extremely computationally efficient for the example model. This approach need not be limited to this model or even to the area of finance. Many SDE systems would benefit from such an approach, particularly those with non-diagonal or non-commutative noise. For Asian, lookback, barrier and digital options special numerical treatments have to be used [29] to achieve $\mathcal{O}(\varepsilon^{-2})$ scaling of the computational cost.

CHAPTER 3

SPLIT-STEP METHODS FOR STIFF SYSTEMS: A CASE STUDY IN CHEMICAL REACTION NETWORKS

Chemical reaction networks (CRNs) are a means by which we may model concentrations of molecular species (or chemical reactants) in dilute, well-mixed volumes as they change over time. CRNs are commonly used to model both chemical and biochemical systems[1]. These networks can be computationally difficult to simulate due to their complexity and time scale. For example, the stochastic Petri network model for the heat shock response of E. Coli described in [66, 67] is comprised of 14 molecular species, 17 chemical reactions with reaction rates ranging from $\mathscr{O}(10^{-11})$ s^{-1} to $\mathscr{O}(10^{6})$ s^{-1}, and has initial conditions ranging from concentrations of $\mathscr{O}(1)$ to $\mathscr{O}(10^{6})$. This model is also a good example of a stiff system, which is a main focus of this chapter.

Stiff systems pose a particular problem when modeled by SDEs. Similarly to stiff deterministic systems modeled by ordinary differential equations (ODEs), numerical methods for solving stiff SDEs can become unstable at large time-steps. In Chapter 2, we use the MLMC method to simulate systems of non-stiff SDEs. Lacking stiffness in an SDE model allows for the initial level of an MLMC simulation to have a single time-step for each sample, i.e. for $\ell = 0$ that $\Delta t_0 = T$ for samples $1, 2, ..., M_0$. As will be shown, the same approach should not be applied to a stiff system without an appropriate numerical method. In this chapter we utilize methods with large stability regions, namely partially implicit split-step

[1] *The Journal of Chemical Physics* is a great resource for the latest developments in the modeling and simulation of CRNs. See the work of Thomas Kurtz and David Anderson (e.g. [5, 49]) for a background in CRNs as it relates to this work. Dan Gillespie has also had an immense contribution to this field. For many of his largest breakthrough studies on the topic, particularly in the simulation of CRNs, see Gillespie et. al. in [11, 12, 21, 30, 31, 32, 33, 34, 35, 36, 37, 61].

methods. Using these methods, large step sizes can be used at lower levels of MLMC so that samples may be generated without oscillations, and the simulation will, therefore, converge at the expected rate. This allows for coarse levels to be used whose samples are much more efficient to compute.

In the following sections, we provide the various required definitions and describe many techniques for simulating CRNs. We then give the necessary background for split-step methods including mean square stability (MS-stability). Next, we perform our strategy on a small, stiff test SDE from [65, 68], before finally implementing our approach on a CRN from [3].

3.1 Modeling and Simulation of Discrete Valued Chemical Reaction Networks

Chemical reaction networks are a common way to model concentrations of reactants in a system. Essential to CRNs is the study of chemical kinetics, i. e. the study of the rates of chemical processes. CRNs attempt to describe the number of chemical reactions within a system that determine molecular concentration for all involved chemical species. The chemical kinetics inherent to a CRN are defined by rate; therefore, it is intuitive that we may use differential equations to model these systems. However, there are many ways to model CRNs. Before continuing with our approach of using SDEs, we will describe some of the difficulties involved and some of the most well known and useful techniques for simulation.

It is particularly difficult to create accurate, predictive models for these systems. The number of unique chemical species, the initial concentrations of each of these species, the number of potential reactions, and the rate in which each reaction occurs must all be factored into these models. These systems can have rather large concentrations, and the reactions may occur at rates differing by several orders of magnitude.

Using classical mechanics on a system of molecules bouncing around is entirely too

complex and of such high dimension that this approach should be disregarded at the current state of computational science. Many different techniques are available that take a probabilistic approach to these systems to reduce the dimension to a conceivable order. These methods also differ in accuracy and complexity. We will discuss a few here and establish some notation.

3.1.1 CRN Formulation

Let set S of d reactants be the particular chemical species in the system, i.e. $S_i \in S$ for $i = 1,2,...,d$. Let R be the set of D reactions such that the concentrations of one or more species is affected for each $R_j \in R$ for $j = 1,2,...,D$. These sets represent a CRN if every species in S either: 1) reacts in some way in the system such that the concentrations of all or some species in S change with time (left hand side of reaction) or 2) has its own concentration changed due to one or more reaction in R (right hand side of reaction). Each reaction also has an associated reaction rate. For the purposes of this work, the reaction rate will be assumed to be constant; however reaction rates may be functions of the system state. More complex models may also take into consideration other important physical properties, such as the energy in the system.

The state of the system is described by $\boldsymbol{X}(t)$, where $X_i(t) \in \boldsymbol{X}(t)$ represents the concentration of species S_i at time t. It is clear that $\boldsymbol{X}(t)$ is a discrete state system, because for every concentration $X_i \in \boldsymbol{X}$ there is a non-negative integer number of molecules. Since for every reaction the system state changes by a discrete amount, a reaction can be called a jump event. The following is an example CRN from [10] provided to better illustrate a CRN formulation.

Let our CRN contain $d = 5$ chemical species $S = \{S_1, S_2, S_3, S_4, S_5\}$ and $D = 3$ reactions

$R = \{R_1, R_2, R_3\}$, where

$$\begin{aligned} R_1 &: S_1 + S_2 \xrightarrow{c_1} S_3 + S_4, \\ R_2 &: 2S_3 \xrightarrow{c_2} S_5, \\ R_3 &: S_4 \xrightarrow{c_3} S_2, \end{aligned} \tag{3.1}$$

and c_1, c_2, and c_3 are the rate constants for each reaction. We define the stoichiometric vectors by the incremental changes made to each species S_i by each of the reactions, i.e.

$$\boldsymbol{v}_1 = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{v}_2 = \begin{bmatrix} 0 \\ 0 \\ -2 \\ 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{v}_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \\ 0 \end{bmatrix}. \tag{3.2}$$

So, for example, $\boldsymbol{v}_1 = [-1 \ -1 \ 1 \ 1 \ 0]^T$ in (3.1), because during reaction R_1 one molecule of each species S_1 and S_2 is lost, one molecule of each species S_3 and S_4 is gained and no change is made to the concentration of species S_5. As is can be helpful for some formulation, a stoichiometric matrix or (state change matrix) V can be constructed from column vectors $\boldsymbol{v}_j$, i.e.

$$V = \begin{bmatrix} -1 & 0 & 0 \\ -1 & 0 & 1 \\ 1 & -2 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{3.3}$$

Clearly in this representation each column is representative of a reaction and each row is

representative of a species.

Propensity functions (or intensity functions) $a_j : \mathbb{Z}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ are of combinatorial design and represents the probability that a particular reaction should occur in unit time. Another way to interpret the value of these functions is as the measurable tendency that a certain reaction is to occur given the current state of the system. The function $a_j(\cdot)$ is defined by multiplying the reaction rate of the corresponding reaction R_j by the total combinations of applicable molecules from the left hand side of the reaction definition . An explanation of this construction can be found in [10, 43]. For the example CRN (3.1), the propensity functions are

$$\begin{aligned} a_1(\boldsymbol{x}) &= c_1 x_1 x_2, \\ a_2(\boldsymbol{x}) &= c_2 x_3 (x_3 - 1)/2, \\ a_3(\boldsymbol{x}) &= c_3 x_4. \end{aligned} \tag{3.4}$$

Similarly to (3.3), it can be helpful to have a matrix form for (3.4). In this case the vector valued propensity function $\boldsymbol{a} : \mathbb{Z}^d_{\geq 0} \to \mathbb{R}^D_{\geq 0}$ is given by

$$\boldsymbol{a}(\boldsymbol{x}) = \begin{bmatrix} c_1 x_1 x_2 \\ c_2 x_3 (x_3 - 1)/2 \\ c_3 x_4 \end{bmatrix}. \tag{3.5}$$

3.1.2 The Chemical Master Equation

A random process that meets the Markov property and changes with jump events can be described as a discrete state Markov process or a jump Markov process [10, 43]. Continuous time Markov chains are used to maintain the property that each $X_i \in \boldsymbol{X}(t)$ has a non-negative integer value for $i = 1, 2, ..., d$ and every value of t, i.e. $\boldsymbol{X} : \mathbb{R}_{\geq 0} \to \mathbb{Z}^d_{\geq 0}$. The chemical

master equation (CME) is the mathematically precise representation for a CRN under this modeling approach. It takes the form

$$\frac{\partial}{\partial t}P(\boldsymbol{x},t|\boldsymbol{X}_0,t_0) = \sum_{j=1}^{D}[a_j(\boldsymbol{x}-\boldsymbol{v}_j)P(\boldsymbol{x}-\boldsymbol{v}_j,t|\boldsymbol{X}_0,t_0) - a_j(\boldsymbol{x})P(\boldsymbol{x},t|\boldsymbol{X}_0,t_0)],$$

where $P(\boldsymbol{x},t|\boldsymbol{X}_0,t_0)$ is the probability that $\boldsymbol{X}(t)=\boldsymbol{x}$ given that $\boldsymbol{X}(t_0)=\boldsymbol{X}_0$ for initial time t_0 and initial concentrations $\boldsymbol{X}_0$ [34]. The solution to the CME perfectly describes the probability distribution of all possible system states as the distribution evolves over time. Though precise, the exact solution to the CME is rarely obtainable [34].

Another option available is to sample many possible outcomes of the system and compute approximations of the moments to the state of the system at desired times. Methods of uncertainty quantification can then be used to establish a confidence interval for a given error tolerance. There are many ways to generate these samples, and analogous to (1.12), the average value of a functional on those samples is known as the Monte Carlo approximation to the expected value of that functional. As was discussed in Chapter 1, Monte Carlo samples are many times referred to as paths. For this application the term 'path' is also well suited, as the sample represents a particular realization of a continuous time Markov chain [3].

Many path generation methods exist. A sample generation method may produce an exact path, i.e. the path represents one possible realization of $\boldsymbol{X}(t)$ with no bias, or it may produce an approximate path where a known order of error would be necessary information for the method. The next sections present several well known sampling methods.

3.1.3 Exact Paths

Multiple algorithms exist across the sciences that compute exact sample paths of continuous time Markov chains. Some are more efficient than others, but they all scale linearly with the number of jump events (e.g. a chemical reaction) [3]. An exact path algorithm

should generate the statistics of the CME as the number of paths $M \to \infty$ in a way that we can quantify [12, 43].

In the field of chemical kinetics the most notable examples for exact path simulation are the stochastic simulation algorithm (SSA) [10, 12, 43], the next reaction method (NRM)[10], and the modified next reaction method (MNRM) [2]. MNRM claims to be the most efficient of the three [3, 2]; however, as an example we have chosen to describe SSA as it is the most classical approach among the listed methods and gives fundamental insight to the chemical process. An exact method is also necessary in the context of this work as verification step for the results of our SDE MLMC simulations.

SSA simulates a sample path by determining two things at each time step: 1) the time until the next reaction and 2) which reaction is to take place. The time until the next reaction (or time step τ) is determined by an exponential probability density function (PDF) of the form

$$\mathbf{P}(\tau \mid \boldsymbol{X}(t)) = a_0\big(\boldsymbol{X}(t)\big) \exp\Big(- a_0\big(\boldsymbol{X}(t)\big)\tau\Big). \tag{3.6}$$

where $a_0(\boldsymbol{X}(t)) = \sum_j^D a_j(\boldsymbol{X}(t))$. Computationally we can sample τ from this distribution with

$$\tau = \frac{1}{a_0(\boldsymbol{X}(t))} \ln\left(\frac{1}{r_1}\right), \tag{3.7}$$

where r_1 is a uniform random number in the unit interval [12].

The next reaction that is to take place is determined by the PDF

$$\begin{aligned} \mathbf{P}\Big(R_j \text{ occurs in the interval } [t, t+\tau) \mid \boldsymbol{X}(t)\Big) = \\ \mathbf{P}\Big(\boldsymbol{X}(t+\tau) = \boldsymbol{X}(t) + \boldsymbol{v}_j \mid \boldsymbol{X}(t)\Big) = \frac{a_j(\boldsymbol{X}(t))}{a_0(\boldsymbol{X}(t))}, \end{aligned} \tag{3.8}$$

which simply states that the probability that reaction j occurs between time t and $t+\tau$ is that reactions share of the total propensity. Computationally we can sample this distribution by taking the smallest integer j such that

$$\sum_{j'=1}^{j} a_{j'}(\boldsymbol{X}(t)) > r_2 a_0(\boldsymbol{X}(t)), \tag{3.9}$$

where r_2 is sampled from the uniform distribution in the unit interval [12].

It is clear why these algorithms scale linearly with the number of jump events; only one reaction occurs at each step. Due to the small time scale in which chemical reactions usually take place coupled with the size and concentration of these systems, these algorithms can easily produce tens of thousands or more time steps in small time spans, even for basic chemical systems. Many approximate path schemes attempt to reduce the number of time steps taken by holding the propensity function constant over a period of time and approximating how many reactions took place during that span, thereby reducing the state updates of $\boldsymbol{X}(t)$. Some of these approaches will be explained in the next section.

3.1.4 Approximate Paths

Approximate path algorithms have been developed to mitigate the computational complexity of exact path methods. Two categories of approximate methods will be discussed in this section: 1) methods that retain the property $\boldsymbol{X}(t) \in \mathbb{Z}^d_{\geq 0}$ by using Poisson random variables to determine the number of each reaction that occurs during each time step and 2) a method that relaxes this property by allowing a continuous, non-negative valued state vector, i.e. $\boldsymbol{X}(t) \in \mathbb{R}^d_{\geq 0}$.

To retain an integer valued system state, we use the random time change representation

$$\boldsymbol{X}(t) = \boldsymbol{X}(0) + \sum_{j=1}^{D} Y_j \left(\int_0^t a_j\big(\boldsymbol{X}(s)\big) ds \right) \boldsymbol{v}_j, \tag{3.10}$$

where Y_j represents independent unit-rate Poisson processes [3, 22, 50]. This representation is exact; however, many approximation methods are generally used to compute the integral.

One of the most basic numerical integration techniques for definite integrals in one dimension divides the limits of integration into a sequence of N non-overlapping subintervals; i.e. for limits of integration $[a,b]$, the subdivision takes the form of sequence $\langle [x_n, x_{n+1}] \mid n \in 0,1,...,N-1 \rangle$ such that

$$\begin{aligned} [a,b] &= \bigcup_{n=0}^{N-1} [x_n, x_{n+1}], \\ \emptyset &= \bigcap_{n=0}^{N-1} [x_n, x_{n+1}). \end{aligned} \tag{3.11}$$

The integrand is then computed at a value on each interval, e.g. $f(\xi_n)$ where $\xi_n \in [x_n, x_{n+1}]$), and a Riemann sum is then made on the product of that value and the width of the corresponding interval[2], i.e.

$$\sum_{n=0}^{N-1} f(\xi_n)\Delta x_n \approx \int_a^b f(x)dx, \tag{3.12}$$

where $\Delta x_n = x_{n+1} - x_n$. In this method the integrand is approximated as constant over each interval, which is analogous to the explicit Euler method, a well-known method for approximating ordinary differential equation initial value problems (ODE IVP).

