Gaussian Process Regression Approach to Pricing Multi-Asset American Options

Christoffel Maboe Mokone

Supervisor:
Associate Professor Peter Ouwehand

A dissertation submitted to the Faculty of Commerce, University of Cape Town, in partial fulfilment of the requirements for the degree of Master of Philosophy.

January 12, 2022

MPhil in Mathematical Finance,
University of Cape Town.

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Philosophy in the University of the Cape Town. It has not been submitted before for any degree or examination in any other University.

January 12, 2022

Abstract

This dissertation explores the problem of pricing American options in high dimensions using machine learning. In particular, the Gaussian Process Regression Monte Carlo (GPR-MC) algorithm developed by Goudenège et al (2019). is explored, and its performance, i.e., its accuracy and efficiency, is benchmarked against the Least Squares Regression Method (LSM) developed by Carriere (1996) and popularised by Longstaff and Schwartz (2001). In this dissertation, American options are approximated by Bermudan options due to limited computing power. To test the performance of GPR-MC, an American geometric mean basket put option, an American arithmetic mean basket put option and an American maximum call option are priced under the multi-asset Black-Scholes and Heston models, using both GPR-MC and LSM. The algorithms are run a 100 times to obtain mean option values, 95% confidence intervals about the means, and average computational times. Numerical results show that the efficiency of GPR-MC is independent of the number of underlying assets, in contrast to the LSM method which is not. At 10 underlying assets, GPR-MC is shown to be more efficient than LSM. Moreover, GPR-MC is reasonably accurate, producing relative errors that are within reasonable bounds.

Acknowledgements

I would like to thank my supervisor, Associate Professor Peter Ouwehand, for his patience, feedback and guidance. I also like to thank my mother- Leah Mokone, my mentor- Thandiwe Rasebote, my family and friends for their continued support.

Contents

1. Introduction 1

2. Literature Review 3
2.1 American Option Pricing Algorithms 3
2.2 Machine learning and Gaussian Process Regression General Applications in Finance 4
2.3 Gaussian Process Regression Applications in American Option Pricing 6

3. Theoretical Development 8
3.1 Gaussian Process 8
3.1.1 Covariance Functions 9
3.1.2 Gaussian Process Regression 10
3.1.3 Hyperparameter Optimisation 13
3.2 An Option Pricing Application 14

4. Dynamic Programming Approach to American Option Pricing 19
4.1 Gaussian Process Regression Monte Carlo (GPR-MC) 22
4.2 Least Squares Regression Method (LSM) 28

5. Numerical Implementation 32
5.1 Numerical Results 33
5.2 Discussion 36

6. Conclusions and Further Research 44

Bibliography 46

A. Simulation 48
A.1 Multi-Asset Black-Scholes Model 48
A.2 Multi-Asset Heston Model 50

B. Quadratic Exponential (QE) Algorithm Parameters 56

List of Figures

3.1 Three sampled functions from a Gaussian prior distribution, under the Squared Exponential and Matérn Covariance functions for 1-dimensional inputs, $x \in [0, 1]$. 10
3.2 Sampled functions from a Gaussian prior distribution, under the Squared Exponential and Matérn Covariance functions for 2-dimensional input vectors; $\boldsymbol{x} \in [0, 1]^2$. 11
3.3 Three sampled functions from a Gaussian posterior distribution given 5 points (the dots), under the Squared Exponential and Matérn Covariance functions for 1-dimensional inputs, $x \in [0, 1]$. 12
3.4 Sampled functions from a Gaussian posterior distribution given 10 points (the dots), under the Squared Exponential and Matérn Covariance functions for 2-dimensional input vectors; $\boldsymbol{x} \in [0, 1]^2$. 13
3.5 A cross section of the actual (training set) and GPR-predicted option value surface at $r = 0.05,\ q = 0.0,\ T - t = 1$. 17
3.6 A cross-section of the GPR-predicted option value surface with its 95% confidence intervals, at $r = 0.05,\ q = 0.0,\ T - t = 1$. 17
3.7 [Left] A cross section of the actual (testing set) and GPR-predicted option value surface at $r = 0.05,\ q = 0.0,\ T - t = 1$. [Right] Error between actual option values (testing set) and GPR predictions. . . . 18

5.1 [Left] GPR-MC & LSM American geometric mean basket put average option prices and 95% confidence intervals for 100 runs, under the Black-Scholes & Heston models. [Right] Relative error between average LSM & GPR-MC option prices. 36
5.2 [Left] GPR-MC & LSM American arithmetic mean basket put average option prices and 95% confidence intervals for 100 runs, under the Black-Scholes & Heston models. [Right] Relative error between average LSM & GPR-MC option prices. 37
5.3 [Left] GPR-MC & LSM American maximum-call average option prices and 95% confidence intervals for 100 runs, under the Black-Scholes & Heston models. [Right] Relative error between average LSM & GPR-MC option prices. 38
5.4 American geometric mean basket put: GPR-MC & LSM Average computational times and range-interval of computational times for 100 runs. 41

5.5 American arithmetic mean basket put: GPR-MC & LSM Average computational times and range-interval of computational times for 100 runs. 42
5.6 American maximum-call: GPR-MC & LSM Average computational times and range-interval of computational times for 100 runs. 43

A.1 10-dimensional Black Scholes stock price path simulation. ($S_i(0) = 100$, $\sigma_i = 0.1$, $\eta_i = 0.0$, $\rho_{ij} = 0.2\,(i \neq j)$, $r = 0.05$, $N = 1000$ time steps) 50
A.2 10-dimensional Heston stochastic volatility and stock price path simulation. ($S_i(0) = 100$, $v_i(0) = 0.1$, $\kappa_i = 0.1$, $\theta_i = 0.15$, $\sigma_i = 0.1$, $\eta_i = 0.0$, $\rho_{ij}^{SS} = 0.1\mathbb{I}_{\{i\neq j\}} + \mathbb{I}_{\{i=j\}}$, $\rho_{ij}^{VV} = \mathbb{I}_{\{i=j\}}$, $\rho_{ij}^{SV} = 0.2\mathbb{I}_{\{i\neq j\}} + \mathbb{I}_{\{i=j\}}$, $r = 0.05$, $N = 1000$ time steps) 55

List of Tables

3.1 Training data input parameters. 15
3.2 Testing data input parameters. 16

5.1 LSM Average Option values and 95% confidence intervals for 100 runs, under the Black-Scholes model. 34
5.2 GPR-MC Average Option values and 95% confidence intervals for 100 runs, under the Black-Scholes model. 34
5.3 LSM Average Option values and 95% confidence intervals for 100 runs, under the Heston model. 34
5.4 GPR-MC Average Option values and 95% confidence intervals for 100 runs, under the Heston model. 35
5.5 LSM Average computational times and range interval of computational times for 100 runs, under the Black-Scholes model. 39
5.6 LSM Average computational times and range interval of computational times for 100 runs, under the Heston model. 39
5.7 GPR-MC Average computational times and range interval of computational times for 100 runs, under the Black-Scholes model. 39
5.8 GPR-MC Average computational times and range interval of computational times for 100 runs, under the Heston model. 40

Chapter 1

Introduction

The pricing of American options is a challenging problem in mathematical finance. In practice, the problem is notoriously difficult to tackle efficiently while also producing accurate results, because analytical solutions do not exist and consequently, one has to resort to numerical schemes. Standard numerical schemes include Monte Carlo simulation, finite difference methods and tree methods. However, such numerical schemes can be inefficient due to their high computational cost. Moreover, the curse of dimensionality makes these schemes impractical for pricing multi-asset American options. Over the years, more efficient methods have been developed. These include the Least Squares Regression Method (LSM) by Carriere (1996), which was later popularised by Longstaff and Schwartz (2001), the random tree method of Broadie and Glasserman (1997), the Hilbert Space approximation of the continuation value by Tsitsiklis and Van Roy (1999), the Malliavin Calculus based optimal stopping times dynamic programming approach of Abbas-Turki and Lapeyre (2011) and the Stochastic Grid Bundling Method (SGBM) of Jain and Oosterlee (2012).

In recent times, researchers have explored the applicability of machine learning for this problem. In particular, Goudenège *et al.* (2019) have developed a method which combines Monte Carlo simulation with Gaussian Process Regression (GPR), which they termed Gaussian Process Regression Monte Carlo (GPR-MC), to price multi-asset American options. GPR-MC is a dynamic programming approach which approximates an American option by partitioning the time axis into discrete exercise times. At each exercise time, a GPR model is trained. In the next exercise time, this model is then used to approximate the continuation value. The efficiency of this method was shown to be independent of the dimensionality of the pricing problem.

This dissertation explores the use of GPR-MC to American option pricing problems in high dimensions. The performance of GPR-MC, i.e., the accuracy and efficiency, is benchmarked against the Least Squares Regression method (LSM)

of Longstaff and Schwartz (2001). The pricing problem is considered under the multi-asset Black-Scholes model and the multi-asset Heston model. The multi-asset American options priced in this dissertation are the geometric mean basket put option, the arithmetic mean basket put option, and the maximum call option.

This dissertation is structured as follows: Chapter 2 provides a review of the literature on American/ Bermudan option pricing algorithms. Next, literature on the general uses of Gaussian Process Regression (GPR) in mathematical finance, is reviewed. Lastly, literature on the applications of GPR in the dynamic programming approach to American/ Bermudan option pricing is explored. Chapter 3 provides the theoretical background of this dissertation, in particular, a theoretical development of GPR. Then, a simple derivative pricing application of GPR is presented. Chapter 4 presents the pricing algorithms (GPR-MC & LSM) from a theoretical perspective. Moreover, sketch algorithms for the numerical implementation of said pricing algorithms are presented. Chapter 5 reports and discusses the numerical results of this dissertation. Chapter 6 concludes the dissertation.

Chapter 2

Literature Review

In this chapter, various literature on American option pricing and the applications of Gaussian Process Regression (GPR) in finance, is reviewed. To this effect, this chapter is structured as follows: firstly, literature on important American option pricing algorithms is reviewed. Secondly, GPR is introduced and literature on its general applications in finance is considered. Lastly, literature on the applications of GPR in American option pricing in particular, is reviewed.

2.1 American Option Pricing Algorithms

On the pricing of American options many methods have been proposed in the literature. These include recombinant tree methods, regression based methods which compute the continuation value using a truncated L^2-basis, Malliavin calculus-based approaches for computing conditional expectations, duality methods for pricing Bermudan options and machine learning techniques (Goudenège *et al.*, 2019).

Abbas-Turki and Lapeyre (2011) approach the problem of pricing American options by using a blend of Monte Carlo simulation and Malliavin calculus. In particular, Abbas-Turki and Lapeyre (2011) consider stopping times on a finite set. Then, they express the dynamic programming algorithm in terms of optimal stopping times at finite exercise times. Then, Malliavin calculus and Monte Carlo simulation are used to approximate the continuation value. Broadie and Glasserman (1997) developed an algorithm which uses a random tree with equal branches at each node. At each exercise date, two estimates are computed-one that is biased above (i.e. overestimates the option value) and one that is biased below (i.e. underestimates the true option value). Broadie and Glasserman (1997) show that the estimates are consistent i.e. they converge to the true option value. Jain and Oosterlee (2012)'s stochastic grid method is a blend between dynamic programming and Monte Carlo simulation, and it is used to price multi-asset Bermudan options. To price using this method, first, the asset prices are generated using some discretisation scheme. Us-

ing the Tower property, the continuation value is then expressed as a conditional expectation of a conditional expectation. The inner conditional expectation is the best estimate of the previous step's option value projected on a lower dimensional space (defined by the payoff function). This conditional expectation is then regressed (in L^2) in this lower dimensional space. Finally, the exercise strategy of this algorithm is applied on a set of new generated paths to determine the earliest exercise time. Then, the expectation of the discounted payoff from this earliest exercise time, is the lower bound of the option value. Longstaff and Schwartz (2001) considered American derivatives with square integrable payoff functions. Then, they developed a dynamic programming algorithm which approximates the continuation function with a truncated L^2 basis, i.e., the continuation value is expressed as a finite linear combination of basis functions, the coefficients of which are estimated using ordinary least squares optimization. Tsitsiklis and Van Roy (1999) formulate the problem of valuing a Bermudan-type derivative based on the price of a single stock as a an optimal stopping problem. This problem is then approximated in a Hilbert space through the use of judiciously selected basis functions.

2.2 Machine learning and Gaussian Process Regression General Applications in Finance

Gaussian Process Regression (GPR), also termed *kriging*, is a type of supervised machine learning technique. More technically, it is a non-parametric Bayesian approach to problems in regression. To this effect, GPR learns the function underlying a training set (of inputs $\boldsymbol{x}$ and output $\boldsymbol{y}$), by imposing a Gaussian prior distribution on the underlying function. Then, using Bayesian analysis, predictions are made via a posterior distribution (which is also Gaussian) given new inputs, $\boldsymbol{x}^*$, and the training set. The reader is encouraged to refer to section 3.1 for more details on GPR.