The ODE IVP takes the form

$$\frac{d\boldsymbol{X}(t)}{dt} = \boldsymbol{f}(t, \boldsymbol{X}(t)), \quad \boldsymbol{X}(0) = \boldsymbol{X}_0, \tag{3.13}$$

where $\boldsymbol{X}, \boldsymbol{f} \in \mathbb{R}^d$. For the solution of this problem at terminal time T, the fundamental

[2]If the minimum value is used, i.e. $\xi_n = x_n$, this is called the *left Riemann sum approximation*. Note that fixed-length intervals are common as they are the easiest to implement, i.e. $\Delta x_n = (b-a)/N \; \forall \, n$.

theorem of calculus gives

$$\boldsymbol{X}(T) = \int_0^T \frac{d\boldsymbol{X}(\zeta)}{d\zeta} = \int_0^T \boldsymbol{f}(\zeta, \boldsymbol{X}(\zeta))d\zeta, \tag{3.14}$$

Using the left Riemann sum approximation version of (3.12) with fixed step sizes, (3.14) is approximated by

$$\sum_{n=0}^{N-1} \boldsymbol{f}(\zeta_n, \boldsymbol{X}(\zeta_n))\Delta\zeta \approx \boldsymbol{X}(T) = \int_0^T \boldsymbol{f}(\zeta, \boldsymbol{X}(\zeta))d\zeta, \tag{3.15}$$

where $\Delta\zeta = T/N$ and $\zeta_n = n\Delta\zeta$. Since the values $\boldsymbol{X}(\zeta_n)$ are unknown (except for $\boldsymbol{X}(\zeta_0)$ as it is given in the IVP), the approximation $\boldsymbol{X}_n \approx \boldsymbol{X}(\zeta_n)$ is used, which is simply the value of the summation through the n^{th} term. For this reason numerical methods of this type are generally written as explicit formulas defined for single steps. The summation on the left hand side of (3.15) written as an explicit formula with approximate values $\boldsymbol{X}_n$ is

$$\boldsymbol{X}_{n+1} = \boldsymbol{X}_n + \boldsymbol{f}(\zeta_n, \boldsymbol{X}_n)\Delta\zeta, \tag{3.16}$$

where the initial value is given in (3.13).

The main difference between the Euler method (3.16) and Riemann sum integral approximations such as (3.12) is that the integrand in (3.16) is a function of the solution; therefore, each step requires the approximation from the previous step with an initial value required to begin. Another important quality with the Euler method is that function $\boldsymbol{f}$ in (3.16) does not need to be an integrand. For instance, the integrand in (3.10) does not map to the same domain as the solution, i.e. $a_j : \mathbb{Z}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ and $\boldsymbol{X}(t) \in \mathbb{Z}^d_{\geq 0}$. However the summation term in (3.10) does map to the same domain, i.e.

$$\sum_{j=1}^{D} Y_j \left(\int_0^t a_j\big(\boldsymbol{X}(s)\big)ds \right) \boldsymbol{v}_j, \;\; \boldsymbol{X} \in \mathbb{Z}^d_{\geq 0}. \tag{3.17}$$

Therefore function $\boldsymbol{f}$ in (3.16) for the approximation of (3.10) is actually the entire summation term and not the integrand. The resulting method is known as Euler τ-leaping[3].

$$\boldsymbol{X}_{n+1} = \boldsymbol{X}_n + \sum_{j=1}^{D} Y_{j,n}\Big(a_j(\boldsymbol{X}_n)\,\tau\Big)\boldsymbol{v}_j, \tag{3.18}$$

where $\boldsymbol{X}_n \approx \boldsymbol{X}(t_n)$, $\tau = t_{n+1} - t_n$, and $Y_{j,n}$ represents independent unit-rate Poisson processes on the time interval (t_n, t_{n+1}).

Many adaptive Euler τ-leaping methods exist, and much work has been done to to control the error by choosing appropriate values for τ depending on the system state [12]. These methods can have complex implementations, but can be beneficial for sufficiently complex systems. However, it is unclear if these methods can work with MLMC, which uses multiple, coordinated fixed-step sizes.

3.1.5 Uncertainty Quantification for Monte Carlo Simulation of CRNs

To approximate the number of paths necessary to achieve a desired level of confidence for an error tolerance for Monte Carlo simulation of CRNs, we have included the procedure from [3]. For demonstration, consider functional $P(\boldsymbol{X}(T)) = X_q(T)$ as the quantity of interest for the simulation, where S_q is the particular chemical species of interest and T is the terminal time of the simulation. Using the following equality we can estimate various qualities about our estimations:

$$\frac{1}{n}\mathrm{Var}\big(\tilde{X}_q(T)\big) = (\tilde{\varepsilon}/z)^2, \tag{3.19}$$

where $\mathrm{Var}(\cdot)$ is variance of its parameter, z is the deviation (z-score) given by the normal inverse cumulative distribution function Φ^{-1} for a given confidence level, $\tilde{X}_q(T)$ is the

[3]The approach of holding the propensity function constant for time-step τ can often be found in the literature as τ-leaping. Due to the similarities with Euler's method, we have chosen to follow David Anderson and Desmond Higham's approach and describe the method as Euler τ-leaping [3].

normalized quantity of interest, and $\tilde{\varepsilon}$ is the normalized error bounds, i.e. confidence interval $X_q(T) \pm \varepsilon \implies \tilde{X}_q(T) \pm \tilde{\varepsilon}$ where $\tilde{X}_q(T) = 1$ and $\tilde{\varepsilon} = \varepsilon / X_q(T)$.

If we wish to determine the number of samples n necessary for a given confidence interval and error bounds, we first estimate the order of the variance of $X_q(T)$ by simulating a few sample paths. Usually 100 is enough because variance converges much faster than the expected value, as demonstrated in Chapter 2. Then we use the normal inverse cumulative distribution function Φ^{-1} to give us a value z. Typically we use at least a 95% confidence interval, giving $z = 1.96$.

Likewise, if we wish to determine the confidence interval that we achieved during our simulation given the error bounds and number of paths that the simulation generated, we first calculate z using the equality above and then use the normal cumulative distribution function $\Phi(z)$ to find our confidence interval.

3.1.6 Multilevel Monte Carlo for Discrete Valued CRNs

The MLMC implementation described in Chapter 2 was performed on random processes driven by Brownian motion. The Poisson processes used in Euler τ-leaping require a different approach. The authors in [3] demonstrate biased and unbiased MLMC schemes for continuous time Markov chains. We haven included an example from [3] to demonstrate how coupling is achieved with Poisson processes in MLMC.

For fixed step size algorithms let $\boldsymbol{X}_\tau(t)$ denote an approximation to the system state where the approximating method has a time step on the magnitude τ. Recall that the distribution of sample paths is not always the true distribution of $\boldsymbol{X}(t)$. Exact methods, such as SSA, do produce the true distribution of $\boldsymbol{X}(t)$ as the number of samples $M \to \infty$. However, the estimator $\boldsymbol{X}_\tau(t)$ produces the distribution that the approximation describes, which includes approximation bias; therefore, MLMC implementations for CRNs that only use approximate Euler τ-leaping methods are all biased.

Consider two Poisson processes $Z_1(t)$ and $Z_2(t)$ which have rates of 13.1 and 13 respectfully and let Y_1 and Y_2 be independent unit-rate Poisson processes giving

$$Z_1(t) = Y_1(13.1t) \quad \text{and} \quad Z_2(t) = Y_2(13t). \tag{3.20}$$

The variance of the difference $Z_1(t) - Z_2(t)$ is the sum of their variances, i.e.

$$\text{Var}(Z_1(t) - Z_2(t)) = \text{Var}(Z_1(t)) + \text{Var}(Z_2(t)) = 26.1t. \tag{3.21}$$

Alternatively we could construct $Z_1(t)$ and $Z_2(t)$ with independent unit-rate Poisson processes Y_1 and Y_2 in the following way

$$Z_1(t) = Y_1(13t) + Y_2(0.1t) \quad \text{and} \quad Z_2(t) = Y_1(13t). \tag{3.22}$$

Since Z_1 will jump together with Z_2 a majority of the time they are tightly coupled. We can write this more generally by introducing a third unit-rate Poisson process Y_3 and let

$$Z_1(t) = Y_1\left(\int_0^t f(s) \wedge g(s) ds\right) + Y_2\left(\int_0^t f(s) - (f(s) \wedge g(s)) ds\right), \tag{3.23}$$

$$Z_2(t) = Y_1\left(\int_0^t f(s) \wedge g(s) ds\right) + Y_3\left(\int_0^t g(s) - (f(s) \wedge g(s)) ds\right), \tag{3.24}$$

where we define $a \wedge b \stackrel{\text{def}}{=} \min\{a, b\}$.

At this point we can couple approximations of different levels. We have three Poisson processes but one is always zero and another has a low variance. It should be noted here that the higher the variance of a Poisson process, the more computationally costly it is to generate.

Let $\eta_\ell(s) \stackrel{\text{def}}{=} \lfloor s/h_\ell \rfloor h_\ell$ and let $Y_{j,k}$ be an independent, unit-rate Poisson process for

$k \in \{1,2,3\}$ and $j \in \{1,2,...,D\}$. Now we can construct an approximation using two different levels,

$$\begin{aligned} Z_\ell(t) &= Z_\ell(0) + \sum_j^D Y_{j,1}\left(\int_0^t a_j(Z_\ell \circ \eta_\ell(s)) \wedge a_j(Z_{\ell-1} \circ \eta_{\ell-1}(s))ds\right)\boldsymbol{\nu}_j \\ &+ \sum_j^D Y_{j,2}\left(\int_0^t a_j(Z_\ell \circ \eta_\ell(s)) - a_j(Z_\ell \circ \eta_\ell(s)) \wedge a_j(Z_{\ell-1} \circ \eta_{\ell-1}(s))ds\right)\boldsymbol{\nu}_j, \end{aligned} \tag{3.25}$$

$$\begin{aligned} Z_{\ell-1}(t) &= Z_{\ell-1}(0) + \sum_j^D Y_{k,1}\left(\int_0^t a_j(Z_\ell \circ \eta_\ell(s)) \wedge a_j(Z_{\ell-1} \circ \eta_{\ell-1}(s))ds\right)\boldsymbol{\nu}_j \\ &+ \sum_j^D Y_{j,3}\left(\int_0^t a_j(Z_{\ell-1} \circ \eta_{\ell-1}(s)) - a_j(Z_\ell \circ \eta_\ell(s)) \wedge a_j(Z_{\ell-1} \circ \eta_{\ell-1}(s))ds\right)\boldsymbol{\nu}_j, \end{aligned} \tag{3.26}$$

With this coupled relationship at least one term will be zero for every j. The computation for these terms is reduced while computing unique paths. However, this scheme is biased and the authors of [3] propose a way to couple an approximate method with an exact one to create an unbiased method. They achieve this by simply replacing the finest step-sized approximation with the modified next reaction method (MNRM), which can be found in [2].

As in our Monte Carlo scheme, MLMC requires a preliminary batch of paths to be computed to estimate the number of paths necessary for a given error and confidence. However, this is a more complicated problem because we must choose the number of paths for each level while maintaining the variance of the estimator under a desired amount. Therefore, a preliminary optimization problem must be constructed to determine the optimal number of paths per level n_ℓ to give the best CPU time and desired accuracy.

First let CPU_ℓ be the CPU time at level ℓ and V_ℓ be $\text{Var}(\hat{Q}_\ell)$, where $\hat{Q}_\ell$ is our estimator at level ℓ. We know that both CPU_ℓ and $1/V_\ell$ scale linearly with n_ℓ so we then introduce a constant K_ℓ such that

$$\text{CPU}_\ell \approx \frac{K_\ell}{V_\ell}.$$

We use the data from our preliminary simulation to estimate K_ℓ. Then we construct an optimization problem to minimize CPU time while keeping the total Variance of our estimator under the required level.

Minimize

$$\sum_\ell \frac{K_\ell}{V_\ell},$$

subject to

$$\sum_\ell V_\ell = (\varepsilon/z)^2,$$

where ε is our error and $\Phi(z)$ is our confidence interval. Note that the first level ℓ_0 and last level L are left open to choose before the optimization [3].

3.2 Modeling and Simulation of CRNs with the Chemical Langevin Equation

If we relax the condition that our state vector $\boldsymbol{X}(t)$ only contains non-negative integer values then it is possible to approximate $\boldsymbol{X}(t)$ as a system of ODEs as in the reaction rate equations (RRE) [10, 43]. Furthermore, if we wish to approximate many sample paths and collect statistical information for our model we can construct a system of SDEs. This is the case with the chemical Langevin equation (CLE) [10, 43, 34].

The CLE for a CRN with d different types of molecules and D types of chemical

reactions is an SDE in the form

$$d\boldsymbol{X}(t) = \sum_{j=1}^{D} a_{j,t}\boldsymbol{v}_j dt + \sum_{j=1}^{D} \sqrt{a_{j,t}}\boldsymbol{v}_j dW_j(t), \tag{3.27}$$

where $a_j : \mathbb{Z}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ and $\boldsymbol{v}_j$ for $j \in 1,2,...,D$ are the associated propensity functions and state change vectors respectfully.

The CLE can very easily be a stiff system as described in the introduction to this chapter. We use partially implicit split-step methods to handle the stiffness inherent to many CLEs. In this section, we will cover the benefits of these methods and utilize those benefits to accelerate MLMC. Note that we assume that the SDE is well-defined, has a unique solution, and all first and second moments and correlation coefficients exist. As mentioned in [45], establishing the conditions under which these properties hold is an open problem.

3.2.1 Split-Step Methods for SDEs

We will compare the Euler-Maruyama method (1.4) and the Milstein method (1.5) with the Split-Step Backward Euler method (SSBE) and the Drifting Split-Step Backward Milstein (DSSBM) [68, 70]. The SSBE method has the form

$$\begin{aligned} \hat{y_n} &= y_n + hf(t_n, \hat{y_n}), \\ y_{n+1} &= \hat{y_n} + \Delta W_n g(t_n, \hat{y_n}), \end{aligned} \tag{3.28}$$

and the DSSMB takes the form

$$\begin{aligned} \hat{y_n} &= y_n + hf(t_n, \hat{y_n}), \\ y_{n+1} &= \hat{y_n} + \Delta W_n g(t_n, \hat{y_n}) + \frac{1}{2} g(t_n, \hat{y_n}) g'(t_n, \hat{y_n})[(\Delta W_n)^2 - h]. \end{aligned} \tag{3.29}$$

Although these methods share the same strong orders of convergence with their fully

explicit counterparts (i.e. Euler-Maruyama and SSBE have strong order of convergence $\frac{1}{2}$ and Milstein and DSSBM have strong order of convergence 1), the stability properties of these methods vary greatly.

Stability of a SDE numerical scheme may also be defined in different ways. We may look at stability for the path trajectory, the strong sense, and stability with respect to the moments, the weak sense [64]. In [69] the authors define an approach taken by [64] as mean-squared stability (MS-stability). The approach of [69] has been used in several papers that we looked at during our literature survey and we will discuss their method to determine MS-stability here.

First, a linear test equation is applied to determine stability regions for the scheme,

$$\begin{cases} dX(t) = \lambda X dt + \mu X dW(t),\ t \in [0,T], \\ X(0) = X_0, \end{cases} \tag{3.30}$$

where the solution is

$$X(t) = X_0 \exp\left((\lambda - \frac{1}{2}\mu^2)t + \mu W(t) \right). \tag{3.31}$$

The method is MS-stable if

$$\bar{R}(\lambda,\mu,h) = \mathbb{E}(R^2(\lambda,\mu,h,J)) < 1, \tag{3.32}$$

We define $R(\lambda,\mu,h,J)$ as the ratio such that,

$$\bar{X}_{n+1} = R(\lambda,\mu,h,J)\bar{X}_n, \tag{3.33}$$

where $J = \frac{\Delta W_n}{\sqrt{h}}$. Then we let $p = \lambda h$ and $q = \mu\sqrt{h}$ and use the inequality

$$R(p,q) = \mathbb{E}(R^2(p,q,J)) < 1, \tag{3.34}$$

be our MS-stability region.

Using this approach we find the stability regions [68, 70]

$$R_{EM} := 1 + p^2 + q^2 + 2p < 1 \qquad \text{for Euler-Maruyama,} \tag{3.35}$$

$$R_{Mil} := 1 + 2p + p^2 + q^2 + q^4/2 < 1 \qquad \text{for Milstein,} \tag{3.36}$$

$$R_{SSBE} := (1+q^2)/(1-p)^2 < 1 \qquad \text{for SSBE and,} \tag{3.37}$$

$$R_{DSSBM} := (1+q^2+q^4/2)/(1-p)^2 < 1 \qquad \text{for DSSBM,} \tag{3.38}$$

where the stability region lies under the curve in Figure 9.

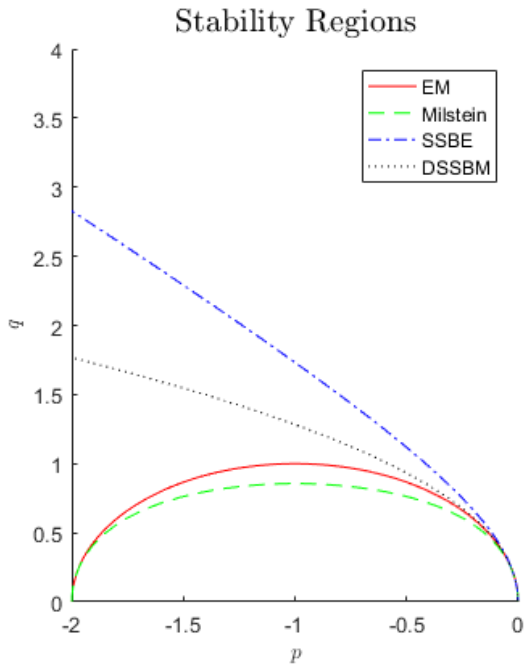


Figure 9: Stability Regions

As shown in Figure 9, the split-step methods are more stable than the fully explicit

methods. Also shown, the Euler-Maruyama method is more stable than Milstein and SSBE is more stable than DSSBM. This is an important property; however, the benefits gained from the strong convergence properties of the Milstein variants is generally more desirable, especially so with MLMC.

3.2.2 Example: Stiff System

Before using our approach on a CRN, we first demonstrate for a small example SDE from [68]. The SDE is given by

$$\begin{aligned} dy_1(t) &= c_1 y_1(t) dt, \\ dy_2(t) &= c_2 y_2(t) dt + 4 y_1(t) dW^{(1)}(t) + 0.5 dW^{(2)}(t), \end{aligned} \tag{3.39}$$

where required terms for the Milstein method (1.5) are

$$\begin{aligned} a_1(\boldsymbol{X}_t, t) &= c_1 y_1(t), \\ a_2(\boldsymbol{X}_t, t) &= c_2 y_2(t), \\ b_{2,1}(\boldsymbol{X}_t, t) &= 4 y_1(t), \\ b_{2,2}(\boldsymbol{X}_t, t) &= 0.5, \\ \frac{\partial}{\partial X_1} b_{2,1}(\boldsymbol{X}_t, t) &= \frac{\partial}{\partial y_1(t)} 4 y_1(t) = 4. \end{aligned} \tag{3.40}$$

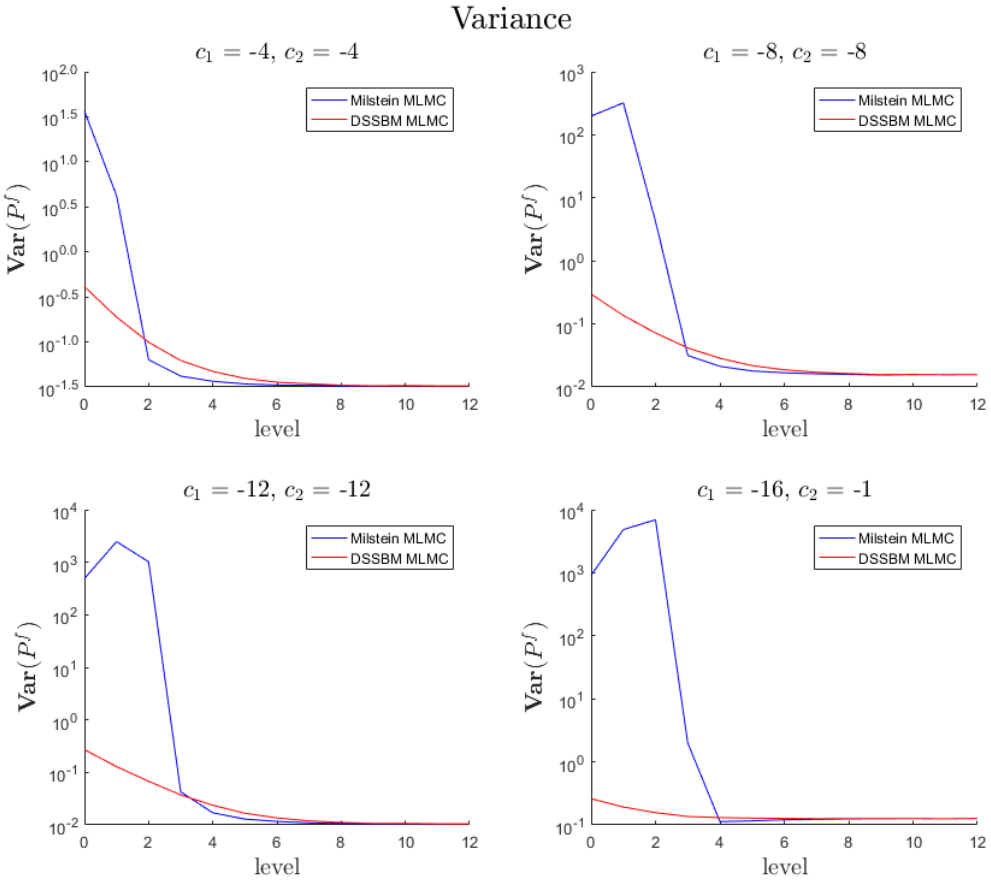
This gives the Milstein formulation for this model of

$$\begin{aligned} y_{1,n+1} &= y_{1,n} + c_1 y_{1,n} \Delta t, \\ y_{2,n+1} &= y_{2,n} + c_2 y_{2,n} \Delta t + 4 y_{1,n} \Delta W_{1,n} + 0.5 \Delta W_{2,n}, \end{aligned} \tag{3.41}$$

which is the same as the Euler-Maruyama fomulation for this simple test case.

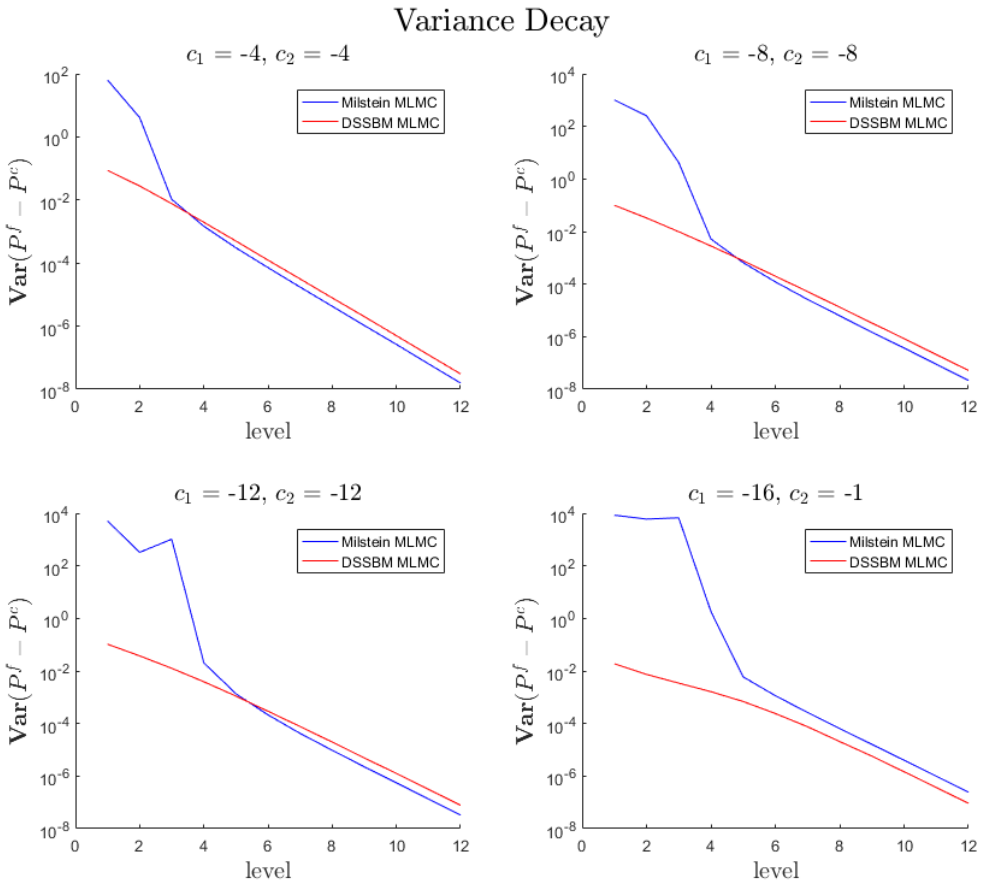
Figure 10 displays the variance of the solution for 4 different parameter sets. The Milstein method was highly unstable at the lower levels of the simulation causing the

approximation of variance to be off by multiple orders of magnitude in each case. Figure 11 shows the variance decay for each correction level of MLMC. Once again, the unstable step sizes using the Milstein method cause low accuracy corrections at the lower levels of the MLMC simulation. Around the fifth level stability is achieved by Milstein MLMC and it performs at approximately the same rate of convergence as DSSBM MLMC. The poor approximations of variance also cause an inaccurate number of samples to be required. Figure 12 shows this problem. Here the variance approximation for Milstein MLMC is so bad that the estimated number of samples required takes many orders of magnitude longer to simulate even for a relatively large error tolerance ($\varepsilon = 0.01$).



Approximate system variance for 4 parameter sets computed with 1000 sample paths per level.

Figure 10: Stiff System Variance

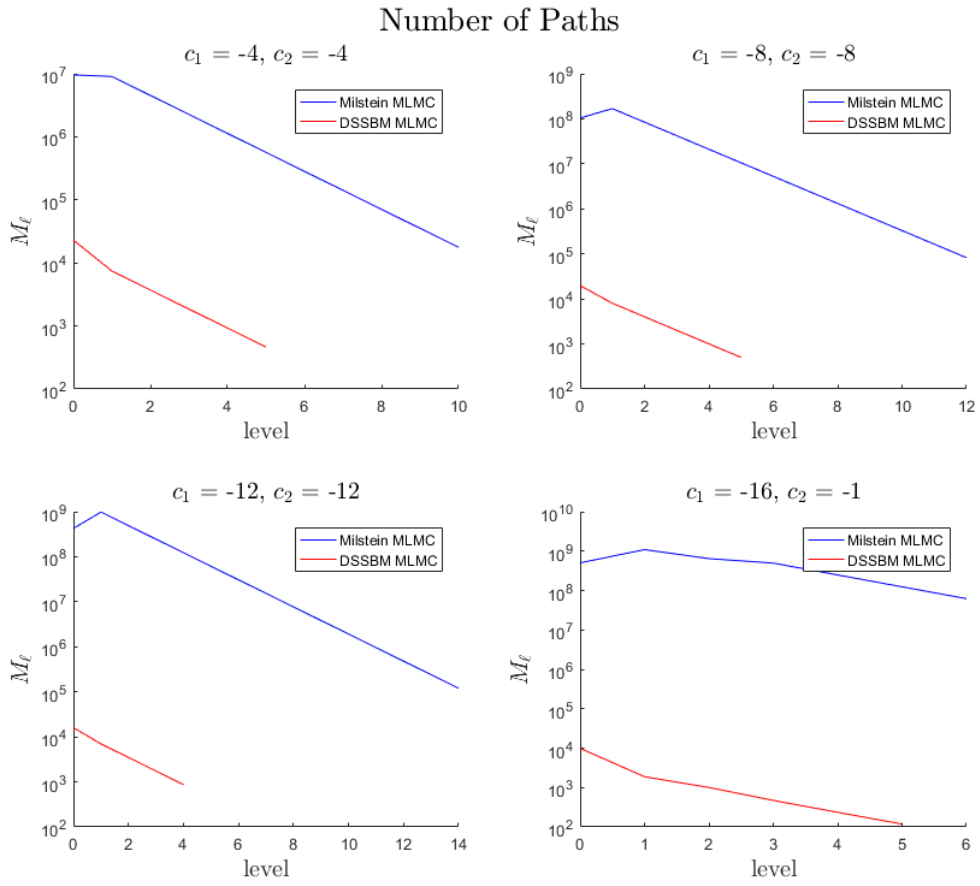


The variance decay for 4 parameter sets computed with 1000 sample paths per level.

Figure 11: Stiff System Variance Decay

There is, however, another solution to this problem. MLMC does not necessarily need to begin with $\Delta t_0 = T$. It is possible to start the simulation with a much more fine discretization. This allows for the initial level of MLMC to start with a stable step size. When starting with $\Delta t_0 = 2^{-5}T$ the results of the example problem turn out much differently. When comparing the previous results in Figures 10-12 with the results in Figures 13-Figure 15, there is a very different picture.