GPR has found applications in a number of different areas in Mathematical Finance. These include pricing of derivatives, Credit Valuation Adjustment (cVA) computations, local volatility calibration, the pricing of insurance products and the fitting of yield curves. De Spiegeleer *et al.* (2018) undertook the pricing of a plain vanilla call option. They considered the problem under both the Heston and the Variance Gamma models. They then contrasted the performance of GPR, i.e., accuracy and speed, to the Fast Fourier Transform (FFT) method. GPR was found to be more efficient than the FFT-method and the accuracy of the two methods was similar. Next, the researchers undertook pricing an American put using GPR, where the underlying evolves according to the binomial model. The GPR was found to be

accurate. To demonstrate the broad applicability of GPR, the researchers also computed the gamma of a cliquet option under the Heston model and contrasted GPR with both an interpolation scheme and a regression model with polynomial basis functions; GPR was more accurate. Further GPR applications to derivative pricing were done by Crépey and Dixon (2019). In their paper, GPR was used to price a European call and put options under the Heston model, and to calculate the the delta and vega of these options. Herfurth (2020) implemented GPR to the classical problem of derivative pricing, in particular, in his dissertation, GPR was implemented to price European vanilla options, American options and Barriers option under both the Black-Scholes model and Heston models. The perfomance of GPR was compared to classical methods such as Monte Carlo simulation, binomial tree model and Fast Fourier Transform method (FFT). In all cases, GPR was found to be more computationally efficient and the errors it produced were within reasonable bounds.

Crépey and Dixon (2019) considered the problem of modelling counterparty risk, in particular, the computation of the Credit Valuation Adjustment (cVA) of a portfolio. This is notorious for being computationally onerous as all trades with all counterparties have to be simultaneously considered (Crépey and Dixon, 2019). To this effect, Crépey and Dixon (2019) proposed a Multi-Gaussian Process Regression (MGPR) approach. MGPR is similar to GPR except it has multiple outputs instead of a single output. As a numerical illustration of MGPR, they considered a portfolio with two long positions in a call option and a short position in a put, where the underlying is governed by the Black-Scholes model. To illustrate broad applicability of MGPR, they computed the Value-at-Risk (VaR) of a one year incremental cVA, and they also computed the cVA of a portfolio of positions in 20 interest rate swaps where the short rate dynamics were that of a multi-factor Hull-White model.

On the problem of calibrating the local volatility for the purpose of pricing options, the common approach is to impose a parametric function for the local volatility and then calibrate its parameters using either Maximum Likelihood Estimation (MLE) or Ordinary Least Squares (OLS) (Tegnér *et al.*, 2017). Tegnér *et al.* (2017) instead proposed a GPR method of calibrating the local volatility for the S&P market data. To this effect, the researchers imposed a Gaussian process on the exponential of local volatility; with mean zero and covariance matrix determined by the Squared Exponential covariance function. To assess the accuracy of the model, the samples where transformed to sample over call prices and implied volatility. The market call prices and sample call prices were similar. The market determined implied volatilities were very close to the sampled implied volatilities and errors were within reasonable bounds, except at shorter maturities. This is due to the fact that

local volatility models tend to produce inconsistent prices when the maturities are shorter (Tegnér *et al.*, 2017).

In insurance, Goudenège *et al.* (2020a) considered the problem of computing the price and sensitivities of a Guaranteed Minimum Withdrawal Benefit (GMWB) Variable Annuity (VA), under the stochastic volatility and stochastic interest rate Heston Hull-White model. In particular, the researchers considered the computation of the no-arbitrage fee of a GMWB and its Delta, which is important for hedging this product. GPR's performance was compared to the Hybrid-Tree PDE (HPDE) method. HPDE is a backward algorithm where the share price is diffused using a finite difference PDE whereas the other sources of noise i.e. stochastic interest rate and stochastic volatility, are diffused using a tree method (Goudenège *et al.*, 2020a) such as the trinomial tree. The GPR prediction errors were quite small and the computation speed was significantly boosted by up to five orders of magnitude.

In their paper, Gonzalvez *et al.* (2019) considered the problem of yield curve fitting. In particular, they considered fitting the US yield curve. This task is usually undertaken by using a parametric model; the Nelson-Siegel being the most popular (Gonzalvez *et al.*, 2019). GPR was used in conjunction with ARX (Autoregressive with Exogenous input) model; they termed this model GP-ARX. In ARX modelling, the time series and some of its previous values and some exogenous factors, are assumed to be related through a non-linear function (Gonzalvez *et al.*, 2019). GP-ARX assumes that this non-linear function is a Gaussian process. To illustrate the performance of GP-ARX, the 2Y and 10Y spot rates were forecast using a single lag for GP-ARX and the 3-month (and its first lag) as exogenous factors. GP-ARX produced a lower Root mean square error compared to ARX for the forecasts, although the differences were not significant.

2.3 Gaussian Process Regression Applications in American Option Pricing

The incorporation of GPR in the dynamic programming approach to pricing American/ Bermudan options was first explored by Ludkovski (2018), as far as the research of this dissertation yields. Ludkovski (2018), constructed a *batched experimental design* for realisations of the market state variable, e.g. asset prices, stochastic volatilities etc. In a *batched experimental design*, independent realisations are generated from a given initial condition. In other words, at time n, P independent realisations are generated. For each of these P, a further M independent realisations (a "batch") are generated, conditional on each of the P realisations as the initial conditions. This approach was later applied by Goudenège *et al.* (2019) when

developing Gaussian Process Regression Monte Carlo (GPR-MC).

More recently, Goudenège *et al.* (2019) and Goudenège *et al.* (2020b) developed GPR based algorithms for pricing multi-asset American options to tackle the curse of dimensionality problem associated with American options in high dimensions. To tackle this problem, Goudenège *et al.* (2019) and Goudenège *et al.* (2020b) proposed three methods based on GPR, which they termed the GPR-Tree, GPR Monte Carlo Control Variate (GPR-MC-CV) and GPR Exact Integration (GPR-EI). GPR-Tree and GPR-MC-CV are similar to GPR Monte Carlo (GPR-MC). GPR-MC works as follows: time is partitioned into N exercise dates or time steps, hence approximating an American option by a Bermudan option (Goudenège *et al.*, 2020b). The algorithm proceeds backwards. The continuation value at time n is computed using GPR as a prediction over time $n+1$ asset prices which are simulated conditional on time n asset prices; training consists of the time $n+1$ asset prices and the continuation values, with the continuation values at N being the payoff. GPR-Tree is similar to GPR-MC, except the dynamics of the underlying at each step are modelled using a binomial tree (Goudenège *et al.*, 2020b). GPR-MC-CV introduces a control variate to GPR-MC. In particular, Goudenège *et al.* (2020b) used a European option price as a control variate at each time step. The GPR-EI utilises a semi-analytical expression to price (Goudenège *et al.*, 2019). The continuation value is approximated using the mean posterior GPR function under the Squared Exponential covariance function. The continuation value is then computed as the discounted value of the posterior mean function over a multidimensional probability density (Goudenège *et al.*, 2019). Goudenège *et al.* (2019) used the GPR-Tree, GPR-MC, GPR-MC-CV and GPR-EI to price a geometric basket put option, an arithmetic basket put option and a call on a maximum option. Goudenège *et al.* (2020b) extended the methods GPR-Tree and GPR-EI, to a non-Markovian model, the rough Bergomi model.

Chapter 3

Theoretical Development

This chapter presents the theoretical background of this dissertation. In particular, a full mathematical treatment of Gaussian Process Regression (GPR) is given. However, the mathematical treatment is only partial because a full Bayesian analysis is not necessary for the purpose of this dissertation. The reader is encouraged to refer to Rasmussen and Williams (2006) for a full Bayesian treatment. In section 4.1, this GPR machinery is then used to develop a dynamic programming algorithm for pricing American/ Bermudan options in high dimensions. Finally, this chapter concludes by presenting a simple derivative pricing application of GPR.

3.1 Gaussian Process

This section presents the theoretical background of GPR. GPR is a type of supervised machine learning technique. More technically, it is a non-parametric Bayesian regression approach to problems in regression. It is non-parametric in the sense that the underlying function is assumed to have some distributional structure without specifying a parametric form, which makes GPR more flexible. Although GPR is a machine learning technique, it is at its core a non-parametric Bayesian regression approach. Hence, this text should be accessible to a reader with a background in Bayesian regression and Gaussian distribution theory. This section is structured as follows: first, Gaussian Processes (GPs) are presented. GPs are important in the development of GPR, because GPR imposes, on the underlying function, a Gaussian Process as a prior distribution. Next, covariance functions are presented. They determine a distribution on a space of functions. Next, Gaussian distribution theory is used to perform Bayesian analysis and a posterior distribution is derived. Covariance functions have parameters (termed hyper-parameters) that have to be estimated. This section concludes by giving a partial Bayesian analysis treatment of selecting these hyper-parameters.

The following discussion of the theory of Gaussian Process Regression or GPR,

follows very closely that of sections 2.2, 4.1, 4.2, 5.2 and 5.4 in Rasmussen and Williams (2006). A *Gaussian process is any collection of random variables such that every finite subset has a multivariate Gaussian distribution*. In this dissertation, the following definition is used.

Definition 3.1 (*Gaussian Process (GP)*)**.** $\{f(\boldsymbol{x}) : \boldsymbol{x} \in \boldsymbol{X}\}$ for $\boldsymbol{X} \subseteq \mathbb{R}^N$ is a Gaussian Process (GP) if for any $d \in \mathbb{N}$ and $\boldsymbol{x}_i \in \boldsymbol{X}$ for $1 \leq i \leq d$, the random vector $(f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_d))^\top$ is jointly multivariate Gaussian. Thus, the Gaussian process can be completely characterized by its mean and covariance functions (Rasmussen and Williams, 2006).

The mean function and covariance functions are defined as follows:

$$
\begin{aligned}
m(\boldsymbol{x}) &= \mathbb{E}\left[f(\boldsymbol{x})\right], \\
k(\boldsymbol{x}, \boldsymbol{x}') &= \mathbb{E}\left[\left(f(\boldsymbol{x}) - m(\boldsymbol{x})\right)\left(f(\boldsymbol{x}') - m(\boldsymbol{x}')\right)\right], \\
&\text{hence,} \\
&f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')).
\end{aligned}
$$

For ease of notation, it is convenient to let the mean function to be zero i.e. $m(\boldsymbol{x}) = 0$ (Rasmussen and Williams, 2006). Then,

$$f(\boldsymbol{x}) \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}')).$$

3.1.1 Covariance Functions

A *kernel* is any function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A kernel is symmetric if $k(\boldsymbol{x}, \boldsymbol{x}^{'}) = k(\boldsymbol{x}^{'}, \boldsymbol{x}),\ \forall \boldsymbol{x}, \boldsymbol{x}^{'} \in \mathcal{X}$.

Definition 3.2 (*Gram Matrix*)**.** Let $\boldsymbol{\mathcal{X}} = \{\boldsymbol{x}_i : i = 1, ..., n\}$ be a set of points. A *Gram matrix* is a matrix K, whose entries are $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x_j})$, for some kernel k.

An $n \times n$ real matrix K is said to be positive semi-definite if $\forall \boldsymbol{v} \in \mathbb{R}^n$, $\boldsymbol{v}^\top K \boldsymbol{v} \geq 0$. The covariance matrix of a Gaussian distribution is symmetric positive semi-definite. This implies the following definition:

Definition 3.3 (*Covariance function*)**.** A Covariance function is any function such that for any vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$, the Gram matrix is symmetric positive semi-definite.

In the literature, the Squared Exponential (SE) and the Matérn covariance functions are popular. The Squared Exponential covariance function (also called a Radial Basis Function (RBF)) has the following functional form

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_f^2 \cdot \exp\Big(- \frac{||\boldsymbol{x} - \boldsymbol{x'}||_2^2}{2l^2} \Big),$$

where l is the horizontal length-scale, also termed the characteristic length scale, and σ_f^2 is the signal variance. The Squared Exponential kernel is infinitely differentiable and thus the Gaussian Process is very smooth in the mean squared sense (Rasmussen and Williams, 2006). This strong smoothness characteristic of the SE kernel may be unsuitable for many physical phenomena and Matérn class covariance functions are recommended (Rasmussen and Williams, 2006).

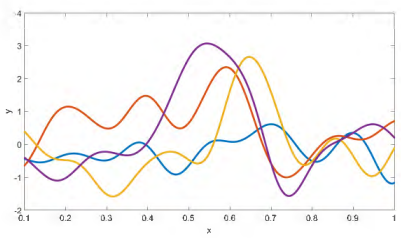
The Matérn class covariance functions have the following form

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{l} ||\boldsymbol{x} - \boldsymbol{x}'||_2 \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{l} ||\boldsymbol{x} - \boldsymbol{x}'||_2 \right),$$

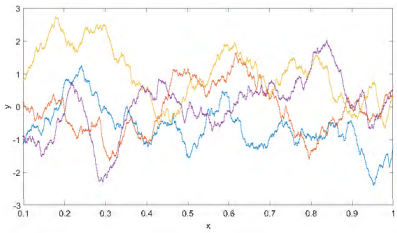
where $\nu, l > 0$ and K_ν is a modified Bessel function. The function f whose covariance function is the Matérn kernel is mean squared differentiable $k-$times if and only if $\nu > k$ (Rasmussen and Williams, 2006). The covariance function is simplified if $\nu = p + \frac{1}{2}$ where p is a positive integer, to become

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \exp\left(-\frac{\sqrt{2\nu}}{l} ||\boldsymbol{x} - \boldsymbol{x}'||_2 \right) \frac{\Gamma(p+1)}{\Gamma(2p+1)} \sum_{i=0}^{p} \frac{(p+i)!}{i!(p-i)!} \left(\frac{\sqrt{8\nu}}{l} ||\boldsymbol{x} - \boldsymbol{x}'||_2 \right)^{p-i},$$

(Rasmussen and Williams, 2006). Figures 3.1 and 3.2 show functions that are sampled from the prior distributions determined by the above-mentioned covariance functions.



(a) Squared Exponential ($l = 0.1,\ \sigma_f = 1$)

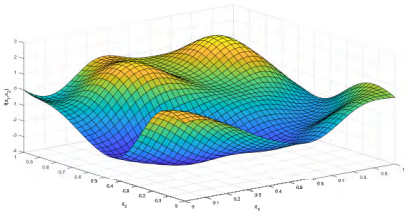


(b) Matern ($l = 0.1,\ \nu = 0.8$)

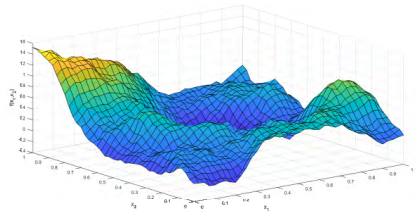
Fig. 3.1: Three sampled functions from a Gaussian prior distribution, under the Squared Exponential and Matérn Covariance functions for 1-dimensional inputs, $x \in [0, 1]$.

3.1.2 Gaussian Process Regression

Suppose a training set $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y}) = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ is given, such that $\boldsymbol{x} \in \boldsymbol{X} \subset \mathbb{R}^d$ and $\boldsymbol{y} \subset \mathbb{R}$. The objective of Gaussian Process Regression is to predict $y^* = f(\boldsymbol{x}^*)$



(a) Squared Exponential ($l = 0.25,\ \sigma_f = 1$)



(b) Matern ($l = 1,\ \nu = 1$)

Fig. 3.2: Sampled functions from a Gaussian prior distribution, under the Squared Exponential and Matérn Covariance functions for 2-dimensional input vectors; $\boldsymbol{x} \in [0, 1]^2$.

for some new inputs $\boldsymbol{x}^* \in \boldsymbol{X}^* \subset \mathbb{R}^d$. Covariance functions play an important role. They determine a distribution on a space of functions. Figures 3.1 and 3.2 represent random draws from these distributions. GPR takes these as prior distributions, and conditions on the observed data to obtain a posterior distribution. To this end, Bayesian analysis is applied.

Noise-free training set observation

Suppose that $\boldsymbol{y} = (y_i)_{i=1}^n$ are noise-free observations i.e. $y_i = f(\boldsymbol{x}_i)$ where f is the GP. Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ be the n inputs in the training set and $\boldsymbol{X}^* = \{\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, ..., \boldsymbol{x}_{n^*}^*\}$ be n^* new inputs. The joint distribution of $\boldsymbol{y}$ and $\boldsymbol{y}^*$ is:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}^* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(\boldsymbol{X}, \boldsymbol{X}) & K(\boldsymbol{X}, \boldsymbol{X}^*) \\ K(\boldsymbol{X}^*, \boldsymbol{X}) & K(\boldsymbol{X}^*, \boldsymbol{X}^*) \end{bmatrix}\right),$$

(Rasmussen and Williams, 2006)
where $K(\boldsymbol{X}^*, \boldsymbol{X}) = (k(\boldsymbol{x}_i^*, \boldsymbol{x}_j))_{ij}$.

To obtain the posterior distribution over the function given data in the training set, a result on the conditional distribution of a multivariate Gaussian distribution is used to obtain:

$$\boldsymbol{y}^* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}^* \sim \mathcal{N}\Big(K(\boldsymbol{X}^*, \boldsymbol{X})K(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}, \\ K(\boldsymbol{X}^*, \boldsymbol{X}^*) - K(\boldsymbol{X}^*, \boldsymbol{X})K(\boldsymbol{X}, \boldsymbol{X})^{-1}K(\boldsymbol{X}, \boldsymbol{X}^*)\Big)$$

(Rasmussen and Williams, 2006).

Noisy training set observations

Now, suppose that the observation are noisy. Furthermore, assume that the noise, ε, in the observations is additive and standard Gaussian. Thus $\boldsymbol{y} = f(\boldsymbol{x}) + \varepsilon$ where $\varepsilon \sim \mathcal{N}\left(0, \sigma_n^2 \mathbf{I}\right)$. In this case

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \mathbf{I} & K(\boldsymbol{X}, \boldsymbol{X}^*) \\ K(\boldsymbol{X}^*, \boldsymbol{X}) & K(\boldsymbol{X}^*, \boldsymbol{X}^*) \end{bmatrix}\right),$$

and

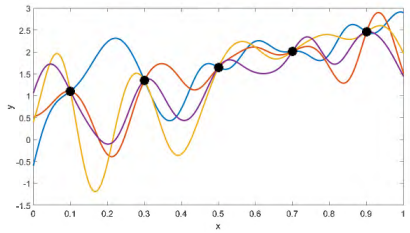
$$\boldsymbol{y}^* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}^* \sim \mathcal{N}\Big(K(\boldsymbol{X}^*, \boldsymbol{X}) \left[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} \boldsymbol{y}, \\ K(\boldsymbol{X}^*, \boldsymbol{X}^*) - K(\boldsymbol{X}^*, \boldsymbol{X}) \left[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} K(\boldsymbol{X}, \boldsymbol{X}^*)\Big)$$

(Rasmussen and Williams, 2006). It follows,

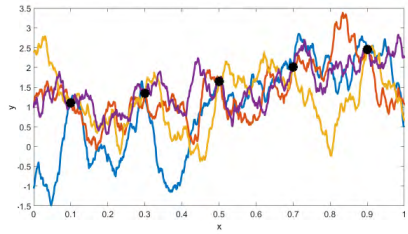
$$\mathbb{E}\left[\boldsymbol{y}^* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}^*\right] = K(\boldsymbol{X}^*, \boldsymbol{X}) \left[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} \boldsymbol{y}, \tag{3.1}$$

$$\text{Cov}\left[\boldsymbol{y}^* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}^*\right] = K(\boldsymbol{X}^*, \boldsymbol{X}^*) - K(\boldsymbol{X}^*, \boldsymbol{X}) \left[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} K(\boldsymbol{X}, \boldsymbol{X}^*), \tag{3.2}$$

$\mathbb{E}\left[\boldsymbol{y}^* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}^*\right]$ is called the prediction of $\boldsymbol{y}^*$. In Bayesian analysis, the predicted value, $\boldsymbol{y}^*$, is a random variable and therefore, it has some uncertainty attached to it. The uncertainty of the prediction may be visualised through the use of 95% confidence intervals around the prediction. The covariance matrix $\text{Cov}\left[\boldsymbol{y}^* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}^*\right]$ is used to compute these confidence intervals. Narrow bounds around the prediction indicate a good prediction, and vice versa. Figures 3.3 and 3.4 show functions that are sampled from the posterior distribution, given a set of points (or observations) which are assumed to be without noise.

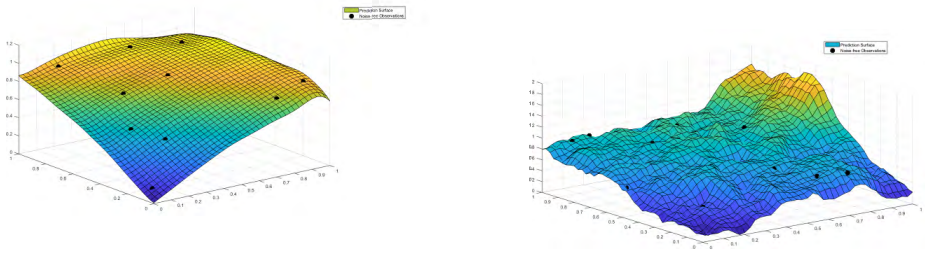


(a) Squared Exponential ($l = 0.1,\ \sigma_f = 1$)



(b) Matern ($l = 0.1,\ \nu = 0.8$)

Fig. 3.3: Three sampled functions from a Gaussian posterior distribution given 5 points (the dots), under the Squared Exponential and Matérn Covariance functions for 1-dimensional inputs, $x \in [0, 1]$.



(a) Squared Exponential ($l = 0.25,\ \sigma_f = 1$)

(b) Matern ($l = 1,\ \nu = 1$)

Fig. 3.4: Sampled functions from a Gaussian posterior distribution given 10 points (the dots), under the Squared Exponential and Matérn Covariance functions for 2-dimensional input vectors; $\boldsymbol{x} \in [0, 1]^2$.

3.1.3 Hyperparameter Optimisation

In GPR, model selection will relate to both the selection of a suitable covariance function from a finite amount of choices, and to the tuning of hyperparameters of covariance functions (Rasmussen and Williams, 2006). To this effect, it is common to specify models in using a hierarchical approach (Rasmussen and Williams, 2006) i.e. inferring different aspects of model selection in different levels. Hierarchical inference occurs as follows; at the bottom level functions are inferred, at the second level hyperparameters are inferred and at the top level a finite set of possible model structures (the type of covariance function) are considered (Rasmussen and Williams, 2006).

The posterior distribution over the functions is given by

$$p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\kappa}, \mathcal{H}) = \frac{\overbrace{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{f}, \boldsymbol{\kappa}, \mathcal{H})}^{\text{Likelihood}} \overbrace{p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H})}^{Prior}}{\underbrace{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H})}_{\text{marginal likelihood}}},$$

Where κ are hyperparameters and

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H}) = \int p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{f}, \boldsymbol{\kappa}, \mathcal{H}) p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H})\ d\mathbf{f}. \tag{3.3}$$

The prior is a Gaussian process: $\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H} \sim \mathcal{N}(0, K(\boldsymbol{X}, \boldsymbol{X}))$.

Suppose the observations are noisy, and that the noise ε is independent identically Gaussian.
In particular, $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\varepsilon}$ and $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma_n^2 \mathbf{I})$.

Thus $\boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{f}, \sigma_n^2 \mathbf{I}\right)$. It follows that

$$p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H}) = \mathcal{N}(\mathbf{0}, K(\boldsymbol{X}, \boldsymbol{X})),$$
$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{f}, \boldsymbol{\kappa}, \mathcal{H}) = \mathcal{N}(\boldsymbol{f}, \sigma_n^2 \mathbf{I}).$$

By evaluating equation (3.3), it can be shown that

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H}) = -\frac{1}{2}\boldsymbol{y}^T \left(\boldsymbol{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \boldsymbol{y} - \frac{1}{2} \det\left(\boldsymbol{K} + \sigma_n^2 \mathbf{I}\right) - \frac{n}{2} \log 2\pi. \tag{3.4}$$

At the level of hyperparameter inference, a prior distribution is imposed on the hyperparameters called a *hyper-prior*. For example, in their paper, Tegnér *et al.* (2017) imposed independent sigmoid Gaussian priors on the hyperparameters. Williams and Rasmussen (1996) used broad Gaussian priors for the hyperparameters. At this stage, the marginal likelihood has the role of a likelihood

$$p(\boldsymbol{\kappa}|\boldsymbol{y}, \boldsymbol{X}, \mathcal{H}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H}) \overbrace{p(\boldsymbol{\kappa}|\boldsymbol{X}, \mathcal{H})}^{hyper-Prior}}{p(\boldsymbol{y}|\boldsymbol{X}, \mathcal{H})}, \tag{3.5}$$

where

$$p(\boldsymbol{y}|\boldsymbol{X}, \mathcal{H}) = \int p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H}) p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H})\, d\kappa. \tag{3.6}$$

The expression at equation (3.6) is usually analytically intractable (Rasmussen and Williams, 2006). Consequently, the full Bayesian inference is usually not performed; instead an approximate treatment of Bayesian inference is preferred- in this dissertation as well. This is done by ignoring the hyperparameter posterior equation (3.5) and instead maximising the log-marginal likelihood with respect to the hyperparameters (Rasmussen and Williams, 2006), i.e.

$$\kappa^* = \arg\max_{\kappa} \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\kappa}, \mathcal{H}).$$

Although it is standard practice to obtain point hyper-parameter estimate by maximising the log-marginal likelihood, other researchers did attempt to perform full Bayesian inference. Tegnér *et al.* (2017) imposed sigmoid Gaussian priors on the hyper-parameters and then obtained a joint posterior over the hyper-parameters. Then used Markov Chain Monte Carlo to sample from this posterior. Others such as Williams and Rasmussen (1996) imposed Gaussian hyper-priors and then used Hybrid Monte Carlo to approximate equation (3.6). A full Bayesian model selection treatment is given in Rasmussen and Williams (2006).

3.2 An Option Pricing Application

In this section, we consider a simple pricing problem. In particular, GPR is used to price a European call option under the Black-Scholes model. Suppose that the

$\mathbb{Q}$-dynamics[1] of the asset price are as follows:

$$dS_t = (r - q)S_t\, dt + \sigma S_t\, dW_t, \quad t \leq T, \tag{3.7}$$

where $r,\ q \geq 0$ and $\sigma > 0$, and W is a Brownian motion process. The payoff function, X, of a European call option (which matures at T) with strike K is $X_T = (S_T - K)^+$[2]. Then, it can be shown that the fair value of the option at $t \leq T$ is

$$\begin{aligned} V_t &= e^{-r(T-t)}\mathbb{E}_{\mathbb{Q}}\left[X_T|\mathcal{F}_t\right], \\ &= e^{-r(T-t)}\mathbb{E}_{\mathbb{Q}}\left[(S_T - K)^+|\mathcal{F}_t\right], \\ &= S_t e^{-q(T-t)}\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2), \end{aligned} \tag{3.8}$$

where,

$$d_1 = \frac{\log\left(\frac{S_t}{K}\right) + \left(r - q + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 = d_1 - \sigma\sqrt{T-t}, \tag{3.9}$$

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{1}{2}z^2}\, dz. \tag{3.10}$$

From equation 3.8, let $C_t := \frac{V_t}{K}$ and $M_t := \frac{S_t}{K}$ (Moneyness), then

$$C_t = M_t e^{-q(T-t)}\Phi(d_1) - e^{-r(T-t)}\Phi(d_2), \tag{3.11}$$

$$d_1 = \frac{\log(M_t) + \left(r - q + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}. \tag{3.12}$$

Moreover, assume that $r = 0.05$ and $q = 0.0$. Equation (3.11) may be used to price a variety of options with matching moneyness. For example, if C_t is known for some M_t, then the option price of an underlying whose initial price is M_t times the strike price, is KC_t. Equation (3.11) is used to generate training and testing (for predictions) data sets. Table 3.1 shows parameters that are used to generate the training set.