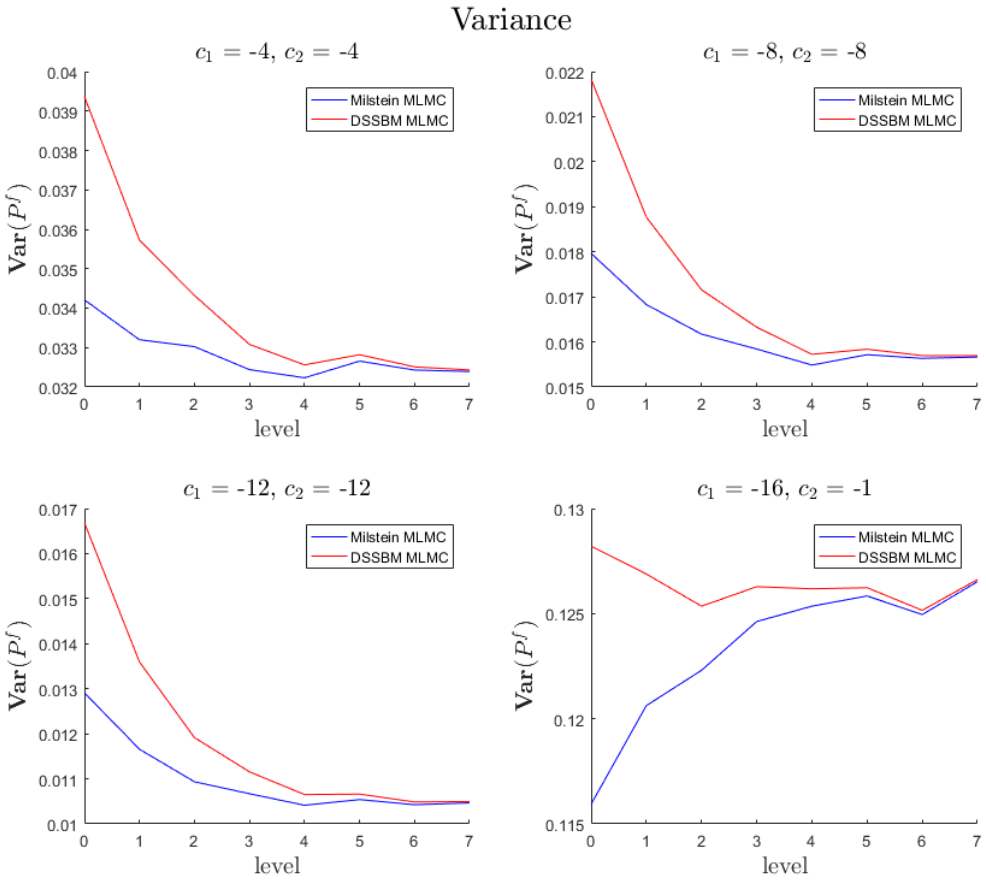
It can be concluded that generally as long as stable step sizes are used at all levels, that the MLMC algorithm effectively balances the cost. However, if the required error tolerance



$\varepsilon = 0.01$

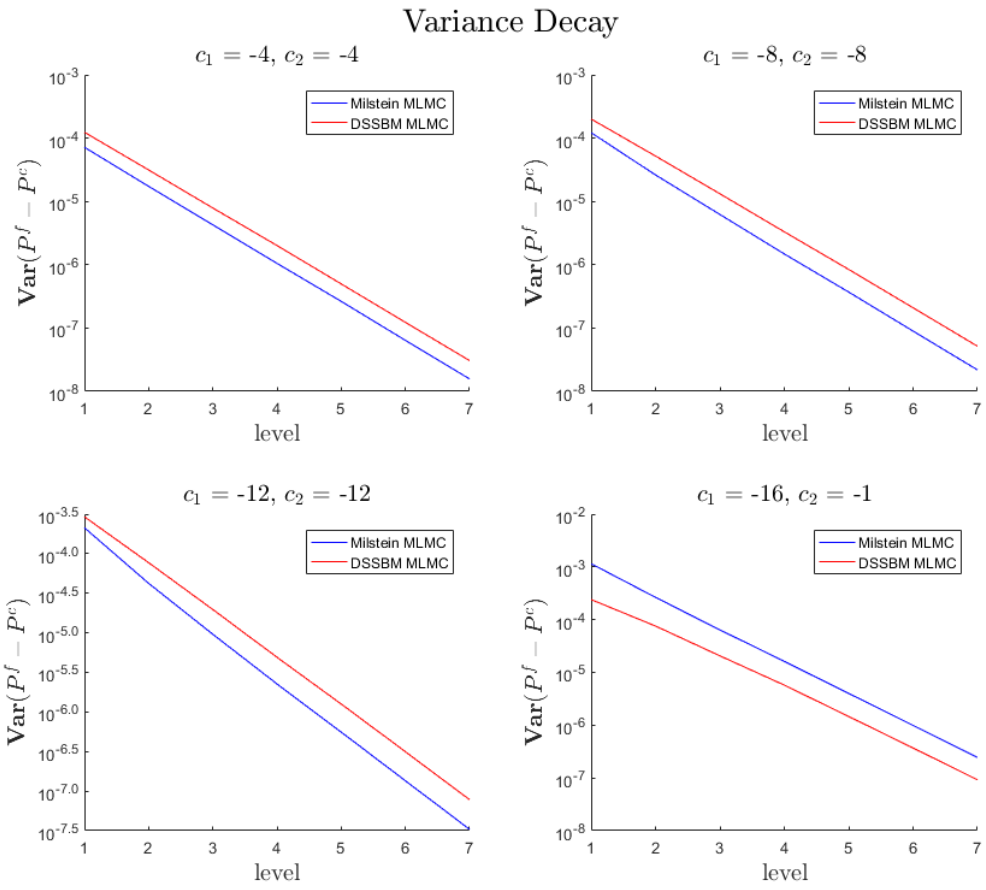
Figure 12: Stiff System Paths Per Level

is larger than the first stable step size for a method, then a partially implicit split-step method should be considered for use with MLMC.



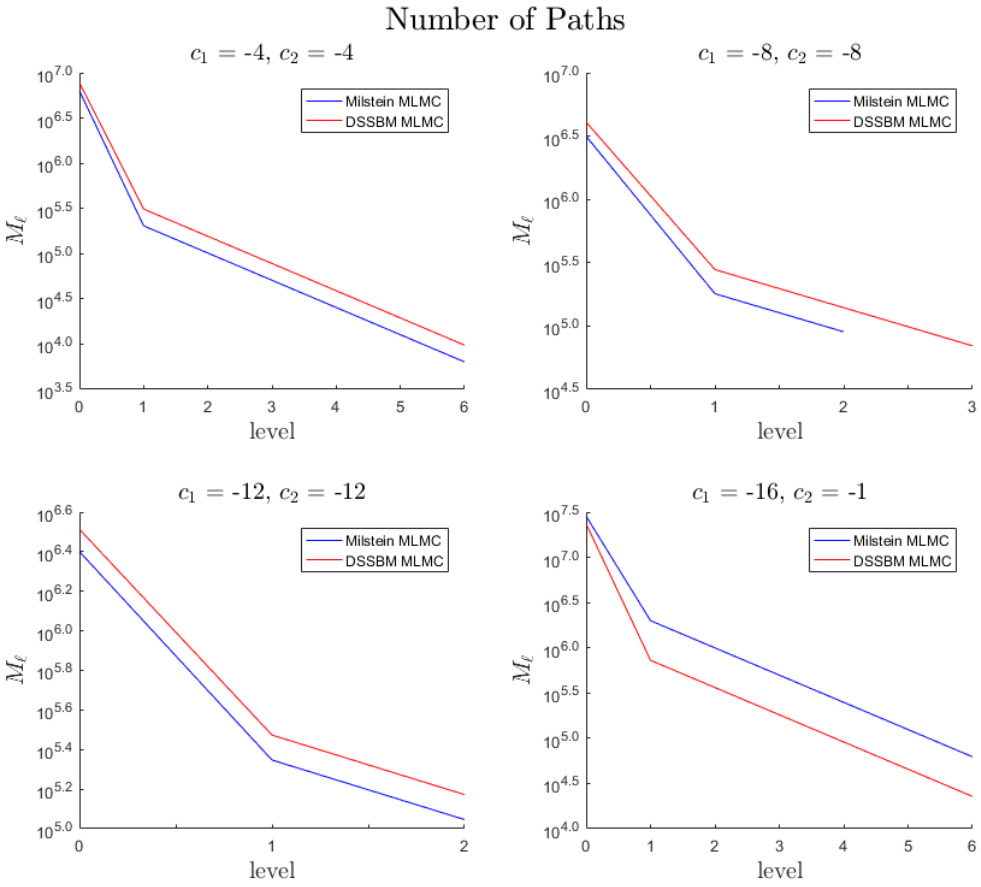
Approximate system variance for 4 parameter sets computed with 1000 sample paths per level.

Figure 13: Stiff System Variance, $\Delta t_0 = 2^{-5}T$



The variance decay for 4 parameter sets computed with 1000 sample paths per level.

Figure 14: Stiff System Variance Decay, $\Delta t_0 = 2^{-5}T$



$\varepsilon = 0.0001$

Figure 15: Stiff System Paths Per Level, $\Delta t_0 = 2^{-5}T$

3.2.3 Example: Gene Transcription CRN

In this section we apply our approach on an actual CRN. The gene transcription and translation model found in [3, 4] with 4 different types of molecules and 5 types of chemical reactions takes the form

$$G \xrightarrow{c_1} G+M,\ M \xrightarrow{c_2} M+P,\ P+P \xrightarrow{c_3} D,\ M \xrightarrow{c_4} \emptyset,\ P \xrightarrow{c_5} \emptyset. \tag{3.42}$$

It is important to notice that G is neither created nor destroyed in this reaction system; therefore, when formulating this system as an SDE we get $dG/dt = 0$. Given this property, we can simplify the problem by removing this equation from the system and incorporating the corresponding initial condition as a parameter, i.e. let the concentration of G at time $t = 0$ be G_0.

To build a system of stochastic differential equations we let

$$\boldsymbol{Y}(t) = \begin{bmatrix} Y_1(t) \\ Y_2(t) \\ Y_3(t) \end{bmatrix}, \tag{3.43}$$

be the concentrations of M, P, and D at time t respectfully. Rate constants for the example problem are give as $c_1 = 25, c_2 = 1000, c_3 = 0.001, c_4 = 0.1, c_5 = 1$. Giving the system

$$G \xrightarrow{25} G+M,\ M \xrightarrow{1000} M+P,\ P+P \xrightarrow{0.001} D,\ M \xrightarrow{0.1} \emptyset,\ P \xrightarrow{1} \emptyset. \tag{3.44}$$

The propensity functions $a_{i,t} = a_i(\boldsymbol{Y}(t))$ are given by

$$\begin{aligned} a_{1,t} &= c_1 G_0, \\ a_{2,t} &= c_2 Y_1(t), \\ a_{3,t} &= \frac{c_3}{2} Y_2(t)(Y_2(t)-1), \\ a_{4,t} &= c_4 Y_1(t), \\ a_{5,t} &= c_5 Y_2(t), \end{aligned} \tag{3.45}$$

and the state change vectors are given by

$$\boldsymbol{\nu}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ \boldsymbol{\nu}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \ \boldsymbol{\nu}_3 = \begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix}, \ \boldsymbol{\nu}_4 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \ \boldsymbol{\nu}_5 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}. \tag{3.46}$$

This produces the CLE for the Gene Transcription CRN as

$$\begin{aligned} dY_1 &= (c_1 G_0 - c_4 Y_1(t))dt + \sqrt{c_1 G_0}\, dW_1(t) - \sqrt{c_4 Y_1(t)}\, dW_4(t), \\ dY_2 &= \Big(c_2 Y_1(t) - c_3 Y_2(t)(Y_2(t)-1) - c_5 Y_2(t) \Big) dt + \sqrt{c_2 Y_1(t)}\, dW_2(t) \\ &\qquad - \sqrt{2 c_3 Y_2(t)(Y_2(t)-1)}\, dW_3(t) - \sqrt{c_5 Y_2(t)}\, dW_5(t), \\ dY_3 &= \frac{c_3}{2} Y_2(t)(Y_2(t)-1)dt + \sqrt{\frac{c_3}{2} Y_2(t)(Y_2(t)-1)}\, dW_3(t), \end{aligned} \tag{3.47}$$

For the Milstein formulation for the first equation

$$
\begin{aligned}
Y_{1,n+1} =& Y_{1,n} + (c_1 G_0 - c_4 Y_{1,n})\Delta t \\
&+ \sqrt{c_1 G_0}\Delta W_{1,n} - \sqrt{c_4 Y_{1,n}}\Delta W_{4,n} \\
&+ \frac{1}{2}\Bigg[- \sqrt{\frac{c_1 c_4 G_0}{4 Y_{1,n}}}(\Delta W_{4,n}\Delta W_{1,n}) \\
&+ \frac{c_4}{2}(\Delta W_{4,n}^2 - \Delta t) \quad \Bigg],
\end{aligned}
\tag{3.48}
$$

for second equation

$$
\begin{aligned}
Y_{2,n+1} =& Y_{2,n} + (c_2 Y_{1,n} - c_3 Y_{2,n}(Y_{2,n}-1) - c_5 Y_{2,n})\Delta t \\
&+ \sqrt{c_2 Y_{1,n}}\Delta W_{2,n} - \sqrt{2c_3 Y_{2,n}(Y_{2,n}-1)}\Delta W_{3,n} - \sqrt{c_5 Y_{2,n}}\Delta W_{5,n} \\
&+ \frac{1}{2}\Bigg[\quad \sqrt{\frac{c_1 c_2 G_0}{4 Y_{1,n}}}(\Delta W_{2,n}\Delta W_{1,n}) \\
&- \sqrt{\frac{c_2 c_4}{4}}(\Delta W_{2,n}\Delta W_{4,n}) \\
&- \sqrt{\frac{2c_2 c_3 Y_{1,n}}{Y_{2,n}(Y_{2,n}-1)}}(Y_{2,n} - \frac{1}{2})(\Delta W_{3,n}\Delta W_{2,n}) \\
&+ 2c_3(Y_{2,n} - \frac{1}{2})(\Delta W_{3,n}^2 - \Delta t) \\
&+ \sqrt{\frac{2c_3 c_5}{(Y_{2,n}-1)}}(Y_{2,n} - \frac{1}{2})(\Delta W_{3,n}\Delta W_{5,n}) \\
&- \sqrt{\frac{c_2 c_5 Y_{1,n}}{4 Y_{2,n}}}(\Delta W_{5,n}\Delta W_{2,n}) \\
&+ \sqrt{\frac{c_3 c_5 (Y_{2,n}-1)}{2}}(\Delta W_{5,n}\Delta W_{3,n}) \\
&+ \frac{c_5}{2}(\Delta W_{5,n}^2 - \Delta t) \quad \Bigg],
\end{aligned}
\tag{3.49}
$$

and for third equation

$$\begin{aligned} Y_{3,n+1} =& Y_{3,n} + \frac{c_3}{2} Y_{2,n}(Y_{2,n}-1)\Delta t \\ &+ \sqrt{\frac{c_3}{2} Y_{2,n}(Y_{2,n}-1)}\Delta W_{3,n} \end{aligned}$$