Parameter	***Range interval***
Moneyness (M_t)	$[0.05, 2]$
Volatility (σ)	$[0.1, 0.2]$
Time to maturity ($T - t$)	$[1, 1.5]$
Risk-free rate (r)	0.05
Dividend yield (q)	0.0

Tab. 3.1: Training data input parameters.

[1] Risk-neutral dynamics

[2] $x^+ := \max\{x, 0\}$

Table 3.1 shows that in the training set, r and q are fixed, while M_t, σ and $T-t$ vary. In other words, the European call option is priced on a 3d grid. Moreover, assume that for each parameter, we pick 10 equally spaced points within that interval, i.e., successive points differ by $\frac{Max-Min}{10}$. In other words, the option is priced (and the GPR model is trained) on a $10 \times 10 \times 10$ grid. Using the notation from the previous section, it follows that $\boldsymbol{x} = [M_t, \sigma, T-t]^\top$, $\boldsymbol{y} = C_t$. For the purposes of illustration, we assume the Squared Exponential covariance function for the GPR model. The training data set, $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) : i = 1, 2, ..., 10^3\}$, is used to train the GPR model in MATLAB R2020a, on an Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80GHz 8.00 GB RAM laptop.

For predictions (or testing), we consider the following parameter ranges, as depicted by table 3.2. The range intervals of the moneyness and volatility parame-

Parameter	***Range interval***
Moneyness (M_t)	$[0.01, 2.5]$
Volatility (σ)	$[0.05, 0.5]$
Time to maturity ($T-t$)	$[1, 1.5]$
Risk-free rate (r)	0.05
Dividend yield (q)	0.0

Tab. 3.2: Testing data input parameters.

ters, have been enlarged. This will enable us to gauge the accuracy of the GPR model for data that lies within the range of the training set and data which lies beyond the range of the training set. GPR typically performs best on test data that lie within a smaller range than the training data. This means that GPR does not usually perform well near the boundaries of the training data ranges. For testing, the performance of the GPR model is tested on a $30 \times 30 \times 30$ grid. In GPR notation, $\{\boldsymbol{x}_i^* : i = 1, 2, ..., 30^3\}$ is the testing (prediction) set.

Figure 3.5 shows that the actual option values computed using equation 3.8 and the GPR-predicted option values (on the testing set) are very close. Figure 3.6 shows that GPR predictions are less certain for input values that lie beyond the bounds of the training set. In fact, the uncertainty increase the farther away the input is from the bounds of the training set. For inputs that are within the bounds of training set, the uncertainty of the prediction is very small. Moreover, GPR is more accurate for inputs that are within the bounds of the training set. The accuracy also decreases the farther away the input is from the bounds of the training set, as depicted by figure 3.7. Thus, GPR generates very accurate predictions for inputs within the bounds of the training set.

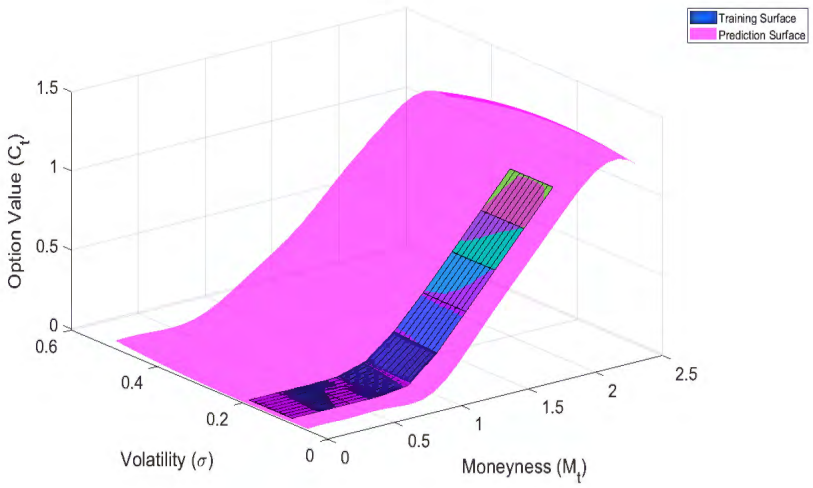


Fig. 3.5: A cross section of the actual (training set) and GPR-predicted option value surface at $r = 0.05,\ q = 0.0,\ T - t = 1$.

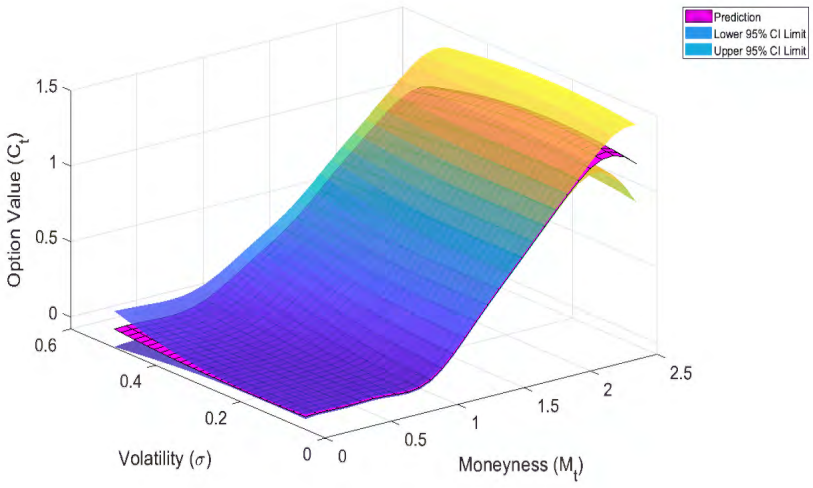


Fig. 3.6: A cross-section of the GPR-predicted option value surface with its 95% confidence intervals, at $r = 0.05,\ q = 0.0,\ T - t = 1$.

The advantage that GPR has over standard methods such as Monte Carlo simulation is that multiple options may be priced at the same time. In our example, at least 30^3 options were priced at the same time. Moreover, the GPR model can be trained and then saved for later use, saving valuable implementation time. It took 0.806014 seconds for GPR to price at least 30^3 options in the testing set, which translates to 2.9852×10^{-5} seconds per option on average.

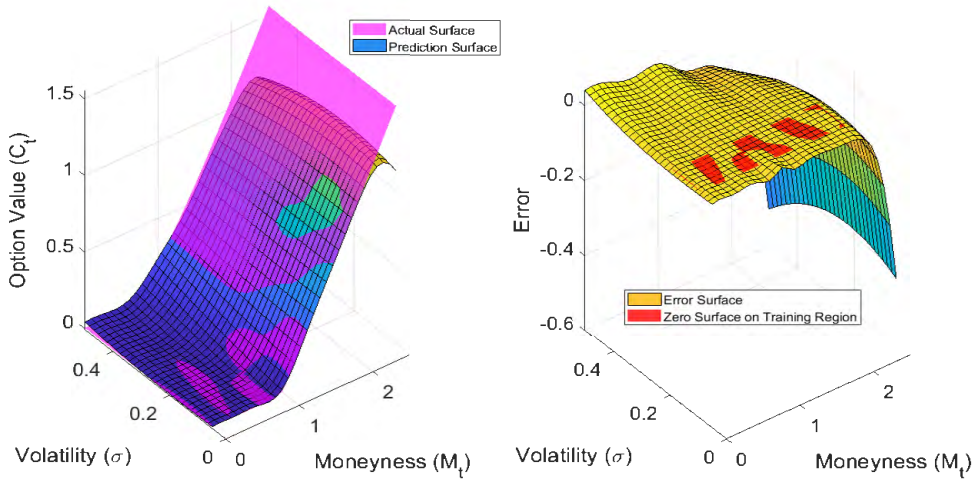


Fig. 3.7: [Left] A cross section of the actual (testing set) and GPR-predicted option value surface at $r = 0.05,\ q = 0.0,\ T - t = 1$. [Right] Error between actual option values (testing set) and GPR predictions.

Chapter 4

Dynamic Programming Approach to American Option Pricing

In this chapter, the pricing of an American option is formulated as an optimal stopping problem. For the purpose of this dissertation, the optimal stopping problem is only considered in discrete time. Consequently, the results derived solve the Bermudan option pricing problem which approximates the American option pricing problem. Then, dynamic programming is shown to be a solution in discrete time. Next, two dynamic programming algorithms, namely, GPR-MC and LSM are presented. Both algorithms are concerned with approximating the continuation value by employing different techniques. GPR-MC employs GPR while LSM employs Least Squares Regression.

Suppose $(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{T}, \mathbb{F}, (\boldsymbol{S}_t)_{t\in\boldsymbol{T}})$ is a multi-period securities market model where $\boldsymbol{S}_t = (S_t^1, S_t^2, ..., S_t^d)$. Suppose the market is complete; then there exists a unique risk neutral measure $\mathbb{Q}$ (Harrison and Kreps, 1979). In a complete market, all contingent claims are attainable and their arbitrage-free value is the expected value (under the $\mathbb{Q}$) of their discounted cashflows (Haugh and Kogan, 2004).

An American option gives its holder the right to exercise the option at any time before the maturity (say T) of the option. Suppose $\Psi_t = \Psi(\boldsymbol{S}_t)$ is an adapted non-negative stochastic process which represents the time t payoff of an American Option. Suppose V_t is the value of the American Option at time t, then clearly $V_t \geq \Psi_t$ as the contrary, i.e. $V_t < \Psi_t$, would admit arbitrage. Define the risk-free money market account process $B_t = e^{\int_0^t r_s\, ds}$, where $(r_t)_{t\geq 0}$ is the risk-free short rate process. Then the fair value of the option is

$$V_t = \sup_{\tau\in\mathbb{T}_{t,T}} \mathbb{E}_{\mathbb{Q}}\left[\frac{B_t\Psi_\tau}{B_\tau}|\mathcal{F}_t\right], \tag{4.1}$$

where $\mathbb{T}_{t,T}$ is the set of stopping times, τ, such that $t \leq \tau \leq T$ (Haugh and Kogan, 2004).

Equation (4.1) is an optimal stopping problem. Dynamic Programming can be used

to tackle this problem (Björk, 2009). In this section, this problem will be considered in discrete time. The arguments that follow are similar to those of Björk (2009).

Consider problem (4.1) in discrete time, i.e.

$$V_n = \sup_{\tau \in \mathbb{T}_{n,T}} \mathbb{E}_{\mathbb{Q}} \left[\frac{B_n \Psi_\tau}{B_\tau} | \mathcal{F}_n \right]. \tag{4.2}$$

This is precisely the value of a Bermudan option, because in the discrete time setting, the option may be exercised at any of a set of finite specified exercise dates.

Consider the following definitions:

- Let $Z_\tau := \frac{B_n \Psi_\tau}{B_\tau}$.
- Let equation (4.2) define the optimal value process V.

$$V_n = \sup_{\tau \in \mathbb{T}_{n,T}} \mathbb{E}_{\mathbb{Q}} \left[\frac{B_n \Psi_\tau}{B_\tau} | \mathcal{F}_n \right] = \sup_{\tau \in \mathbb{T}_{n,T}} \mathbb{E}_{\mathbb{Q}} \left[Z_\tau | \mathcal{F}_n \right]. \tag{4.3}$$

- A stopping time $\hat{\tau}_n$ such that $\mathbb{E}_{\mathbb{Q}} \left[Z_{\hat{\tau}_n} | \mathcal{F}_n \right] = \sup_{\tau \in \mathbb{T}_{n,T}} \mathbb{E}_{\mathbb{Q}} \left[Z_\tau | \mathcal{F}_n \right]$ is said to be optimal.

Fix a time n and consider the value of the following stopping strategies:

1. If $\hat{\tau}_n$ (i.e. the optimal stopping time) is used, then the value of this strategy is V_n by definition.
2. If stopping is done immediately, then the value of this strategy is $\Psi_n = \frac{B_n \Psi_n}{B_n} = Z_n$.
3. Suppose stopping does not occur at time n and then we proceed to time $n+1$. At time $n+1$, the optimal stopping time is used i.e. $\hat{\tau}_{n+1}$ is used. The latter statement implies that we behave optimally at time $n+1$ and onwards. Then by definition, the value of this strategy is V_{n+1} at time $n+1$.
   At time n this value (i.e. V_{n+1}) is equal to the best estimate of the discounted value of V_{n+1}, given all the information up to time n i.e. $\mathbb{E}_{\mathbb{Q}} \left[\frac{B_n}{B_{n+1}} V_{n+1} | \mathcal{F}_n \right]$. To see this, consider the following argument: $\hat{\tau}_{n+1}$ is an optimal time, thus at time n, the value is $\mathbb{E}_{\mathbb{Q}} \left[Z_{\hat{\tau}_{n+1}} | \mathcal{F}_n \right]$ by definition. Then using the tower

property and the definition of V_{n+1},

$$\begin{aligned}
\mathbb{E}_{\mathbb{Q}}\left[Z_{\hat{\tau}_{n+1}}|\mathcal{F}_n\right] &= \mathbb{E}_{\mathbb{Q}}\left[\mathbb{E}_{\mathbb{Q}}\left[Z_{\hat{\tau}_{n+1}}|\mathcal{F}_{n+1}\right]|\mathcal{F}_n\right] \\
&= \mathbb{E}_{\mathbb{Q}}\left[\mathbb{E}_{\mathbb{Q}}\left[\frac{B_n\Psi_{\hat{\tau}_{n+1}}}{B_{\hat{\tau}_{n+1}}}|\mathcal{F}_{n+1}\right]|\mathcal{F}_n\right] \\
&= \mathbb{E}_{\mathbb{Q}}\left[\mathbb{E}_{\mathbb{Q}}\left[\frac{B_n\Psi_{\hat{\tau}_{n+1}}}{B_{\hat{\tau}_{n+1}}}\cdot\frac{B_{n+1}}{B_{n+1}}|\mathcal{F}_{n+1}\right]|\mathcal{F}_n\right] \\
&= \mathbb{E}_{\mathbb{Q}}\left[\frac{B_n}{B_{n+1}}\mathbb{E}_{\mathbb{Q}}\left[\frac{B_{n+1}\Psi_{\hat{\tau}_{n+1}}}{B_{\hat{\tau}_{n+1}}}|\mathcal{F}_{n+1}\right]|\mathcal{F}_n\right] \\
&= \mathbb{E}_{\mathbb{Q}}\left[\frac{B_n}{B_{n+1}}V_{n+1}|\mathcal{F}_n\right]. \qquad (4.4)
\end{aligned}$$

(Björk, 2009). Equation (4.4) is called the *continuation value*.

The strategy at (1.) is the optimal strategy, hence its value is greater than or equal to the value of the strategies (2.) and (3.). Hence,

$$V_n \geq Z_n, \qquad (4.5)$$

$$V_n \geq \mathbb{E}_{\mathbb{Q}}\left[\frac{B_n}{B_{n+1}}V_{n+1}|\mathcal{F}_n\right]. \qquad (4.6)$$

(Björk, 2009).

At time n it is either optimal to stop immediately or not to stop immediately. In the former case, $\hat{\tau}_n = n$ and $V_n = Z_n$; and in the latter case, $\hat{\tau}_n = \hat{\tau}_{n+1}$ and $V_n = \mathbb{E}_{\mathbb{Q}}\left[\frac{B_n}{B_{n+1}}V_{n+1}|\mathcal{F}_n\right]$ (Björk, 2009). Using the latter statement together with equation (4.5) and (4.6), it follows that

$$V_n = \max\{Z_n, \mathbb{E}_{\mathbb{Q}}\left[\frac{B_n}{B_{n+1}}V_{n+1}|\mathcal{F}_n\right]\} \qquad (4.7)$$

$$= \max\{\Psi_n, \mathbb{E}_{\mathbb{Q}}\left[\frac{B_n}{B_{n+1}}V_{n+1}|\mathcal{F}_n\right]\}. \qquad (4.8)$$

(Björk, 2009). The arguments above show the following result:

Proposition 4.1. *The optimal value process V is the solution of the following backward recursion:*

$$V_n = max\{\Psi_n, \mathbb{E}_{\mathbb{Q}}\left[\frac{B_n}{B_{n+1}}V_{n+1}|\mathcal{F}_n\right]\}, \qquad (4.9)$$

$$V_T = \Psi_T. \qquad (4.10)$$

Proof. Proof follows from arguments above. □

In the next two sections, two dynamic programming techniques for valuing American/Bermudan options are introduced.

4.1 Gaussian Process Regression Monte Carlo (GPR-MC)

GPR Monte Carlo (GPR-MC) is a dynamic programming algorithm which approximates the value of an American option by the value of a Bermudan option on the same basket of assets (Goudenège *et al.*, 2020b). In this section, the GPR-MC approach is presented in a similar manner as in section 3.3.1 of Goudenège *et al.* (2019), adapted to the market state variable setting.

Let $\boldsymbol{Y}_t := (y_1(t), y_2(t), ..., y_H(t))^\top$ denote an H-dimensional market state variable of market information, e.g., asset prices, volatilities, interest rates etc. In particular, under the d-dimensional Black-Scholes model, $\boldsymbol{Y}_t = \boldsymbol{S}_t = (S_1(t), S_2(t), ..., S_d(t))^\top$ where $S_i(t)$ is the time t price of the i^{th} asset; and under the d-dimensional Heston model, $\boldsymbol{Y}_t = (S_1(t), v_1(t), S_2(t), v_2(t), ..., S_d(t), v_d(t))^\top$ where $S_i(t)$ and $v_i(t)$ are the time t price and stochastic volatility process of the i^{th} asset respectively. Assume that the $\mathbb{Q}$-dynamics of $\boldsymbol{Y}_t$ are as follows,

$$d\boldsymbol{Y}_t = \mu(\boldsymbol{Y}_t)\, dt + \sigma(\boldsymbol{Y}_t)\, d\boldsymbol{W}_t, \tag{4.11}$$

where $\boldsymbol{W}$ is an H-dimensional $\mathbb{Q}$-Brownian motion with independent components, $\mu(\boldsymbol{Y}_t) \in \mathbb{R}^H$ and $\sigma(\boldsymbol{Y}_t) \in \mathbb{R}^{H\times H}$.

Suppose the time axis $[0, T]$ is partitioned into N exercise times such that $\Delta t := \frac{T}{N}$ is the time increment and $t_n := n\Delta t$, for $n = 1, 2, ..., N$, is the n^{th} exercise date. Define v^{Berm} as the value of a Bermudan option. Moreover, define $D(s,t) := \frac{B_s}{B_t}$, Then

$$v^{Berm}(t_n, \boldsymbol{Y}_{t_n}) = \max\Big(\Psi(\boldsymbol{Y}_{t_n}), \mathbb{E}_{\mathbb{Q}}\left[D(t_n, t_{n+1}) v^{Berm}\left(t_{n+1}, \boldsymbol{Y}_{t_{n+1}}\right) | \mathcal{F}_{t_n}\right]\Big). \tag{4.12}$$

(Goudenège *et al.*, 2019)
where Ψ is the payoff function. For ease of exposition, let $\boldsymbol{x} := \boldsymbol{Y}_{t_n}$ be the market state variable at exercise date t_n. Then

$$v^{Berm}(t_n, \boldsymbol{x}) = \max\left(\Psi(\boldsymbol{x}), \mathbb{E}_{\boldsymbol{x},t_n}\left[D(t_n, t_{n+1}) v^{Berm}\left(t_{n+1}, \boldsymbol{Y}_{t_{n+1}}\right)\right]\right). \tag{4.13}$$

(Goudenège *et al.*, 2019)
If $v^{Berm}(t_{n+1}, \cdot)$ is known, then $v^{Berm}(t_n, \cdot)$ can be calculated by approximating the expectation $\mathbb{E}_{\boldsymbol{x},t_n}\left[v^{Berm}\left(t_{n+1}, \boldsymbol{Y}_{t_{n+1}}\right)\right]$ and then evaluating equation (4.13). To approximate the expectation, a set of P points,

$$X^n = \{\boldsymbol{x}^{n,p} = \left(x_1^{n,p}, ..., x_H^{n,p}\right)^\top : \ p = 1, 2, ..., P\} \subseteq \mathbb{R}^H,$$

are considered (Goudenège *et al.*, 2019). These represent the realisations of the market state variable, $\boldsymbol{Y}$, at t_n. For each $\boldsymbol{x}^{n,p} \in X^n$, generate M possible realisations

of $\boldsymbol{Y}_{t_{n+1}}$ according to the conditional distribution $\boldsymbol{Y}_{t_{n+1}}|\boldsymbol{Y}_{t_n} = \boldsymbol{x}^{n,p}$ by means of a one-step ahead simulation. Symbolically, for each $\boldsymbol{x}^{n,p} \in X^n$, simulate a set of M realisations

$$\tilde{X}_p^n = \{\tilde{\boldsymbol{x}}^{n,p,m} = \left(\tilde{x}_1^{n,p,m}, ..., \tilde{x}_H^{n,p,m}\right)^\top \ : \ m = 1, 2, ..., M\},$$

such that

$$\tilde{\boldsymbol{x}}^{n,p,m} \sim \boldsymbol{Y}_{t_{n+1}}|\boldsymbol{Y}_{t_n} = \boldsymbol{x}^{n,p} \qquad \text{for } m = 1, ..., M,$$

i.e., where $\tilde{\boldsymbol{x}}^{n,p,m}$ has same the distribution as $\boldsymbol{Y}_{t_{n+1}}|\boldsymbol{Y}_{t_n} = \boldsymbol{x}^{n,p}$, which is used to generate one-step ahead realisations. Then, the value of the option for each $\boldsymbol{x}^{n,p} \in X^n$ can be approximated by

$$\hat{v}^{Berm}(t_n, \boldsymbol{x}^{n,p}) = \max\Big(\Psi(\boldsymbol{x}^{n,p}), \frac{1}{M}\sum_{m=1}^{M} D(t_n, t_{n+1}) v^{Berm}\left(t_{n+1}, \tilde{\boldsymbol{x}}^{n,p,m}\right)\Big) \quad (4.14)$$

(Goudenège *et al.*, 2019).
The value of function $v^{Berm}(T, \cdot)$ where $t_N = T$, is simply the payoff function $\Psi(\cdot)$ and thus it is known. By equation (4.14), it follows that $v^{Berm}(t_{N-1}, \boldsymbol{x}^{N-1,p})$ is also known for all $p = 1, 2, ..., P$. To compute $v^{Berm}(t_{N-2}, \boldsymbol{x}^{N-2,p})$ requires the knowledge of $v^{Berm}(t_{N-1}, \cdot)$ for all points in $\tilde{X}_p^{N-2}$ for all p. However from the previous step, $\hat{v}^{Berm}(t_{N-1}, \cdot)$ is only known for points X^{N-1}, thus equation (4.14) cannot be evaluated directly. This problem can be solved by approximating the function $\hat{v}^{Berm}(t_{N-1}, \cdot)$ using Gaussian Process Regression (GPR).

The set $\mathcal{D} = \{\left(\boldsymbol{x}^{N-1,p}, \hat{v}^{Berm}(t_{N-1}, \boldsymbol{x}^{N-1,p})\right) \ : \ p = 1, ..., P\}$ is the training set in this step (Goudenège *et al.*, 2019). Proceed similarly for steps $N-2, ..., 1$. Define $v_n^{Berm,GPR}(\cdot)$ as the GPR-approximation of $\hat{v}^{Berm}(t_n, \cdot)$ trained using the set $\{\left(\boldsymbol{x}^{n,p}, \hat{v}^{Berm}(t_n, \boldsymbol{x}^{n,p})\right) : p = 1, .., P\}$. It follows that the value of the option at time t_{n-1} can be computed using the prediction/posterior conditioned on $\tilde{X}_p^{n-1}$ for all p as

$$\hat{v}^{Berm}(t_{n-1}, \boldsymbol{x}^{n-1,p}) = \max\Big(\Psi(\boldsymbol{x}^{n-1,p}), \frac{1}{M}\sum_{m=1}^{M} D(t_{n-1}, t_n) v_n^{Berm,GPR}\left(\tilde{\boldsymbol{x}}^{n-1,p,m}\right)\Big), \quad (4.15)$$

(Goudenège *et al.*, 2019).
$v_{n-1}^{Berm,GPR}$ is trained and then used in the next step. At time 0 the value of the option can be computed as

$$\hat{v}^{Berm}(0, \boldsymbol{Y}_0) = \max\Big(\Psi(\boldsymbol{Y}_0), \frac{1}{M}\sum_{m=1}^{M} D(0, t_1) v_1^{Berm,GPR}\left(\tilde{\boldsymbol{x}}^{0,m}\right)\Big), \quad (4.16)$$

where $\tilde{\boldsymbol{x}}^{0,1}, ..., \tilde{\boldsymbol{x}}^{0,M}$ are the realisations of $\boldsymbol{Y}_{t_1}|\boldsymbol{Y}_0$ (Goudenège *et al.*, 2019).

In the next subsection, an algorithm for numerically implementing GPR-MC is presented.

GPR-MC Numerical Implementation

Recall the $\mathbb{Q}$-dynamics of the market state variable, $\boldsymbol{Y}_t$, are as follows,

$$d\boldsymbol{Y}_t = \mu(\boldsymbol{Y}_t)\, dt + \sigma(\boldsymbol{Y}_t)\, d\boldsymbol{W}_t, \tag{4.17}$$

where $\boldsymbol{W}$ is an H-dimensional $\mathbb{Q}$-Brownian motion with independent components, $\mu(\boldsymbol{Y}_t) \in \mathbb{R}^H$ and $\sigma(\boldsymbol{Y}_t) \in \mathbb{R}^{H\times H}$. As noted above, a set of market state realisations, $X^n = \{\boldsymbol{x}^{n,p} = (x_1^{n,p}, ..., x_H^{n,p})^\top : \ p = 1, ..., P\}$, is required at time t_n i.e. $\boldsymbol{x}^n := \boldsymbol{Y}_{t_n}$, of which $\boldsymbol{x}^{n,p}$ is the p^{th} realisation of $\boldsymbol{Y}_{t_n}$. Moveover, for each $\boldsymbol{x}^{n,p} \in X^n$, a one-step Monte Carlo simulation is used to generate the set $\tilde{X}_p^n = \{\tilde{\boldsymbol{x}}^{n,p,m} = (x_1^{n,p,m}, ..., x_H^{n,p,m})^\top : \ m = 1, ..., M\}$ according to the conditional distribution of $\boldsymbol{Y}_{t_{n+1}}|\boldsymbol{Y}_{t_n} = \boldsymbol{x}^{n,p}$. In what follows, algorithms for generating the aforementioned sets, which are critical for GPR-MC, are presented under the Black-Scholes and Heston model. Then, an algorithm for the numerical implementation of GPR-MC is shown.