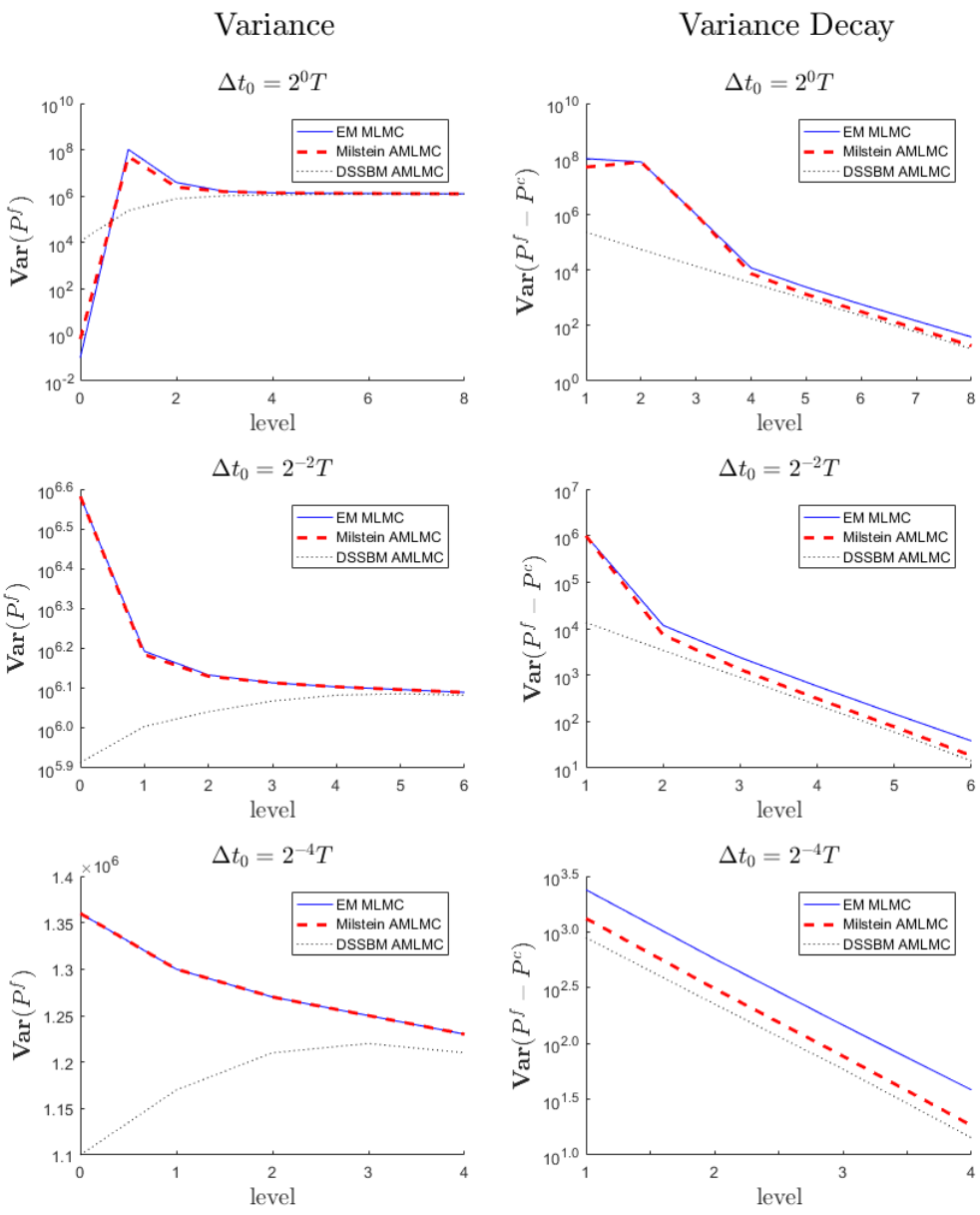
$$\begin{aligned} &+\frac{1}{2}\Bigg[\quad \sqrt{\frac{c_2 c_3 Y_{1,n}}{2Y_{2,n}(Y_{2,n}-1)}}(Y_{2,n}-\frac{1}{2})(\Delta W_{3,n}\Delta W_{2,n}) \\ &\quad - c_3(Y_{2,n}-\frac{1}{2})(\Delta W_{3,n}^2 - \Delta t) \\ &\quad - \sqrt{\frac{c_3 c_5}{2(Y_{2,n}-1)}}(Y_{2,n}-\frac{1}{2})(\Delta W_{3,n}\Delta W_{5,n}) \quad \Bigg]. \end{aligned} \tag{3.50}$$

The implicit drift terms for DSSBM

$$\begin{aligned} \overline{Y}_{1,n} =& (Y_{1,n} + c_1 G_0 \Delta t)/(1 + c_4 \Delta t), \\ \overline{Y}_{2,n} =& \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \\ \overline{Y}_{3,n} =& Y_{3,n} + \frac{c_3}{2}\overline{Y}_{2,n}(\overline{Y}_{2,n}-1)\Delta t, \end{aligned} \tag{3.51}$$

where $a = c_3\Delta t$, $b = 1 - c_3\Delta t + c_5\Delta t$, and $c = -Y_{2,n} - c_2\Delta t\overline{Y}_{1,n}$ with details for (3.47) and (3.51) included in the Appendix.

Figure 16 provides the results of this simulation, which are as expected. DSSBM provided a much more precise approximation to variance at levels with large step sizes. When smaller step sizes were used at the initial levels of MLMC the stability issue was removed. Note that Euler-Maruyama MLMC and Milstein AMLMC performed similarly in all cases where stability was a problem.



1000 sample paths per level

Figure 16: Gene Transcription Model Variance/Variance Decay

CHAPTER 4

ACCELERATED ITERATIVE SOLVERS FOR PDE SYSTEMS WITH RANDOM INPUT IN PARALLEL

In the simulation of partial differential equations (PDEs) with random input that require iterative solvers, we have presented that the use of previous samples can be used to provide better initial guesses for the solver [16]. For elliptic problems (linear or nonlinear) we use an finite element method (FEM) iterative solver that improves the initial guess during sampling. By using previous samples to build an interpolant that predicts the solution value at subsequent sample points, we can greatly reduce the number of iterations per Monte Carlo sample. We improve upon the findings by modifying these algorithms to take advantage of parallel computing environments. The changes provided in this chapter slightly reduce the benefits of these methods; however, the original algorithms were completely serial. The speedup provided by parallel computation has a much greater effect than reduced benefits.

4.1 PDEs with Random Coefficients

Let $\mathscr{L}$ be a elliptic operator (linear or nonlinear) defined on a bounded Lipschitz domain $D \subset \mathbb{R}^d$, $d = 1,2,3$, with boundary ∂D. Operator $\mathscr{L}$ has a random coefficient $a(\mathbf{x}, \omega_a)$, $\mathbf{x} \in D$, $\omega_a \in \Omega_a$, defined on the complete probability space $(\Omega_a, \mathscr{F}_a, \mathsf{P}_a)$. Here, Ω_a denotes the sample space of possible outcomes, $\mathscr{F}_a \subset 2^{\Omega_a}$ is the σ-algebra of the events, and P_a is a complete probability measure on $\mathscr{F}_a$ [8, 15].

Let the solution of this problem be the random function $u : \overline{D} \times \Omega \rightarrow \mathbb{R}^m$ for which

the following stochastic boundary-value problem is satisfied

$$\begin{cases} \mathscr{L}(a)(u) &= -f \quad \text{in } D \times \Omega, \\ \mathscr{B}(u) &= 0 \qquad \text{on } \partial D \times \Omega, \end{cases} \tag{4.1}$$

where operator $\mathscr{B}$ defines suitable boundary conditions and $f(\mathbf{x}, \omega_f)$ is a forcing random field defined analogously to $a(\mathbf{x}, \omega_a)$.

4.1.1 Karhunen-Loève Expansion

Random fields $a(\mathbf{x}, \omega_a)$ and $f(\mathbf{x}, \omega_f)$ allow finite-dimensional representation either because the problem itself can be described by a finite number of uncorrelated random variables or because they can be efficiently approximated by truncated random fields $a(\mathbf{x}, \omega_a) = a(\mathbf{x}, y_1^a, ..., y_{N_a}^a)$ and $f(\mathbf{x}, \omega_f) = f(\mathbf{x}, y_1^f, ..., y_{N_f}^f)$. We use the Karhunen-Loève expansion representation for random fields. It is assumed that the random field has a continuous covariance function $Cov(\mathbf{x}_1, \mathbf{x}_2)$ and can be represented as a series of uncorrelated random variables. This yields a finite-dimensional representation of the random field by truncation of the series

$$a(\mathbf{x}, \omega_a) \approx \mathbb{E}\left[a(\mathbf{x}, \cdot)\right] + \sum_{n=1}^{N_a} \sqrt{\lambda_n} b_n(\mathbf{x}) y_{a,n}(\omega_a), \tag{4.2}$$

where λ_n and b_n are the eigenvalues and corresponding eigenfunctions for the covariance function and $y_{a,n}(\omega_a)$ are uncorrelated random variables.

4.1.2 Multilevel Monte Carlo for PDEs with Random Input

The Monte Carlo method used to approximate the expected value of the random function $u(\mathbf{x},\omega)$ has the form

$$\mathbb{E}[u(\mathbf{x},\omega)] \approx u_M^{MC}(\mathbf{x},\mathbf{y}) = \frac{1}{M}\sum_{m=1}^{M} u(\mathbf{x},\mathbf{y}_m), \tag{4.3}$$

where independent and identically distributed random vectors $\mathbf{y}_m$ are sampled from a given probability density function and $u(\mathbf{x},\mathbf{y}_m)$ is the solution of deterministic problem corresponding to the parameter value $\mathbf{y}_m$. To account for the discretization error of from our PDE solver we shall denote the approximation by $u_{h,M}^{MC}(\mathbf{x},\mathbf{y})$ with the lower index h denotes spatial discretization of the domain D.

A major advantage of Monte Carlo is that the convergence rate does not depend on dimension of the sample space and regularity of the random function $u(\mathbf{x},\mathbf{y})$. However, it should be noted that the variance of the solution also influences the error for given M.

The multilevel Monte-Carlo method for our PDE requires a hierarchical family of nested finite element discretizations.

$$V_{h_0} \subset V_{h_1} \subset ... \subset V_{h_k} \subset ... \subset W(D), \tag{4.4}$$

where each V_{h_k} corresponds to the finite dimensional space of continuous piecewise polynomial functions defined on the triangulation τ_{h_k} of the domain D and $h_k := \max_{\tau\in\tau_{h_k}} diam(\tau)$ is the maximum mesh spacing parameter.

Let $u_{h_k}(\mathbf{y})$ denote the finite element projection of the solution $u(x,\mathbf{y})$ onto V_{h_k}. Then solution on the finest discretization level L is given as the telescoping series

$$u_{h_L}(\mathbf{y}) = u_{h_0}(\mathbf{y}) + \sum_{\ell=1}^{L}\Big(u_{h_\ell}(\mathbf{y}) - u_{h_{\ell-1}}(\mathbf{y})\Big), \tag{4.5}$$

i.e. it can be represented as a solution on the coarsest mesh plus corrections calculated on finer meshes.

By the linearity of the expectation we obtain

$$\mathbb{E}[u_{h_L}(\mathbf{y})] = \mathbb{E}[u_{h_0}(\mathbf{y})] + \sum_{\ell=1}^{L} \mathbb{E}\Big[u_{h_\ell}(\mathbf{y}) - u_{h_{\ell-1}}(\mathbf{y})\Big]. \tag{4.6}$$

Which provides the MLMC estimator

$$\mathbb{E}[u(\mathbf{x},\omega)] \approx u_{h_L,M}^{MLMC} = \sum_{\ell=0}^{L} \left[u_{h_\ell,M_\ell}^{MC} - u_{h_{\ell-1},M_{\ell-1}}^{MC}\right] = \sum_{\ell=0}^{L} \sum_{m_\ell=1}^{M_\ell} \frac{1}{M_\ell}\left[u_{h_\ell}^{m_\ell} - u_{h_{\ell-1}}^{m_\ell}\right], \tag{4.7}$$

where $u_{h_\ell}^{m_\ell} = u_{h_\ell}(\mathbf{y}_{m_\ell})$, M_ℓ is the number of random samples generated on the level ℓ and $u_{h_{-1}}^{m_{-1}} = 0$.

4.1.3 Error Analysis of Multilevel Approximation and Cost

Consider the error bounds for multilevel Monte Carlo

$$\begin{aligned}\mathbb{E}\left[\left\|\tilde{u}_{h_L,M_L}^{MLMC} - \mathbb{E}\left[u(\mathbf{x},\mathbf{y})\right]\right\|_{\tilde{W}(D)}\right] \leq \underbrace{\left\|\mathbb{E}\left[u_h(\mathbf{x},\mathbf{y}) - u(\mathbf{x},\mathbf{y})\right]\right\|_{\tilde{W}(D)}}_{\text{I := Discretization error}} \\ + \underbrace{\mathbb{E}\left[\left\|u_{h_L,M_L}^{MLMC} - \mathbb{E}\left[u_h(\mathbf{x},\mathbf{y})\right]\right\|_{\tilde{W}(D)}\right]}_{\text{II := Sampling error}} + \underbrace{\mathbb{E}\left[\left\|\tilde{u}_{h_L,M_L}^{MLMC} - u_{h_L,M_L}^{MLMC}\right\|_{\tilde{W}(D)}\right]}_{\text{III := Solver error}}\end{aligned} \tag{4.8}$$

where $\tilde{u}_{h_L,M_L}^{MLMC}$ is the actually computed value of the estimator and $\tilde{W}(D)$ is a suitable spatial function space. In the above expression, the first error component is the spatial error and it does not depend on the sampling method we use. The second error component is the sampling error and the third error component represents additional bias introduced by inexact computation of the MLMC estimator. To achieve the desired accuracy ε it is sufficient to

balance the error between all three components in the following way

$$\mathbb{E}\left[\left\|\tilde{u}_{h_L,M_L}^{MLMC} - \mathbb{E}\left[u(\mathbf{x},\mathbf{y})\right]\right\|_{\tilde{W}(D)}\right] \leq I + II + III \tag{4.9}$$
$$= \frac{\varepsilon}{3} + \frac{\varepsilon}{3} + \frac{\varepsilon}{3} = \varepsilon.$$

Now consider computational cost of the MLMC method

$$C^{MLMC} \leq \sum_{\ell=0}^{L} C_\ell J_\ell + \sum_{\ell=0}^{L} M_\ell C_\ell J_\ell \tag{4.10}$$

So to build a more effective cost effective algorithm, our goal is to minimize the above expression with respect to the constraints $I < \varepsilon/3$, $II < \varepsilon/3$ and $III < \varepsilon/3$. The work in [16] was performed to minimize the error in III, by providing a better initial guess to an iterative solver. We provide the cost analysis from [16] as the analytical basis for our approach.

4.1.4 Better Initial Guesses for Iterative Solvers

The purpose of the work in [16], was to accelerate this process by using iterative solvers (linear and nonlinear) and information from previous samples to inform the initial guesses for subsequent samples. For instance, for a linear problem we can use linear solvers, such as conjugate gradient, GMRES, or restarted GMRES (modified GMRES). For nonlinear problems we can use Newton's iterative method or fixed point iterations along with any linear solver for the system solve.

To extract an initial guess from collected samples, first mesh-free interpolation methods were examined such as Shepard's Interpolation. It was later determined that predicting qualities of unstructured data points using previous data is the one of the main objectives in the field of data mining. Many data mining techniques are analogous to those used

in mesh-free interpolation. In this context there are thoroughly investigated methods for extracting new data from old data (in our case the initial guess for our PDE solver). In an effort to keep the computational overhead of this method low, it was determined that a nearest neighbors approach would be efficient and effective.

The basic nearest neighbor algorithm simply searches gathered data for a data object that is "closest" to the new data object and predicts the new object's classification value to be the same as that of the closest value. In our context the data object is the set of random parameters to a sample along with the corresponding PDE solution as the classification value. This also means that some sort of measure would be necessary. Euclidean distance works for our purposes. The k-Nearest Neighbor methods (kNN) method simply takes the closest k neighbors to the new data object and predicts a classification value that is a weighted average of those neighbors.

4.1.5 Example: Linear Elliptic Equation

The first test problem for this approach will be the linear elliptic equation given in [59].

$$\begin{cases} -\nabla\Big(a(\mathbf{x},\omega_a)\nabla u(\mathbf{x},\omega)\Big) &= f(\mathbf{x},\omega_f) \qquad \text{in } D\times\Omega, \\ u(\mathbf{x},\omega) &= 0 \qquad \text{on } \partial D\times\Omega, \end{cases} \tag{4.11}$$

with $u(\mathbf{x},\omega)\in W(D)=H_0^1(D)$.

To approximate $a(\cdot,\omega)$ we use a Karhunen-Loève expansion

$$\log(a_N(x,\omega)-0.5)=1+Y_1(\omega)\left(\frac{\sqrt{\pi}L}{2}\right)^{1/2}+\sum_{n=2}^{N}\zeta_n\phi_n(x)Y_n(\omega) \tag{4.12}$$

where $\boldsymbol{Y}:=\{Y_n\}_{n\in N}$ is a sequence of independent, uniformly distributed random variables on $[-\sqrt{3},\sqrt{3}]$ and $\{\zeta_n\}_{n\in N}$ and $\{\phi_n\}_{n\in N}$ are the eigenvalues and eigenfunctions of the

covariance operator with kernel function

$$C[\log(a_N - 0.5)](x_1, x_2) = \exp\left\{-\frac{(x_1 - x_2)^2}{L_c^2}\right\}. \tag{4.13}$$

So

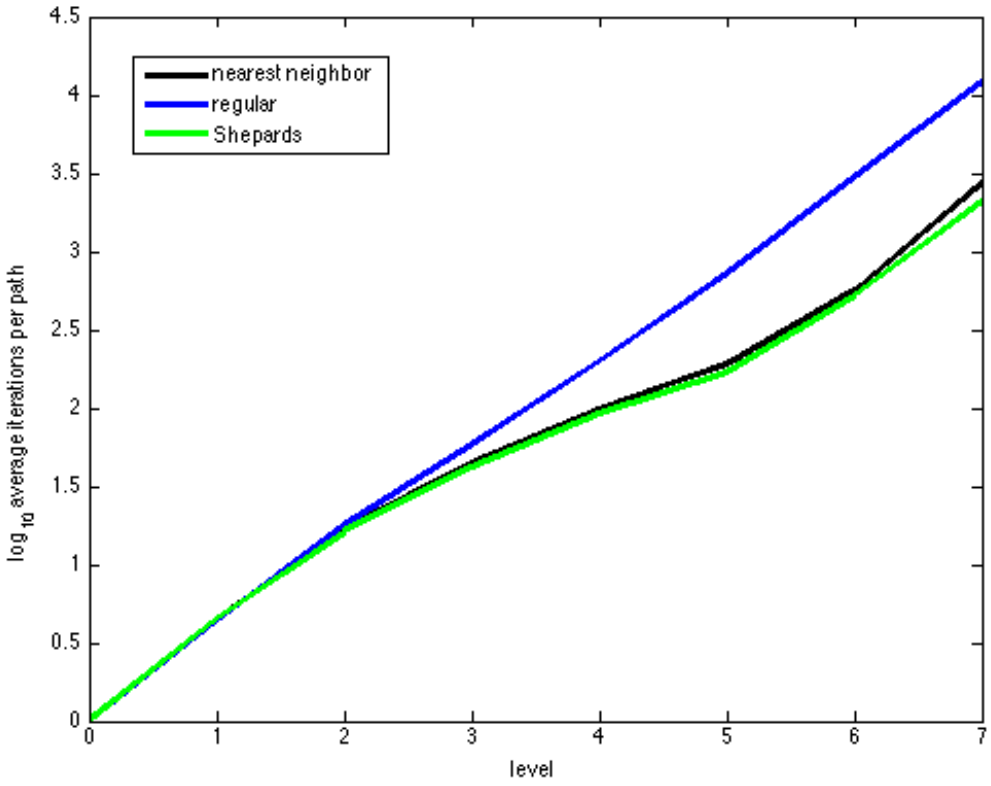
$$\zeta_n := (\sqrt{\pi}L)^{1/2} \exp\left(\frac{-(\lfloor \frac{n}{2} \rfloor \pi L)^2}{8}\right) \text{ if } n > 1, \tag{4.14}$$

and

$$\phi_n(x) := \begin{cases} \sin(\frac{\lfloor \frac{n}{2} \rfloor \pi x}{L_p}) \text{ if } n \text{ even}, \\ \cos(\frac{\lfloor \frac{n}{2} \rfloor \pi x}{L_p}) \text{ if } n \text{ odd}, \end{cases} \tag{4.15}$$

for $x \in [0, d]$, L_c is the physical correlation length for the random field a, $L_p = \max\{d, 2L_c\}$ and $L = L_c/L_p$.

In Figure 17, 'regular' indicates using zero initial guess for the iterative solver during sampling. For sample i, the nearest neighbor scheme searches all previous N dimensional points $\boldsymbol{Y}^{[j]}$, $j = 1..i-1$, and uses the solution corresponding to the closes point by Euclidean distance to point $\boldsymbol{Y}^{[i]}$. Since every $\boldsymbol{Y}$ is uniformly distributed, the search space is distributed homogeneously. On an implementation level, a K-d tree may be used to efficiently search this space repeatedly, which will always do at least as good as an exhaustive search and more likely closer to $\mathcal{O}(i \log i)$ as long as $i >> 2^N$. Shepard's Interpolation was also tested. In this approach, a mesh free interpolant is built using all previous points from all previously computed levels. Though this approach produces better initial guesses for the solver, the complexity of the interpolation is much to high to be considered further.

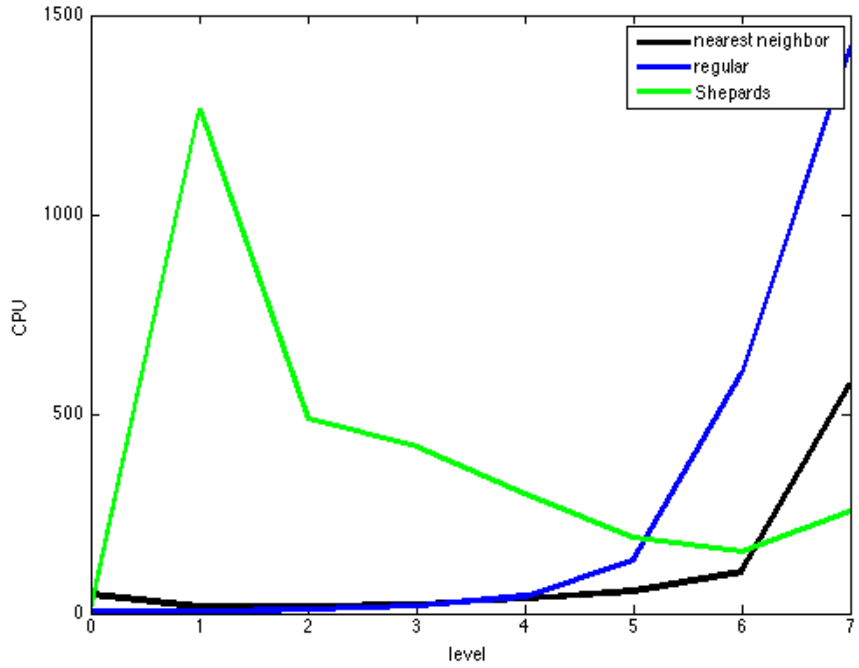


The average iterations required on each level for convergence for 3 initial guess schemes. Uses *regular* or a zero matrix, *nearest neighbor* or the solution value of the closest sample in the sample set, and *Shepards* or Shepard's interpolation [16].

Figure 17: Linear Elliptic Equation Average Iterations per Sample Path

4.1.6 Example: Steady Navier-Stokes

The second test problem used will get the steady state Navier-Stokes system. The random fields will take the same form as in (4.11).



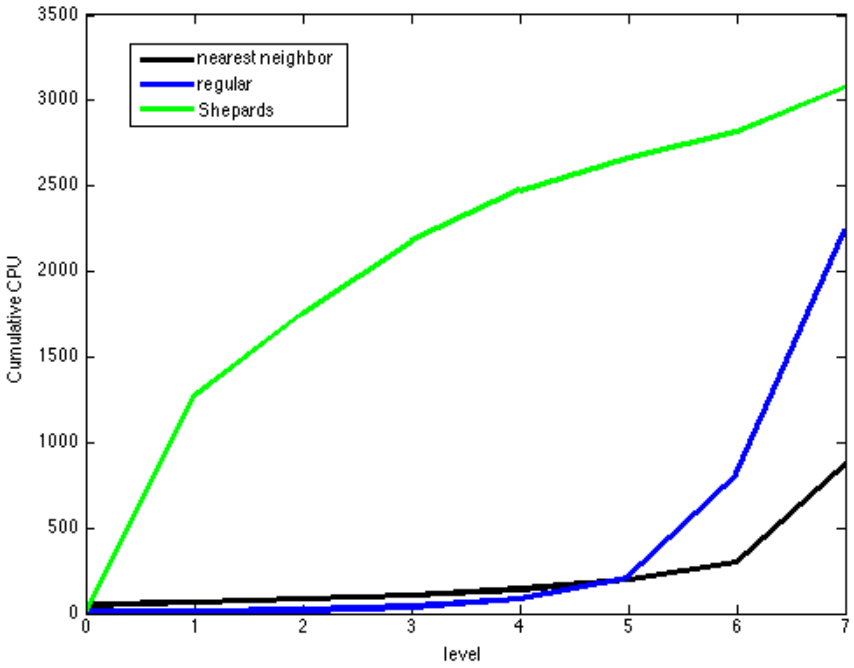
The amount of time each level took to compute for the 3 initial guess schemes. Uses *regular* or a zero matrix, *nearest neighbor* or the solution value of the closest sample in the sample set, and *Shepards* or Shepard's interpolation [16].

Figure 18: Linear Elliptic Equation Serial CPU Time

$$\begin{cases} \mu\nabla^2\boldsymbol{\upsilon} &= \rho\boldsymbol{\upsilon}\cdot\nabla\boldsymbol{\upsilon} + \nabla\pi + \rho f \qquad \text{in } D\times\Omega, \\ \nabla\cdot\boldsymbol{\upsilon} &= 0 \\ \boldsymbol{\upsilon} &= \boldsymbol{g} \qquad\qquad\qquad\qquad\quad \text{on } \partial D\times\Omega, \end{cases} \tag{4.16}$$

where $\boldsymbol{\upsilon}(\mathbf{x},\omega) \in$ homogeneous Sobolev space $D^{1,2}$ with seminorm , $p \in L^2(D)$ and $\mu = \mu(\omega_\mu)$, $\rho = \rho(\omega_\rho)$ and $f = f(\mathbf{x},\omega_f)$ are random fields representing viscosity, density and body forces respectively.

In Figure 21, we see that using more neighbors in the k-Nearest Neighbors search has diminishing returns asymptotically, as expected. In this scheme we use a equally weighed

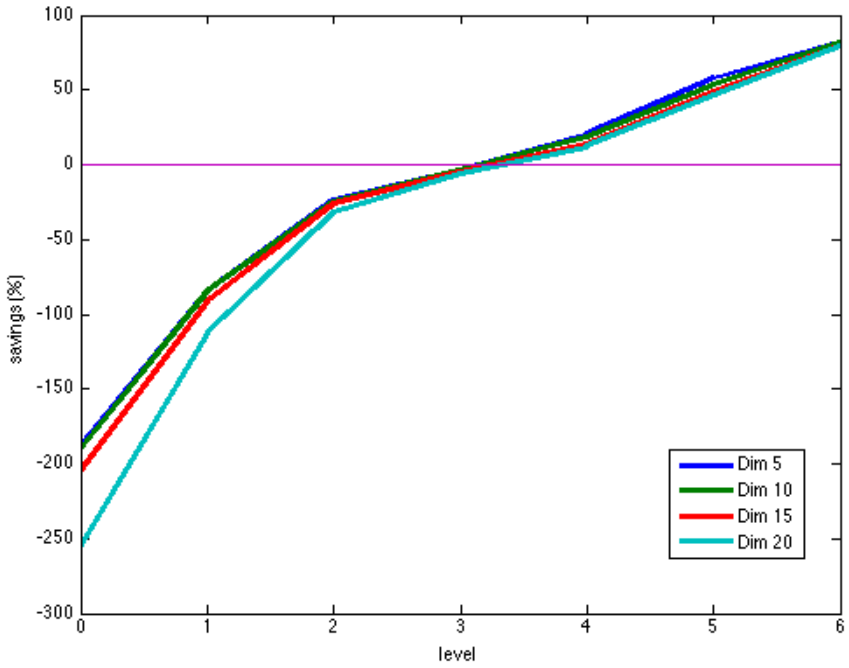


The total time used to compute the simulation [16].

Figure 19: Linear Elliptic Equation Cumulative CPU Time

average for the k neighbors. Other weighting schemes may also be tested such as inverse distances weighting. Zero in this figure represents zero initial guess for the solver. This is only used as a baseline comparison. It is typical to use the solution to the corresponding Stokes system to the Navier-Stokes system for the non-linear solver. Comparison with this traditional initial guess is used in Figures 22-23.

In Figure 22, the new method is tested against zero initial guess as well as the typical Stokes solution as the initial guess. This figure shows the average iterations per sample for a single Monte Carlo level with 500,000 samples. As expected the zero initial guess and Stokes initial guess both remain constant, as they do not improve during sampling. The nearest neighbor approach with $k = 1$ trends toward the Stokes solution line, but not enough samples were taken to see it cross. We have determined that this scheme is an



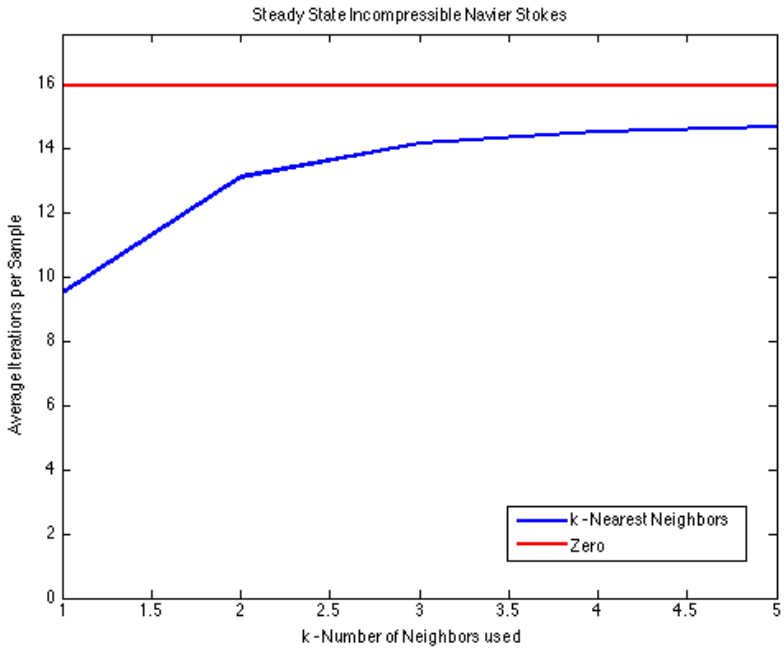
The computational savings for the nearest neighbor approach with $N = 5$, 10, 15, 20. This shows that Monte Carlo is dimension independent; however, the overhead cost of the nearest neighbor is more that it is worth until level 3 or 4 [16].

Figure 20: Linear Elliptic Equation Computation Nearest Neighbor Savings

improvement for large enough sample size, but it would be beneficial to demonstrate this with a Navier-Stokes system with a large non-linear term. In this case the Stokes solution should be such a poor initial guess that the nearest neighbor scheme should more quickly demonstrate an advantage (i.e. in fewer samples). However, we show that our parallel algorithm implementation will show savings even with only 500,000 samples.