Black-Scholes Model

The risk-neutral dynamics of S under the d-dimensional Black-Scholes model are of the form,

$$\begin{aligned} &dS_i(t) = (r - \eta_i)S_i(t)\, dt + \sigma_i S_i(t)\, dW_i(t) \qquad i = 1, ..., d, \\ &d\langle W_i, W_j\rangle_t = \rho_{ij}\, dt, \quad |\rho_{ij}| \leq 1, \\ &1 \leq i, j \leq d. \end{aligned} \tag{4.18}$$

where $r, \eta_i \geq 0$ and $\sigma_i > 0\ \forall i$. Under this model, the sets $X^n(\forall n)$ and $\tilde{X}_p^n(\forall n, p)$ are generated as follows;

For $n= N-1, ..., 1$

Generate $\left(Z_S^{p,1}, ..., Z_S^{p,d}\right)^\top \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{I}_d)$,

compute $\left(\tilde{Z}_S^{p,1}, ..., \tilde{Z}_S^{p,d}\right)^\top := \boldsymbol{L}\left(Z_S^{p,1}, ..., Z_S^{p,d}\right)^\top$,

$\forall p = 1, ..., P \quad$ where $\boldsymbol{L}\boldsymbol{L}^\top = \boldsymbol{\Gamma}, \quad \boldsymbol{\Gamma} = (\rho_{ij})_{i,j=1}^d$.

$$\boldsymbol{x}^{n,p} := \begin{bmatrix} S_1^p(t_n) \\ \vdots \\ S_d^p(t_n) \end{bmatrix} = \begin{bmatrix} S_1(0)\text{exp}\left[(r - \eta_1 - \frac{\sigma_1^2}{2})t_n + \sigma_1\sqrt{t_n}\tilde{Z}_S^{p,1}\right] \\ \vdots \\ S_d(0)\text{exp}\left[(r - \eta_d - \frac{\sigma_d^2}{2})t_n + \sigma_d\sqrt{t_n}\tilde{Z}_S^{p,d}\right] \end{bmatrix}. \tag{4.19}$$

Generate $\left(Z_S^{p,m,1}, ..., Z_S^{p,m,d}\right)^\top \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{I}_d)$,

compute $\left(\tilde{Z}_S^{p,m,1}, ..., \tilde{Z}_S^{p,m,d}\right)^\top := \boldsymbol{L}\left(Z_S^{p,m,1}, ..., Z_S^{p,m,d}\right)^\top$,

$\forall p = 1, ..., P,\ m = 1, ..., M.$

$$\tilde{\boldsymbol{x}}^{n,p,m} := \begin{bmatrix} S_1^p(t_n)\text{exp}\left[(r - \eta_1 - \frac{\sigma_1^2}{2})\Delta t + \sigma_1\sqrt{\Delta t}\tilde{Z}_S^{p,m,1}\right] \\ \vdots \\ S_d^p(t_n)\text{exp}\left[(r - \eta_d - \frac{\sigma_d^2}{2})\Delta t + \sigma_d\sqrt{\Delta t}\tilde{Z}_S^{p,m,d}\right] \end{bmatrix}. \tag{4.20}$$

End

At $n = 0$

Generate $\left(Z_S^{m,1}, ..., Z_S^{m,d}\right)^\top \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{I}_d)$.

Compute $\left(\tilde{Z}_S^{m,1}, ..., \tilde{Z}_S^{m,d}\right)^\top := \boldsymbol{L}\left(Z_S^{m,1}, ..., Z_S^{m,d}\right)^\top$,

$\forall m = 1, ..., M.$

$$\tilde{\boldsymbol{x}}^{0,m} = \begin{bmatrix} S_1(0)\text{exp}\left[(r - \eta_1 - \frac{\sigma_1^2}{2})\Delta t + \sigma_1\sqrt{\Delta t}\tilde{Z}_S^{m,1}\right] \\ \vdots \\ S_d(0)\text{exp}\left[(r - \eta_d - \frac{\sigma_d^2}{2})\Delta t + \sigma_d\sqrt{\Delta t}\tilde{Z}_S^{m,d}\right] \end{bmatrix}. \tag{4.21}$$

At this point it is imperative to bring to the reader's attention that the vectors $\boldsymbol{x}^{1,p}, ..., \boldsymbol{x}^{n,p}, ..., \boldsymbol{x}^{N,p}$ $\forall p$ have been generated in such a way that they are serially uncorrelated, otherwise GPR-MC will produce incorrect results. Of course, in the Black-Scholes model, generating these vectors is trivial since an analytical expression for the asset price exists. When considering models whose stochastic differential equations (SDEs) admit no analytical solution, numerical schemes such as Euler-Maruyama and Multi-Quadratic Exponential (MQE) have to be implemented and these create serial correlation (along each path). An example of where this problem occurs is when simulating the Heston model using the aforementioned numerical schemes. To overcome this issue, this paper proceeds as follows: simu-

late a path from time 0 time t_1. Set $\boldsymbol{x}^{1,p}$ to the simulated asset prices at t_1. Generate new random numbers and then simulate a path from time 0 time t_2. Set $\boldsymbol{x}^{2,p}$ to the simulated asset prices at time t_2. Proceed similarly for times $t_3, ..., t_{N-1}$. Then the vectors $\boldsymbol{x}^{1,p}, ..., \boldsymbol{x}^{n,p}, ..., \boldsymbol{x}^{N,p}$ $\forall p$ are serially uncorrelated since they are generated using different random numbers. Next, an algorithm for generating Heston asset prices is presented.

Heston Model

The risk-neutral dynamics of the d-dimensional Heston model are of the form,

$$dS_i(t) = (r - \eta_i)S_i(t)\, dt + \sqrt{v_i(t)}S_i(t)\, dW_i(t), \tag{4.22}$$
$$dv_i(t) = \kappa_i(\theta_i - v_i(t))\, dt + \sigma_i\sqrt{v_i(t)}\, d\tilde{W}_i(t), \tag{4.23}$$

such that

$$\begin{aligned}
&d\langle W_i, W_j\rangle_t = \rho_{ij}^{SS} dt,\\
&d\langle W_i, \tilde{W}_j\rangle_t = \rho_{ij}^{SV} dt\\
&d\langle \tilde{W}_i, \tilde{W}_j\rangle_t = \rho_{ij}^{VV} dt,\\
&\theta_i, \kappa_i, \sigma_i > 0,\\
&\rho_{ij}^{SS}, \rho_{ij}^{SV}, \rho_{ij}^{VV} \in [-1, 1],\\
&1 \leq i, j \leq d.
\end{aligned}$$

The reader should refer to A.2 for restrictions imposed on the correlation structure of the above model.

Equation (A.25) can be expressed in vector notation as follows

$$\begin{aligned}
\boldsymbol{X}(t_{n+1}) =& \boldsymbol{X}(t_n) + (r - \eta_i)\Delta\mathbf{1}_d - \frac{1}{2}\left(\gamma_1\boldsymbol{v}(t_n) + \gamma_2\boldsymbol{v}(t_{n+1})\right)\\
&+\boldsymbol{P}\boldsymbol{f} + \boldsymbol{s}\boldsymbol{L}^*\boldsymbol{Z},
\end{aligned} \tag{4.24}$$

where $\boldsymbol{X}(\cdot) = (X_1(\cdot), ..., X_d(\cdot))^\top$,
$\boldsymbol{v}(\cdot) = (v_1(\cdot), ..., v_d(\cdot))^\top$, $\boldsymbol{P} = diag(\rho_1, ..., \rho_d)$,
$\boldsymbol{f} = (f_1, ..., f_d)^\top$, $\boldsymbol{s} = diag(s_1, ..., s_d)$, $\mathbf{1}_d = (1, ..., 1)^\top$,
$\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, where $\boldsymbol{I}_d$ is the $d \times d$ Identity matrix,
$\boldsymbol{L}^*$ is defined in A.17.

Continuing from the prior discussion, $X^n(\forall n)$ and $\tilde{X}^{n,p}$ $(\forall n, p)$ may be generated as follows:

For $n = N-1, ..., 1$

For $k = 1, ..., n$

– Given $\boldsymbol{v}^p(t_{k-1}) = (v_1^p(t_{k-1}), ..., v_d^p(t_{k-1}))^\top$, Apply (1.) & (2.) of the MQE scheme in A.2 to obtain $\boldsymbol{v}^p(t_k)\ \forall p = 1, ..., P$.

– Generate $\left(Z_S^{p,1}, ..., Z_S^{p,d}\right)^\top \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{I}_d)\ \forall p = 1, ..., P$.

– Given $\boldsymbol{v}^p(t_k),\ \boldsymbol{v}^p(t_{k-1})$ & $\boldsymbol{X}^p(t_{k-1})$[a], compute $\boldsymbol{X}^p(t_k)$ using equation (4.24) $\forall p = 1, ..., P$.

– Set $\boldsymbol{S}^p(t_k) := \left(\exp(X_1^p(t_k)), ..., \exp(X_d^p(t_k))\right)^\top$.

End

Set $\boldsymbol{x}^{n,p} := \begin{bmatrix} \boldsymbol{S}^p(t_n) \\ \boldsymbol{v}^p(t_n) \end{bmatrix}\ \forall p = 1, ..., P$.

– Given $\boldsymbol{v}^p(t_n) = (v_1^p(t_n), ..., v_d^p(t_n))^\top$, Apply (1.) & (2.) of the MQE scheme in A.2 to obtain $\boldsymbol{v}^{p,m}(t_{n+1})\ \forall p = 1, ..., P,\ m = 1, ..., M$.

– Generate $\left(Z_S^{p,m,1}, ..., Z_S^{p,m,d}\right)^\top \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{I}_d)\ \forall p = 1, ..., P,\ m = 1, ..., M$.

– Given $\boldsymbol{v}^{p,m}(t_{n+1}),\ \boldsymbol{v}^p(t_n)$ & $\boldsymbol{X}^p(t_n)$, compute $\boldsymbol{X}^{p,m}(t_{n+1})$ using equation (4.24) $\forall p = 1, ..., P,\ m = 1, ..., M$.

– Set $\boldsymbol{S}^{p,m}(t_{n+1}) := \left(\exp(X_1^{p,m}(t_{n+1})), ..., \exp(X_d^{p,m}(t_{n+1}))\right)^\top$.

Set $\tilde{\boldsymbol{x}}^{n,p,m} := \begin{bmatrix} \boldsymbol{S}^{p,m}(t_{n+1}) \\ \boldsymbol{v}^{p,m}(t_{n+1}) \end{bmatrix}\ \forall p = 1, ..., P,\ m = 1, ..., M$.

End

– Given $\boldsymbol{v}^p(0) = (v_1(0), ..., v_d(0))^\top$, Apply (1.) & (2.) of the MQE scheme in A.2 to obtain $\boldsymbol{v}^m(t_1)\ \forall m = 1, ..., M$.

– Generate $\left(Z_S^{m,1}, ..., Z_S^{m,d}\right)^\top \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{I}_d)\ m = 1, ..., M$.

– Given $\boldsymbol{v}^m(t_1),\ \boldsymbol{v}(0)$ & $\boldsymbol{X}(0)$, compute $\boldsymbol{X}^m(t_1)$ using equation (4.24) $\forall m = 1, ..., M$.

– Set $\boldsymbol{S}^m(t_1) := (\exp(X_1^m(t_1)), ..., \exp(X_d^m(t_1)))^\top$.

Set $\tilde{\boldsymbol{x}}^{0,m} := \begin{bmatrix} \boldsymbol{S}^m(t_1) \\ \boldsymbol{v}^m(t_1) \end{bmatrix}\ \forall m = 1, ..., M$.

[a] Not to be confused with the GPR-MC set $X^n\ \forall n$

GPR-MC Algorithm

Consider an American option with payoff function $\Psi(\cdot)$ and maturity T. Partition the time axis $[0, T]$ into N discrete exercise dates $t_n := n\Delta t$, where $n = 1, ..., N$ and $\Delta t := \frac{T}{N}$. Then the GPR-MC algorithm proceeds backwards as follows:

Pre-processing

Generate $\boldsymbol{x}^{n,p},\ \tilde{\boldsymbol{x}}^{0,m},\ \tilde{\boldsymbol{x}}^{p,n,m}\ \forall p = 1, ..., P;\ n = 1, ..., N-1;\ m = 1, ..., M.$

At t_{N-1}

Compute $\hat{v}^{Berm}(t_{N-1}, \boldsymbol{x}^{p,N-1}) = \max\left\{\Psi(\boldsymbol{x}^{p,N-1}),\ \frac{1}{M}\sum_{m=1}^{M} D(t_{N-1}, t_N)\Psi(\tilde{\boldsymbol{x}}^{p,m,N-1})\right\}$ for $p = 1, ..., P$.