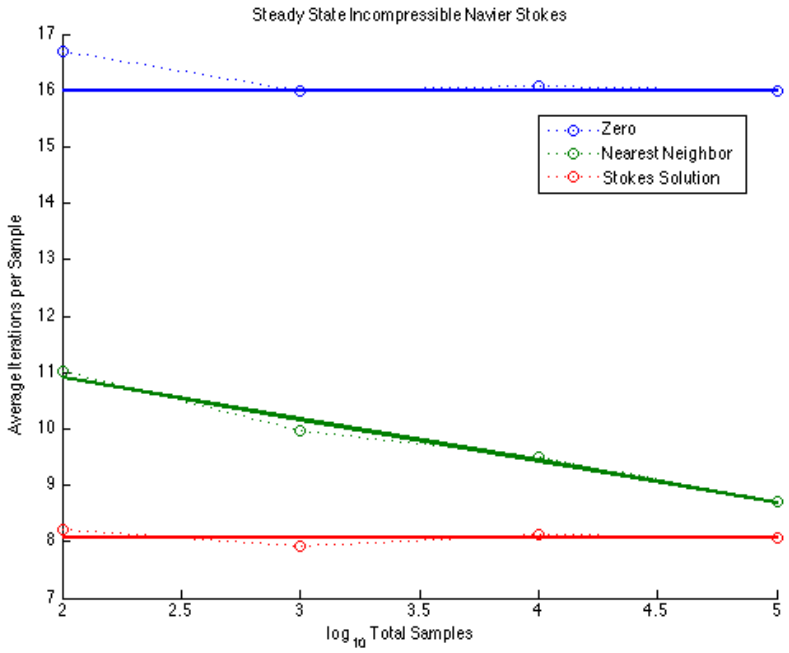
By sorting the random points using a K-dimensional tree[1] we may localize the random input into regions thereby retaining most of the advantage of the nearest neighbor approach, while benefiting from parallel computation. These results are shown in Figure 24.

[1]Note that by convention k-d trees and k nearest neighbors both use lowercase k; therefore here we will use uppercase for K-d trees.



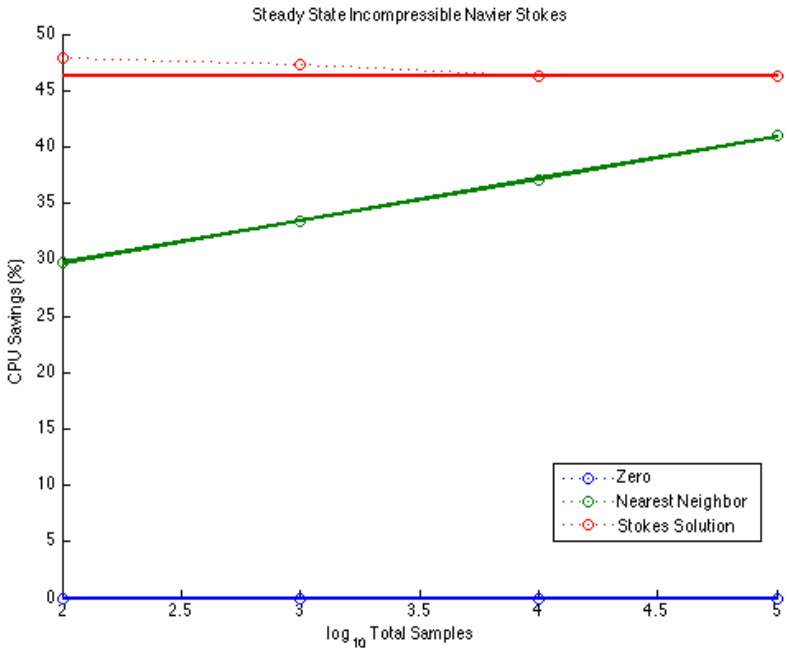
The average number of iterations per sample for k neighbors used, 'Zero' is a baseline for comparison, where zero initial guess is used [16].

Figure 21: Navier-Stokes Simulation Average Iterations for k Neighbors



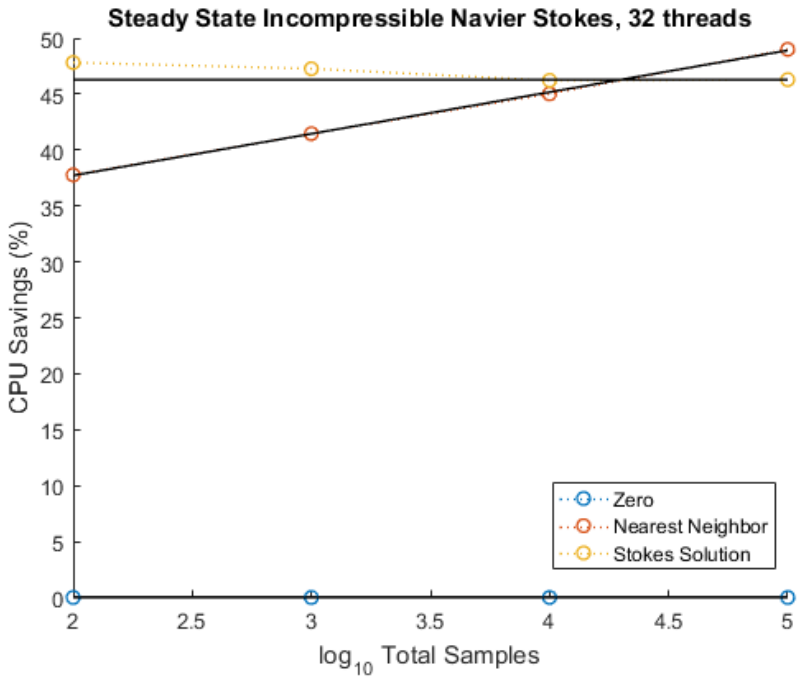
Average iterations as the sample size increases for Monte Carlo simulation using 3 schemes. Solid regression line fit to data points [16].

Figure 22: Navier-Stokes Simulation Average Iterations per Total Samples



Computational savings over using the 'zero' solution as the sample size increases for Monte Carlo simulation. Solid regression line fit to data points [16].

Figure 23: Navier-Stokes Simulation CPU Savings per Total Samples



Computational savings over using the 'zero' solution as the sample size increases for Parallel Monte Carlo simulation using regional separation of the random space. Solid regression line fit to data points [16].

Figure 24: Navier-Stokes Simulation Parallel CPU Savings per Total Samples

CHAPTER 5

CONCLUSIONS

The development of multilevel Monte Carlo has lowered the computational complexity of a wide variety of problems with more applications being developed regularly[1]. MLMC avoids the "curse of dimensionality", which gives it a major advantage over other methods when simulating large systems.

A parallel implementation of antithetic MLMC for the solution of systems of stochastic differential equations has shown to be hugely beneficial to computational efficiency. This approach is particularly useful for systems with high-dimensional, non-commutative, non-diagonal noise. Partially implicit solvers may be employed for stiff systems where stability is an issue. The use of a solver with a large step-size stability region allows for the initial level of an MLMC implementation to be taken at a much more coarse discretization than otherwise.

The improvement to iterative solvers using MLMC shown in [16] is evident. By appropriately sorting and dividing the random space as a preliminary step to sending the work to the computing nodes, much of the benefit of the method can be retained while utilizing parallel computation.

The techniques used in this work provide a computational benefit to a wide variety of problems. It is the author's belief that there are still many techniques left to explore in the further development and application of MLMC.

[1] see http://people.maths.ox.ac.uk/gilesm/mlmc_community.html for an extensive list of MLMC applications

REFERENCES

[1] Alexander, C. and Venkatramanan, A. Analytic approximations for multi-asset option pricing. *Mathematical Finance*, 22(4):667–689, 2012.

[2] Anderson, D. F. A modified next reaction method for simulating chemical systems with time dependent propensities and delays. *The Journal of Chemical Physics*, 127(21):214107, 2007.

[3] Anderson, D. F. and Higham, D. Multi-level Monte Carlo for continuous time Markov chains with applications in biochemical kinetics. *SIAM Multiscale Modeling and Simulation*, 10(1):146–179, 2012.

[4] Anderson, D. F. and Koyama, M. Weak error analysis of numerical methods for stochastic models of population processes. *Multiscale Modeling & Simulation*, 10(4):1493–1524, 2012.

[5] Anderson, D. F. and Kurtz, T. G. Continuous time Markov chain models for chemical reaction networks. In Koeppl, H., Setti, G., di Bernardo, M., and Densmore, D., editors, *Design and Analysis of Biomolecular Circuits*, pages 3–42. Springer, 2011.

[6] Bakshi, G., Cao, C., and Chen, Z. Empirical performance of alternative option pricing models. *The Journal of Finance*, 52(5):2003–2049, 1997.

[7] Barth, A. and Lang, A. Multilevel Monte Carlo method with applications to stochastic partial differential equations. *Int. Journal of Computer Mathematics*, 89(18):2479–2498, 2012.

[8] Barth, A., Schwab, C., and Zollinger, N. Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients. *Numerische Mathematik*, 119(1):123–161, 2011.

[9] Black, F. and Scholes, M. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.

[10] Cai, X. and Wang, X. Stochastic modeling and simulation of gene networks–a review of the state-of-the-art research on stochastic simulations. *IEEE Signal Processing Magazine*, 2007.

[11] Cao, Y., Gillespie, D. T., and Petzold, L. R. The slow-scale stochastic simulation algorithm. *The Journal of Chemical Physics*, 122(1):014116, 2005.

[12] Cao, Y., Gillespie, D. T., and Petzold, L. R. Efficient step size selection for the tau-leaping simulation method. *The Journal of Chemical Physics*, 124(4):044109, 2006.

[13] Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., and Skadron, K. A performance study of general-purpose applications on graphics processors using CUDA. *Journal of Parallel and Distributed Computing*, 68(10):1370–1380, 2008.

[14] Clark, J. and Cameron, R. The maximum rate of convergence of discrete approximations for stochastic differential equations. In Grigelionis, B., editor, *Stochastic Differential Systems Filtering and Control*, volume 25 of *Lecture Notes in Control and Information Sciences*, pages 162–171. Springer Berlin Heidelberg, 1980.

[15] Cliffe, K., Giles, M. B., Scheichl, R., and Teckentrup, A. L. Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients. *Computing and Visualization in Science*, 14(1):3–15, 2011.

[16] Colgin, Z., Clayton, W., Guannan, Z., Reshniak, V., and Khaliq, A. Accelerated multilevel Monte Carlo for iterative solvers. Presented at the 2014 SIAM Annual Meeting on July 8, 2014, Chicago, IL.

[17] Colgin, Z., Lay, H. A., Reshniak, V., and Khaliq, A. Q. M. On the implementation of multilevel Monte Carlo simulation of the stochastic volatility and interest rate model. submitted (2016).

[18] Dick, J., Kuo, F., Peters, G., and Sloan, I. *Monte Carlo and Quasi-Monte Carlo Methods 2012*. Springer Proceedings in Mathematics & Statistics. Springer Berlin Heidelberg, 2013.

[19] Dimits, A. M., Cohen, B. I., Caflisch, R. E., Rosin, M., and Ricketson, L. Higher-order time integration of Coulomb collisions in a plasma using Langevin equations. *Journal of Computational Physics*, 242:561–580, 2013.

[20] Ecuyer, P. and Owen, A. *Monte Carlo and Quasi-Monte Carlo Methods 2008*. Mathematics and Statistics. Springer Berlin Heidelberg, 2010.

[21] El Samad, H., Khammash, M., Petzold, L., and Gillespie, D. Stochastic modeling of gene regulatory networks. *International Journal of Robust and Nonlinear Control*, 15(15):691–711, 2005.

[22] Ethier, S. N. and Kurtz, T. G. *Markov processes: characterization and convergence*, volume 282. John Wiley & Sons, 2009.

[23] Fang, K., Hickernell, F., and Niederreiter, H. *Monte Carlo and Quasi-Monte Carlo Methods 2000: Proceedings of a Conference held at Hong Kong Baptist University, Hong Kong SAR, China, November 27 – December 1, 2000*. Springer Berlin Heidelberg, 2011.

[24] Gaines, J. G. and Lyons, T. J. Random generation of stochastic area integrals. *SIAM Journal on Applied Mathematics*, 54(4):1132–1146, 1994.

[25] Gaines, J. G. and Lyons, T. J. Variable step size control in the numerical solution of stochastic differential equations. *SIAM Journal on Applied Mathematics*, 57(5):1455–1484, 1997.

[26] Giles, M. B. Multilevel Monte Carlo path simulation. *Operations Research*, 56(3):607–617, 2008.

[27] Giles, M. B. and Szpruch, L. Antithetic multilevel Monte Carlo estimation for multi-dimensional SDEs without Lévy area simulation. *The Annals of Applied Probability*, 24(4):1585–1620, 2014.

[28] Giles, M. B. and Waterhouse, B. J. Multilevel quasi-Monte Carlo path simulation. *Advanced Financial Modelling, Radon Series on Computational and Applied Mathematics*, pages 165–181, 2009.

[29] Giles, M. Improved multilevel Monte Carlo convergence using the Milstein scheme. In Keller, A., Heinrich, S., and Niederreiter, H., editors, *Monte Carlo and Quasi-Monte Carlo methods 2006*, pages 343–358. Springer Berlin Heidelberg, 2008.

[30] Gillespie, D. T. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[31] Gillespie, D. T. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[32] Gillespie, D. T. *Markov processes: an introduction for physical scientists*. Elsevier, 1991.

[33] Gillespie, D. T. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(1):404–425, 1992.

[34] Gillespie, D. T. The chemical Langevin equation. *The Journal of Chemical Physics*, 113(1):297–306, 2000.

[35] Gillespie, D. T. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[36] Gillespie, D. T. Stochastic simulation of chemical kinetics. *Annu. Rev. Phys. Chem.*, 58:35–55, 2007.

[37] Gillespie, D. T. and Petzold, L. R. Improved leap-size selection for accelerated stochastic simulation. *The Journal of Chemical Physics*, 119(16):8229–8234, 2003.

[38] Glasserman, P. *Monte Carlo Methods in Financial Engineering*, volume 53. Springer Science & Business Media, 2003.

[39] Grzelak, L. A. and Oosterlee, C. W. On the Heston model with stochastic interest rates. *SIAM Journal on Financial Mathematics*, 2(1):255–286, 2011.

[40] Heinrich, S. Multilevel Monte Carlo methods. In *Large-scale Scientific Computing*, pages 58–67. Springer, 2001.

[41] Heston, S. L. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6(2):327–343, 1993.

[42] Higham, D. J. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM Review*, 43(3):525–546, 2001.

[43] Higham, D. J. Modeling and simulating chemical reactions. *SIAM review*, 50(2):347–368, 2008.

[44] Higham, D. J. An introduction to multilevel Monte Carlo for option valuation. *International Journal of Computer Mathematics*, 92(12):2347–2360, 2015.

[45] Higham, D. J. and Khanin, R. Chemical master versus chemical Langevin for first-order reaction networks. *Open Applied Mathematics Journal*, 2:59–79, 2008.

[46] Keller, A., Heinrich, S., and Niederreiter, H. *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Springer Berlin Heidelberg, 2007.

[47] Kloeden, P. and Platen, E. *Numerical Solution of Stochastic Differential Equations*. Applications of Mathematics. Springer-Verlag, 1992.

[48] Kloeden, P. E., Platen, E., and Wright, I. The approximation of multiple stochastic integrals. *Stochastic Analysis and Applications*, 10(4):431–441, 1992.

[49] Kurtz, T. G. The relationship between stochastic and deterministic models for chemical reactions. *The Journal of Chemical Physics*, 57(7):2976–2978, 1972.

[50] Kurtz, T. G. *Approximation of population processes*, volume 36. SIAM, 1981.

[51] Malham, S. J. and Wiese, A. Efficient almost-exact Lévy area sampling. *Statistics & Probability Letters*, 88:50–55, 2014.

[52] Medvedev, A. and Scaillet, O. Pricing American options under stochastic volatility and stochastic interest rates. *Journal of Financial Economics*, 98(1):145–159, 2010.

[53] Müller-Gronbach, T. *Strong Approximation of Systems of Stochastic Differential Equations*. PhD thesis, 2002.