Let $\mathcal{D} = \left\{(\boldsymbol{x}^{p,N-1}, \hat{v}^{Berm}(t_{N-1}, \boldsymbol{x}^{p,N-1})) :\ p = 1, ..., P\right\}$ be a training set.

Train a GPR model using $\mathcal{D}$ to obtain $v_{N-1}^{Berm,GPR}$.

At $t_k \quad (k = N-2, ..., 1)$

Compute

$$\hat{v}^{Berm}(t_k, \boldsymbol{x}^{p,k}) = \max\left\{\Psi(\boldsymbol{x}^{p,k}),\ \frac{1}{M}\sum_{m=1}^{M} D(t_k, t_{k+1})\ v_{k+1}^{Berm,GPR}(\tilde{\boldsymbol{x}}^{p,m,k})\right\}$$

for $p = 1, ..., P$.

Let $\mathcal{D} = \left\{(\boldsymbol{x}^{p,k}, \hat{v}^{Berm}(t_k, \boldsymbol{x}^{p,k})) :\ p = 1, ..., P\right\}$ be a training set.

Train a GPR model using $\mathcal{D}$ to obtain $v_k^{Berm,GPR}$.

At $t_0 = 0$

$$\hat{v}(0, \boldsymbol{Y}_0) = \max\left\{\Psi(\boldsymbol{Y}_0), \frac{1}{M}\sum_{m=1}^{M} D(0, t_1)\ v_1^{Berm,GPR}(\tilde{\boldsymbol{x}}^{0,m})\right\}. \tag{4.25}$$

4.2 Least Squares Regression Method (LSM)

The Least Squares Regression Method of Longstaff and Schwartz (2001) approximates the continuation value by employing regression on a truncated L^2 basis (Goudenège *et al.*, 2019). This approximation is done by expressing the conditional expectation

$$\mathbb{E}_{\mathbb{Q}}\left[D(t_{n-1}, t_n)v(t_n, \boldsymbol{Y}_{t_n})|\boldsymbol{Y}_{t_{n-1}}\right],$$

as a linear combination of basis functions $\phi_i : \mathbb{R}^H \to \mathbb{R}$ as

$$\begin{aligned}\mathbb{E}_{\mathbb{Q}}\left[D(t_{n-1},t_n)v(t_n,\boldsymbol{Y}_{t_n})|\boldsymbol{Y}_{t_{n-1}}\right] &= \sum_{i=1}^{\infty} c_{n-1}^i \phi_i(\boldsymbol{Y}_{t_{n-1}}) \\ &\approx \sum_{i=1}^{K} c_{n-1}^i \phi_i(\boldsymbol{Y}_{t_{n-1}}) \\ &= \boldsymbol{c}_{n-1}^{\top}\boldsymbol{\Phi}(\boldsymbol{Y}_{t_{n-1}}),\end{aligned} \tag{4.26}$$

where $\boldsymbol{\Phi}(\cdot) = (\phi_1(\cdot),\ ...,\ \phi_K(\cdot))^{\top}$, $\boldsymbol{c}_{n-1} = (c_{n-1}^1,\ ...,\ c_{n-1}^K)^{\top}$ (Shen, 2014) and $\boldsymbol{Y}$ is as already defined.
Define the squared error by

$$L := \mathbb{E}_{\mathbb{Q}}\Big(\mathbb{E}_{\mathbb{Q}}\left[D(t_{n-1},t_n)v(t_n,\boldsymbol{Y}_{t_n})|\boldsymbol{Y}_{t_{n-1}}\right] - \boldsymbol{c}_{n-1}^{\top}\boldsymbol{\Phi}(\boldsymbol{Y}_{t_{n-1}})\Big)^2, \tag{4.27}$$

(Shen, 2014).
Then choose $\boldsymbol{c}_{n-1}$ that minimise 4.27. It can be shown that

$$\boldsymbol{c}_{n-1} = \left(\mathbb{E}_{\mathbb{Q}}\left[\boldsymbol{\Phi}(\boldsymbol{Y_{t_{n-1}}})\boldsymbol{\Phi}(\boldsymbol{Y_{t_{n-1}}})^{\top}\right]\right)^{-1}\mathbb{E}_{\mathbb{Q}}\left[\boldsymbol{\Phi}(\boldsymbol{Y_{t_{n-1}}})D(t_{n-1},t_n)v(t_n,\boldsymbol{Y}_{t_n})\right], \tag{4.28}$$

(Shen, 2014).
Now, Monte Carlo Simulation can be used to estimate $\boldsymbol{c}_{n-1}$. In particular, P independent paths, $\boldsymbol{y}_{t_n}(p)$[1] for $p = 1, ..., P$ & $n = 1, ..., N$, can be generated with initial value $\boldsymbol{y}_{t_0}$. Then using the Law of Large Numbers, the $K \times K$ matrix $\mathbb{E}_{\mathbb{Q}}\left[\boldsymbol{\Phi}(\boldsymbol{Y_{t_{n-1}}})\boldsymbol{\Phi}(\boldsymbol{Y_{t_{n-1}}})^{\top}\right]$ can be approximated by

$$\frac{1}{P}\sum_{p=1}^{P}\boldsymbol{\Phi}(\boldsymbol{y}_{t_{n-1}}(p))\boldsymbol{\Phi}(\boldsymbol{y}_{t_{n-1}}(p))^{\top}, \tag{4.29}$$

(Shen, 2014).
and the $K \times 1$ matrix $\mathbb{E}_{\mathbb{Q}}\left[\boldsymbol{\Phi}(\boldsymbol{Y_{t_{n-1}}})D(t_{n-1},t_n)v(t_n,\boldsymbol{S}_{t_n})\right]$ can be approximated as

$$\frac{1}{P}\sum_{p=1}^{P}\boldsymbol{\Phi}(\boldsymbol{y}_{t_{n-1}}(p))D(t_{n-1},t_n)v(t_n,\boldsymbol{y}_{t_n}(p)), \tag{4.30}$$

assuming that $v(t_n, \boldsymbol{s}_{t_n}(p))$ is known for all $p = 1, ..., P$. Then the least squares estimate $\hat{\boldsymbol{c}}_{n-1}$ of $\boldsymbol{c}_{n-1}$ becomes

$$\hat{\boldsymbol{c}}_{n-1} = \Big(\sum_{p=1}^{P}\boldsymbol{\Phi}(\boldsymbol{y}_{t_{n-1}}(p))\boldsymbol{\Phi}(\boldsymbol{y}_{t_{n-1}}(p))^{\top}\Big)^{-1}\sum_{p=1}^{P}\boldsymbol{\Phi}(\boldsymbol{y}_{t_{n-1}}(p))D(t_{n-1},t_n)v(t_n,\boldsymbol{y}_{t_n}(p)), \tag{4.31}$$

[1] A realisation of $\boldsymbol{Y}_{t_n}$

(Shen, 2014). Define,

$$\boldsymbol{F} := \begin{bmatrix} \phi_1(\boldsymbol{y}_{t_{n-1}}(1)) & \phi_1(\boldsymbol{y}_{t_{n-1}}(2)) & \cdots & \phi_1(\boldsymbol{y}_{t_{n-1}}(P)) \\ \phi_2(\boldsymbol{y}_{t_{n-1}}(1)) & \phi_2(\boldsymbol{y}_{t_{n-1}}(2)) & \cdots & \phi_2(\boldsymbol{y}_{t_{n-1}}(P)) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_K(\boldsymbol{y}_{t_{n-1}}(1)) & \phi_K(\boldsymbol{y}_{t_{n-1}}(2)) & \cdots & \phi_K(\boldsymbol{y}_{t_{n-1}}(P)) \end{bmatrix},$$

$$\bar{\boldsymbol{V}} := D(t_{n-1}, t_n) \begin{bmatrix} v(t_n, \boldsymbol{y}_{t_n}(1)) \\ \vdots \\ v(t_n, \boldsymbol{y}_{t_n}(P)) \end{bmatrix},$$

then,

$$\hat{\boldsymbol{c}}_{n-1} = (\boldsymbol{F}\boldsymbol{F}^\top)^{-1}\boldsymbol{F}\bar{\boldsymbol{V}}. \tag{4.32}$$

and the least squares estimate of the continuation value becomes

$$\mathbb{E}_{\mathbb{Q}}\left[D(t_{n-1}, t_n)v(t_n, \boldsymbol{Y}_{t_n})|\boldsymbol{Y}_{t_{n-1}}\right] \approx \hat{\mathbf{c}}_{n-1}^\top \boldsymbol{\Phi}(\boldsymbol{Y}_{t_{n-1}}). \tag{4.33}$$

A high-level representation of the LSM algorithm is shown below, as presented in Glasserman (2013):

1. Simulate P independent stock price paths $\{\boldsymbol{Y}_{t_n}(p)\}_{n=1}^N$ for $p = 1, \ ..., \ P$.
2. At maturity (i.e. t_N), set $v(t_N, \boldsymbol{Y}_{t_N}(p)) = \Psi(\boldsymbol{Y}_{t_N}(p))$ for $p = 1, \ ..., \ P$.
3. Given $D(t_{n-1}, t_n)v(t_n, \boldsymbol{Y}_{t_n}(p))$ for $p = 1, \ ..., \ P$ and $n = 1, ..., N$, use Least Squares Regression shown above to estimate $\hat{\boldsymbol{c}}_{n-1}$ for $n = N, \ ..., \ 2$. Compute $\mathbb{E}_{\mathbb{Q}}\left[D(t_{n-1}, t_n)v(t_n, \boldsymbol{Y}_{t_n})|\boldsymbol{Y}_{t_{n-1}}\right] \approx \hat{\mathbf{c}}_{n-1}^\top \boldsymbol{\Phi}(\boldsymbol{Y}_{t_{n-1}})$.
4. Set $v(t_{n-1}, \boldsymbol{Y}_{t_{n-1}}(p)) = \begin{cases} \Psi(\boldsymbol{Y}_{t_{n-1}}(p)) & \text{if } \Psi(\boldsymbol{Y}_{t_{n-1}}(p)) \geq \hat{\boldsymbol{c}}_{n-1}^\top \boldsymbol{\Phi}(\boldsymbol{Y}_{t_{n-1}}(p)), \\ \hat{\boldsymbol{c}}_{n-1}^\top \boldsymbol{\Phi}(\boldsymbol{Y}_{t_{n-1}}(p)) & \text{if } \Psi(\boldsymbol{Y}_{t_{n-1}}(p)) < \hat{\boldsymbol{c}}_{n-1}^\top \boldsymbol{\Phi}(\boldsymbol{Y}_{t_{n-1}}(p)). \end{cases}$
5. $v(0, \boldsymbol{Y}_0) = \max\left\{\Psi(\boldsymbol{Y}_0), \frac{1}{P}\sum_{p=1}^P D(0, t_1)v(t_1, \boldsymbol{Y}_{t_1}(p))\right\}$.

LSM Numerical Implementation

In this subsection, a sketch implementation of the LSM algorithm is presented. Basis functions used to price the various option types under the Black-Scholes and Heston models, are also chosen.

In this dissertation, the following multi-asset American options are priced; a geometric mean basket put, an arithmetic mean basket put and a maximum call. The chosen basis functions under each model are shown below:

1. ***Black-Scholes model***
   All options have the following basis functions:

$$\left\{ \prod_{k=1}^{d} S_k^{\alpha_k} \,:\, \sum_{k=1}^{d} \alpha_k \leq 4,\ \alpha_k \in \mathbb{N}_0 \ \forall k = 1, ..., d \right\}.$$

   These are precisely the terms of a 4^{th} degree d-dimensional polynomial.

2. ***Heston model***
   All options have the following basis functions:

$$\left\{ \prod_{k=1}^{d} S_k^{\alpha_{2k-1}} \cdot v_k^{\alpha_{2k}} \,:\, \sum_{k=1}^{2d} \alpha_k \leq 3,\ \alpha_k \in \mathbb{N}_0 \ \forall k = 1, ..., 2d \right\}.$$

   These are precisely the terms of a 3^{rd} degree $2d$-dimensional polynomial.

In addition to the basis functions above, an extra function is added depending on the type of option. The geometric mean basket, the arithmetic mean basket, and the maximum options have $\left(\prod_{i=1}^{d} S_i\right)^{\frac{1}{d}}$, $\frac{1}{d}\sum_{i=1}^{d} S_i$ and $\max_{1\leq i\leq d} S_i$ added on, respectively.