[54] Niederreiter, H. *Monte Carlo and Quasi-Monte Carlo 2002: Proceedings of a Conference held at the National University of Singapore, Republic of Singapore, November 25–28, 2002*. Springer Berlin Heidelberg, 2011.

[55] Niederreiter, H. and Shiue, P. *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing: Proceedings of a conference at the University of Nevada, Las*

Vegas, Nevada, USA, June 23–25, 1994. Lecture Notes in Statistics. Springer New York, 2012.

[56] Niederreiter, H. and Spanier, J. *Monte Carlo and Quasi-Monte Carlo methods, 1998: proceedings of a conference held at the Claremont Graduate University, Claremont, California, USA, June 22-26, 1998*. Springer, 2000.

[57] Niederreiter, H. and Talay, D. *Monte Carlo and Quasi-Monte Carlo Methods 2004*. Springer Berlin Heidelberg, 2006.

[58] Niederreiter, H. *Monte Carlo and Quasi-Monte Carlo Methods 1996: Proceedings of a Conference at the University of Salzburg, Austria, July 9-12, 1996*. Lecture Notes in Statistics. Springer New York, 1998.

[59] Nobile, F., Tempone, R., and Webster, C. G. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46(5):2309–2345, 2008.

[60] Plaskota, L. and Woźniakowski, H. *Monte Carlo and Quasi-Monte Carlo Methods 2010*. Springer Proceedings in Mathematics & Statistics. Springer Berlin Heidelberg, 2012.

[61] Rathinam, M., Petzold, L. R., Cao, Y., and Gillespie, D. T. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *The Journal of Chemical Physics*, 119(24):12784–12794, 2003.

[62] Ricketson, L. Three improvements to multi-level Monte Carlo simulation of SDE systems. *arXiv preprint arXiv:1309.1922*, 2013.

[63] Rydén, T. and Wiktorsson, M. On the simulation of iterated Itô integrals. *Stochastic Processes and Their Applications*, 91(1):151–168, 2001.

[64] Saito, Y. and Mitsui, T. Stability analysis of numerical schemes for stochastic differential equations. *SIAM Journal on Numerical Analysis*, 33(6):2254–2267, 1996.

[65] Singh, S. and Raha, S. Five-stage Milstein methods for SDEs. *International Journal of Computer Mathematics*, 89(6):760–779, 2012.

[66] Sotiropoulos, V. and Kaznessis, Y. N. An adaptive time step scheme for a system of stochastic differential equations with multiple multiplicative noise: chemical Langevin equation, a proof of concept. *The Journal of Chemical Physics*, 128(1):014103, 2008.

[67] Srivastava, R., Peterson, M., and Bentley, W. Stochastic kinetic analysis of the Escherichia coli stress circuit using σ32-targeted antisense. *Biotechnology and Bioengineering*, 75(1):120–129, 2001.

[68] Voss, D. A. and Khaliq, A. Q. Split-step Adams–Moulton Milstein methods for systems of stiff stochastic differential equations. *International Journal of Computer Mathematics*, 92(5):995–1011, 2015.

[69] Wang, P. and Liu, Z. Stabilized Milstein type methods for stiff stochastic systems. *Journal of Numerical Mathematics and Stochastics*, 1(1):33–44, 2009.

[70] Wang, P. and Liu, Z. Split-step backward balanced milstein methods for stiff stochastic systems. *Applied Numerical Mathematics*, 59(6):1198–1213, 2009.

[71] Wiktorsson, M. Joint characteristic function and simultaneous simulation of iterated Itô integrals for multiple independent Brownian motions. *Annals of Applied Probability*, 11(2):470–487, 2001.

[72] Xia, Y. and Giles, M. Multilevel path simulation for jump-diffusion SDEs. In Plaskota, L. and Woniakowski, H., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2010*, pages 695–708. Springer, 2012.

APPENDIX

Appendix A: Gene Transcription CLE Details

The chemical Langevin equation with N different types of molecules and M types of chemical reactions

$$d\boldsymbol{Y}(t) = \sum_{j=1}^{M} \boldsymbol{v}_j a_{j,t} dt + \sum_{j=1}^{M} \boldsymbol{v}_j \sqrt{a_{j,t}} dW_j(t),$$

has the matrix formulation of

$$d\boldsymbol{Y}(t) = \boldsymbol{V}\boldsymbol{a}_t dt + \boldsymbol{V}\boldsymbol{b}_t^{dW}.$$

We can formulate the gene transcription CLE with functions

$$\boldsymbol{a}_t = \boldsymbol{a}(\boldsymbol{Y}(t)) = \begin{bmatrix} a_{1,t} \\ a_{2,t} \\ a_{3,t} \\ a_{4,t} \\ a_{5,t} \end{bmatrix} = \begin{bmatrix} c_1 G_0 \\ c_2 Y_1(t) \\ \frac{c_3}{2} Y_2(t)(Y_2(t)-1) \\ c_4 Y_1(t) \\ c_5 Y_2(t) \end{bmatrix},$$

$$\boldsymbol{V} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & -2 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$\boldsymbol{b}_t^{dW} = \boldsymbol{b}^{dW}(\boldsymbol{Y}(t),t) = \begin{bmatrix} b_{1,t}\,dW_1(t) \\ b_{2,t}\,dW_2(t) \\ b_{3,t}\,dW_3(t) \\ b_{4,t}\,dW_4(t) \\ b_{5,t}\,dW_5(t) \end{bmatrix} = \begin{bmatrix} \sqrt{c_1 G_0}\,dW_1(t) \\ \sqrt{c_2 Y_1(t)}\,dW_2(t) \\ \sqrt{\frac{c_3}{2} Y_2(t)(Y_2(t)-1)}\,dW_3(t) \\ \sqrt{c_4 Y_1(t)}\,dW_4(t) \\ \sqrt{c_5 Y_2(t)}\,dW_5(t) \end{bmatrix},$$

so that

$$d\boldsymbol{Y}(t) = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & -2 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_{1,t} \\ a_{2,t} \\ a_{3,t} \\ a_{4,t} \\ a_{5,t} \end{bmatrix} dt + \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & -2 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_{1,t}\,dW_1(t) \\ b_{2,t}\,dW_2(t) \\ b_{3,t}\,dW_3(t) \\ b_{4,t}\,dW_4(t) \\ b_{5,t}\,dW_5(t) \end{bmatrix},$$

and, therefore,

$$d\boldsymbol{Y}(t) = \begin{bmatrix} dY_1 \\ dY_2 \\ dY_3 \end{bmatrix} = \begin{bmatrix} (a_{1,t} - a_{4,t})dt \\ (a_{2,t} - 2a_{3,t} - a_{5,t})dt \\ a_{3,t}\,dt \end{bmatrix} + \begin{bmatrix} b_{1,t}\,dW_1(t) - b_{4,t}\,dW_4(t) \\ b_{2,t}\,dW_2(t) - 2b_{3,t}\,dW_3(t) - b_{5,t}\,dW_5(t) \\ b_{3,t}\,dW_3(t) \end{bmatrix}.$$

To approximate the CLE (3.47) with the Milstein discretization (1.5) we first determine the drift and diffusion terms. The drift terms $f_{i,t} = f_i(\boldsymbol{X}(t),t)$ and diffusion terms $g_{ij,t} = g_{i,j}(\boldsymbol{X}(t),t)$ for a stochastic system with $i = 1..N$ equations and m-dimensional uncorrelated

noise take the following form

$$dX_i = f_{i,t}dt + \sum_{j=1}^{m} g_{ij,t}dW_j,$$

Therefore, the non-zero drift and diffusion terms in (3.47) are

$$\begin{aligned}
f_{1,t} = \quad & (a_{1,t} - a_{4,t}) && = c_1 G_0 - c_4 Y_1(t), \\
f_{2,t} = \quad & (a_{2,t} - 2a_{3,t} - a_{5,t}) && = c_2 Y_1(t) - c_3 Y_2(t)(Y_2(t)-1) - c_5 Y_2(t), \\
f_{3,t} = \quad & a_{3,t} && = \tfrac{c_3}{2} Y_2(t)(Y_2(t)-1),
\end{aligned}$$

$$\begin{aligned}
g_{11,t} = \quad & b_{1,t} && = \sqrt{c_1 G_0}, \\
g_{14,t} = \quad & -b_{4,t} && = -\sqrt{c_4 Y_1(t)}, \\
g_{22,t} = \quad & b_{2,t} && = \sqrt{c_2 Y_1(t)}, \\
g_{23,t} = \quad & -2b_{3,t} && = -\sqrt{2c_3 Y_2(t)(Y_2(t)-1)}, \\
g_{25,t} = \quad & -b_{5,t} && = -\sqrt{c_5 Y_2(t)}, \\
g_{33,t} = \quad & b_{3,t} && = \sqrt{\tfrac{c_3}{2} Y_2(t)(Y_2(t)-1)},
\end{aligned}$$

and non-zero partial derivative terms

$$\frac{\partial}{\partial X_1} g_{14,t} = -\frac{\partial}{\partial Y_1} b_{4,t} = -\frac{c_4}{2\sqrt{Y_1(t)c_4}} = -\sqrt{\frac{c_4}{4Y_1(t)}},$$

$$\frac{\partial}{\partial X_1} g_{22,t} = \frac{\partial}{\partial Y_1} b_{2,t} = \frac{c_2}{2\sqrt{Y_1(t)c_2}} = \sqrt{\frac{c_2}{4Y_1(t)}},$$

$$\frac{\partial}{\partial X_2} g_{23,t} = -2\frac{\partial}{\partial Y_2} b_{3,t} = -\frac{\sqrt{2}c_3(2Y_2(t)-1)}{2\sqrt{Y_2(t)c_3(Y_2(t)-1)}} = -\sqrt{2c_3}\frac{(Y_2(t)-\frac{1}{2})}{\sqrt{Y_2(t)(Y_2(t)-1)}},$$

$$\frac{\partial}{\partial X_2} g_{25,t} = -\frac{\partial}{\partial Y_2} b_{5,t} = -\frac{c_5}{2\sqrt{Y_2(t)c_5}} = -\sqrt{\frac{c_5}{4Y_2(t)}},$$

$$\frac{\partial}{\partial X_2} g_{33,t} = \frac{\partial}{\partial Y_2} b_{3,t} = \frac{\sqrt{2}c_3(2Y_2(t)-1)}{4\sqrt{Y_2(t)c_3(Y_2(t)-1)}} = \sqrt{\frac{c_3}{2}}\frac{(Y_2(t)-\frac{1}{2})}{\sqrt{Y_2(t)(Y_2(t)-1)}},$$

Then simplify:

The first equation

$$\begin{aligned} Y_{1,n+1} =& Y_{1,n} + (c_1 G_0 - c_4 Y_{1,n})\Delta t \\ &+ \sqrt{c_1 G_0}\Delta W_{1,n} - \sqrt{c_4 Y_{1,n}}\Delta W_{4,n} \end{aligned}$$

$$\begin{aligned} &+\frac{1}{2}\Bigg[-\sqrt{\frac{c_1 c_4 G_0}{4Y_{1,n}}}(\Delta W_{4,n}\Delta W_{1,n}) \\ &\quad + \frac{c_4}{2}(\Delta W_{4,n}^2 - \Delta t) \quad \Bigg], \end{aligned}$$

the second equation

$$
\begin{aligned}
Y_{2,n+1} =& Y_{2,n} + (c_2 Y_{1,n} - c_3 Y_{2,n}(Y_{2,n}-1) - c_5 Y_{2,n})\Delta t \\
&+ \sqrt{c_2 Y_{1,n}}\Delta W_{2,n} - \sqrt{2c_3 Y_{2,n}(Y_{2,n}-1)}\Delta W_{3,n} - \sqrt{c_5 Y_{2,n}}\Delta W_{5,n}
\end{aligned}
$$

$$
\begin{aligned}
&+\frac{1}{2}\Bigg[\quad \sqrt{\frac{c_1 c_2 G_0}{4Y_{1,n}}}(\Delta W_{2,n}\Delta W_{1,n}) \\
&\quad - \sqrt{\frac{c_2 c_4}{4}}(\Delta W_{2,n}\Delta W_{4,n}) \\
&\quad - \sqrt{\frac{2c_2 c_3 Y_{1,n}}{Y_{2,n}(Y_{2,n}-1)}}(Y_{2,n}-\frac{1}{2})(\Delta W_{3,n}\Delta W_{2,n}) \\
&\quad + 2c_3(Y_{2,n}-\frac{1}{2})(\Delta W_{3,n}^2 - \Delta t) \\
&\quad + \sqrt{\frac{2c_3 c_5}{(Y_{2,n}-1)}}(Y_{2,n}-\frac{1}{2})(\Delta W_{3,n}\Delta W_{5,n}) \\
&\quad - \sqrt{\frac{c_2 c_5 Y_{1,n}}{4Y_{2,n}}}(\Delta W_{5,n}\Delta W_{2,n}) \\
&\quad + \sqrt{\frac{c_3 c_5 (Y_{2,n}-1)}{2}}(\Delta W_{5,n}\Delta W_{3,n}) \\
&\quad + \frac{c_5}{2}(\Delta W_{5,n}^2 - \Delta t) \quad \Bigg],
\end{aligned}
$$

and the third equation

$$
\begin{aligned}
Y_{3,n+1} =& Y_{3,n} + \frac{c_3}{2} Y_{2,n}(Y_{2,n}-1)\Delta t \\
&+ \sqrt{\frac{c_3}{2} Y_{2,n}(Y_{2,n}-1)}\Delta W_{3,n}
\end{aligned}
$$

$$
\begin{aligned}
&+\frac{1}{2}\Bigg[\sqrt{\frac{c_2 c_3 Y_{1,n}}{2Y_{2,n}(Y_{2,n}-1)}}(Y_{2,n}-\frac{1}{2})(\Delta W_{3,n}\Delta W_{2,n}) \\
&\quad - c_3(Y_{2,n}-\frac{1}{2})(\Delta W_{3,n}^2 - \Delta t) \\
&\quad - \sqrt{\frac{c_3 c_5}{2(Y_{2,n}-1)}}(Y_{2,n}-\frac{1}{2})(\Delta W_{3,n}\Delta W_{5,n}) \Bigg].
\end{aligned}
$$

To use DSSBM we solve drift term implicitly:

for the first equation

$$
\begin{aligned}
\overline{Y}_{1,n} =& Y_{1,n} + (c_1 G_0 - c_4 \overline{Y}_{1,n})\Delta t \\
\overline{Y}_{1,n} =& (Y_{1,n} + c_1 G_0 \Delta t)/(1 + c_4 \Delta t),
\end{aligned}
$$

for the second equation

$$\overline{Y}_{2,n} = Y_{2,n} + (c_2\overline{Y}_{1,n} - c_3\overline{Y}_{2,n}(\overline{Y}_{2,n} - 1) - c_5\overline{Y}_{2,n})\Delta t$$

$$c_3\Delta t\overline{Y}_{2,n}^2 + (1 - c_3\Delta t + c_5\Delta t)\overline{Y}_{2,n} - Y_{2,n} - c_2\Delta t\overline{Y}_{1,n} = 0$$

$$\begin{aligned} a &= c_3\Delta t \\ b &= 1 - c_3\Delta t + c_5\Delta t \\ c &= -Y_{2,n} - c_2\Delta t\overline{Y}_{1,n} \end{aligned}$$

$$\overline{Y}_{2,n} = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \overline{Y}_{2,n} = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad \text{not a solution,}$$

and for the third equation

$$\overline{Y}_{3,n} = Y_{3,n} + \frac{c_3}{2}\overline{Y}_{2,n}(\overline{Y}_{2,n} - 1)\Delta t.$$