Recall a prior definition of $\boldsymbol{Y}$. An algorithm based on LSM is presented below:

−Generate P market state variable paths $(\boldsymbol{Y}_{t_n}(\omega))_{n=1}^{N}$ $\forall \omega \in \{\omega_1, ..., \omega_P\}$.
−Set $\hat{v}_{t_N}(\omega) := \Psi(\boldsymbol{Y}_{t_N}(\omega))\ \forall \omega \in \{\omega_1, ..., \omega_P\}$.
For $k = N-1, ..., 1$
  −Compute realised continuation values
  $\hat{v}_{t_k}(\omega) = D(t_k, t_{k+1})\hat{v}_{t_{k+1}}(\omega)\ \forall \omega \in \{\omega_1, ..., \omega_P\}$.
  −Let $\Omega^* := \{\omega \in \{\omega_1, ..., \omega_P\} :\ \Psi(\boldsymbol{Y}_{t_k}(\omega)) > 0\} \subseteq \{\omega_1, ..., \omega_P\}$.
  − Regression will be implemented only on in-the-money paths, hence,
  − set $A := \{\boldsymbol{Y}_{t_k}(\omega) :\ \omega \in \Omega^*\},\ B := \{\hat{v}_{t_k}(\omega) :\ \omega \in \Omega^*\}$,
  −Perform least squares regression on B and $f(\boldsymbol{c}_k, \boldsymbol{\Phi}, A)$[a] to obtain $\hat{\boldsymbol{c}}_k$.
  Compute fitted continuation values $f(\hat{\boldsymbol{c}}_k, \boldsymbol{\Phi}, A)$.
  −Set $\hat{v}_{t_k}(\omega) = \Psi(\boldsymbol{Y}_{t_k}(\omega))$ if $\Psi(\boldsymbol{Y}_{t_k}(\omega)) > f(\hat{\boldsymbol{c}}_k, \boldsymbol{\Phi}, A)(\omega)$ for $\omega \in \Omega^*$.
End

$$\hat{v}_0 = \max\left\{ \Psi(\boldsymbol{Y}_0), \frac{1}{P}\sum_{i=1}^{P} D(0, t_1)\hat{v}_{t_1}(\omega_i) \right\}.$$

[a] $f(\boldsymbol{c}_k, \boldsymbol{\Phi}, A)(\omega) = \sum_{i=1}^{K} c_k^i \phi_i(\boldsymbol{Y}_{t_k}(\omega))\ \forall\, \boldsymbol{Y}_{t_k}(\omega) \in A,$

$\hat{v}_{t_k}(\omega) \approx \sum_{i=1}^{K} c_k^i \phi_i(\boldsymbol{Y}_{t_k}(\omega)) \qquad \forall \hat{v}_{t_k}(\omega) \in B,\ \boldsymbol{Y}_{t_k}(\omega) \in A.$

Chapter 5

Numerical Implementation

This chapter presents numerical results from pricing multi-asset American/ Bermudan options under the multi-asset Black-Scholes and multi-asset Heston models. Asset prices prices are generated through simulation. In particular, the multi-asset Black-Scholes asset prices are generated using the exact scheme presented in *Appendix A.1*, since an analytical solution of the Black-Scholes SDE exists. The multi-asset Heston model is simulated using MQE, which is a multi-dimensional extension of the QE scheme and it is presented in *Appendix A.2*.

In this dissertation, Black-Scholes asset prices are generated assuming the following parameters;

$$
\begin{aligned}
&S_i(0) = 100,\ r = 0.05,\ \sigma_i = 0.1,\ \eta_i = 0.0,\\
&\rho_{ij} = \begin{cases} 1 & i = j \\ 0.2 & i \neq j \end{cases},\\
&1 \leq i, j \leq d.
\end{aligned}
$$

The Heston asset prices are generated assuming that;

$$
\begin{aligned}
&S_i(0) = 100,\ v_i(0) = 0.1,\ r = 0.05,\ \kappa_i = 0.1,\ \theta_i = 0.15,\ \sigma_i = 0.1,\ \eta_i = 0.0,\\
&\rho_{ij}^{SS} = \begin{cases} 1 & i = j \\ 0.1 & i \neq j \end{cases},\ \rho_{ij}^{SV} = \begin{cases} 0.2 & i = j \\ 0 & i \neq j \end{cases},\ \rho_{ij}^{VV} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases},\\
&1 \leq i, j \leq d.
\end{aligned}
$$

The option pricing problem is considered in several dimensions, namely, $2-$, $5-$ and $10-$dimensions. The American/Bermudan options priced are the Geometric mean basket put option, Arithmetic mean basket put option and a Maximum call

option. These have the following payoff processes,

$$\text{Geometric mean basket put} : \Psi(\boldsymbol{S}_t) = \left[K - \left(\prod_{i=1}^{d} S_i(t)\right)^{\frac{1}{d}}\right]^+ \quad t \leq T, \tag{5.1}$$

$$\text{Arithmetic mean basket put} : \Psi(\boldsymbol{S}_t) = \left[K - \frac{1}{d}\sum_{i=1}^{d} S_i(t)\right]^+ \quad t \leq T, \tag{5.2}$$

$$\text{Max-Call} : \Psi(\boldsymbol{S}_t) = \left[\max_{1\leq i\leq d} S_i(t) - K\right]^+ \quad t \leq T. \tag{5.3}$$

Moreover, it is assumed that $K = 100$ and $T = 1$. In this dissertation, the accuracy and efficiency of GPR-MC is tested. The LSM algorithm is used as a benchmark against which GPR-MC is evaluated. LSM is implemented using $N = 10$ exercise times, with $P = 50000$ paths for each asset. GPR-MC on the other hand, is implemented using $N = 10$ exercise times, $P = 1000$ paths per asset, and $M = 200$ inner simulation paths at each node, (p, n), where $p \leq P,\ n \leq N$. N is quite small and as a result, the options priced are more Bermudan than American. This is a minor issue because N may be set as high as possible, taking computational power into account. As N increases, the Bermudan option price will approach, from below, the price of the American option equivalent. Both LSM and GPR-MC (assuming the Squared Exponential covariance function) were run 100 times to obtain mean option prices and 95% confidence intervals about the means. The average computational times of the 100 runs, as well as the minimum and maximum times are also reported. GPR-MC computational times include training times because GPR-MC is not a *fast derivative pricing scheme*[1]. The reader is encouraged to refer to section 3.2 for an example of a *fast derivative pricing scheme*. All computations were performed on MATLAB R2020a, on an Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80GHz 8.00 GB RAM laptop. All pricing results can be found in tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8.

5.1 Numerical Results

The accuracy of GPR-MC generally decreases as the dimension (d) increases, as can be seen from figures 5.1, 5.2 and 5.3. As the dimension increases, so does the relative error between the GPR-MC prices and LSM price (the benchmark). The GPR-

[1] Derivatives are priced using another scheme using a range of parameters. A GPR model is trained on these option prices (as output), with the range of parameters being the input. This GPR model is then stored for future use. The model may be used to price new options so long as the input parameters lie within the range of the parameters used to train the model. In this case, training time can be omitted.

d	*Geometric Mean Put*		*Arithmetic Mean Put*		*Max-Call*	
	Average Option Value	95% CI	Average Option Value	95% CI	Average Option Value	95% CI
2	1.6638	[1.6605;1.6672]	1.6245	[1.6218;1.6272]	10.4266	[10.4153;10.4379]
5	1.1263	[1.1242;1.1285]	1.0762	[1.0720;1.0805]	15.6393	[15.6276;15.6510]
10	0.9310	[0.9227;0.9393]	0.8764	[0.8685;0.8843]	19.3829	[19.3719;19.3938]

Tab. 5.1: LSM Average Option values and 95% confidence intervals for 100 runs, under the Black-Scholes model.

d	*Geometric Mean Put*		*Arithmetic Mean Put*		*Max-Call*	
	Average Option Value	95% CI	Average Option Value	95% CI	Average Option Value	95% CI
2	1.6754	[1.6533;1.6976]	1.6312	[1.6096;1.6528]	10.4429	[10.3955;10.4903]
5	1.1471	[1.1302;1.1639]	1.0855	[1.0693;1.1016]	15.5902	[15.5441;15.6363]
10	0.9554	[0.9399;0.9710]	0.9071	[0.8940;0.9202]	19.4186	[19.3716;19.4656]

Tab. 5.2: GPR-MC Average Option values and 95% confidence intervals for 100 runs, under the Black-Scholes model.

d	*Geometric Mean Put*		*Arithmetic Mean Put*		*Max-Call*	
	Average Option Value	95% CI	Average Option Value	95% CI	Average Option Value	95% CI
2	8.0909	[8.0778;8.1040]	7.4714	[7.4598;7.4829]	25.4797	[25.4361;25.5233]
5	6.0694	[6.0609;6.0779]	5.0226	[5.0153;5.0300]	44.7397	[44.6972;44.7822]
10	5.3736	[5.3656;5.3817]	4.1655	[4.1592;4.1717]	61.6032	[61.5565;61.6499]

Tab. 5.3: LSM Average Option values and 95% confidence intervals for 100 runs, under the Heston model.

MC 95% confidence intervals intersect those of LSM, at low dimensions ($d \leq 5$). Moreover, the true (i.e., average) option values, as obtained using LSM, generally lie below those obtained using GPR-MC. Furthermore, LSM mean option values lie within the GPR-MC 95% confidence bounds. Thus, GPR-MC is reasonably accu-

d	*Geometric Mean Put*		*Arithmetic Mean Put*		*Max-Call*	
	Average Option Value	95% CI	Average Option Value	95% CI	Average Option Value	95% CI
2	8.1974	[8.1305;8.2643]	7.5961	[7.5270;7.6652]	25.5024	[25.3485;25.6564]
5	6.1281	[6.0762;6.1800]	5.0817	[5.0353;5.1281]	44.4695	[44.3063;44.6327]
10	5.2077	[5.1682;5.2472]	4.0332	[3.9929;4.0735]	61.4504	[61.3174;61.5834]

Tab. 5.4: GPR-MC Average Option values and 95% confidence intervals for 100 runs, under the Heston model.

rate, notwithstanding the overestimated option prices.

GPR-MC prices that are obtained under the Black-Scholes model are generally more accurate than those under the Heston model. Under both models, Maximum call GPR-MC option prices are more accurate than those of the geometric mean basket and arithmetic mean basket options. Under the Black-Scholes model, the relative errors (for all options) lie within the interval $(0.15\%, 3.5\%)$ while under the Heston model, the errors lie within $(0.09\%, 3.5\%)$.

GPR-MC is less efficient to implement than LSM at low dimensions, $d \leq 5$, and is more efficient at higher dimensions, $d \geq 10$. There is also a huge variation in implementation times for GPR-MC than LSM, as depicted by the wider bounds for GPR-MC computational times compared to the narrower LSM bounds; see figures 5.4, 5.5 and 5.6.

As the dimension increases, there is no significant difference in the computational times of implementing GPR-MC. Thus, the efficiency of GPR-MC does not depend on the dimensionality of the pricing problem. On the other hand, LSM computational times increases with the dimensionality of the pricing problem. Moreover, LSM times under the Heston model are significantly greater than those under the Black-Scholes; GPR-MC computational times seem to be independent of the model choice as the Heston and Black-Scholes computational times are similar. At high dimensions, i.e., $d \geq 10$, LSM is significantly more inefficient.

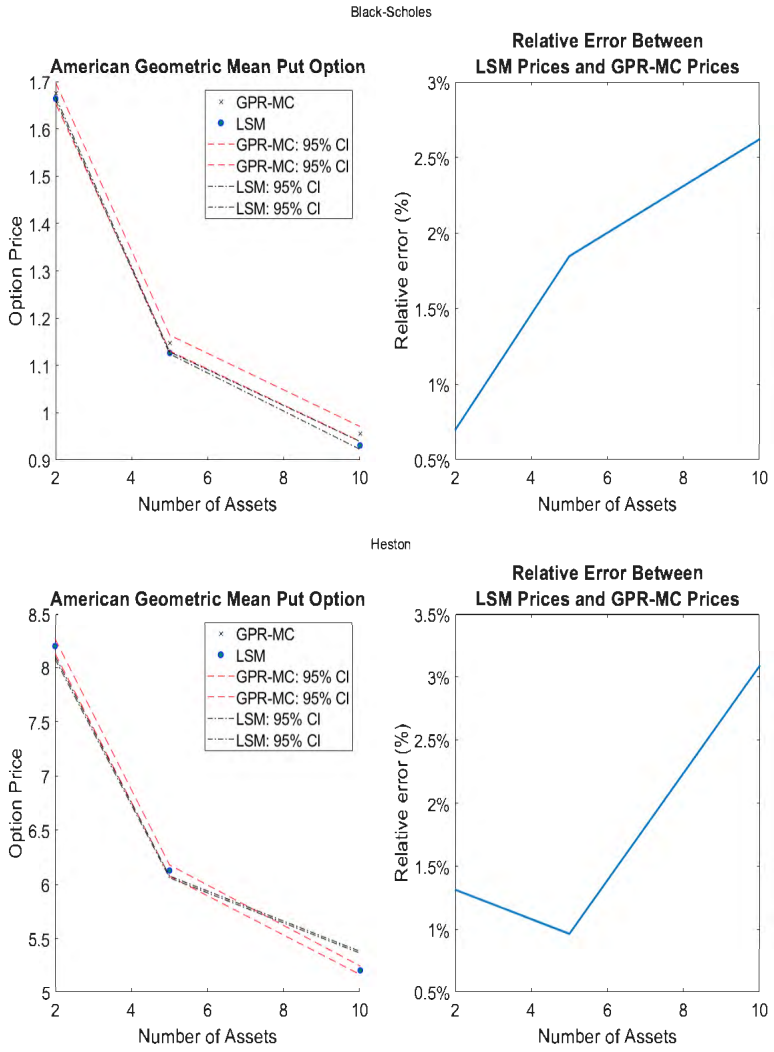


Fig. 5.1: [Left] GPR-MC & LSM American geometric mean basket put average option prices and 95% confidence intervals for 100 runs, under the Black-Scholes & Heston models. [Right] Relative error between average LSM & GPR-MC option prices.

5.2 Discussion

GPR-MC is less efficient than the LSM algorithm at dimensions $d \leq 5$, and more more efficient at higher dimensions $d \geq 10$. Similar to other Machine Learning algorithms, training the GPR model (for GPR-MC) to fit the data is slow. In particular, training the model involves choosing the covariance function's hyper-parameters which optimise the log-likelihood function. Making predictions using out-of-sample points involves matrix inversion. LSM on the other hand is a Least Squares Regres-

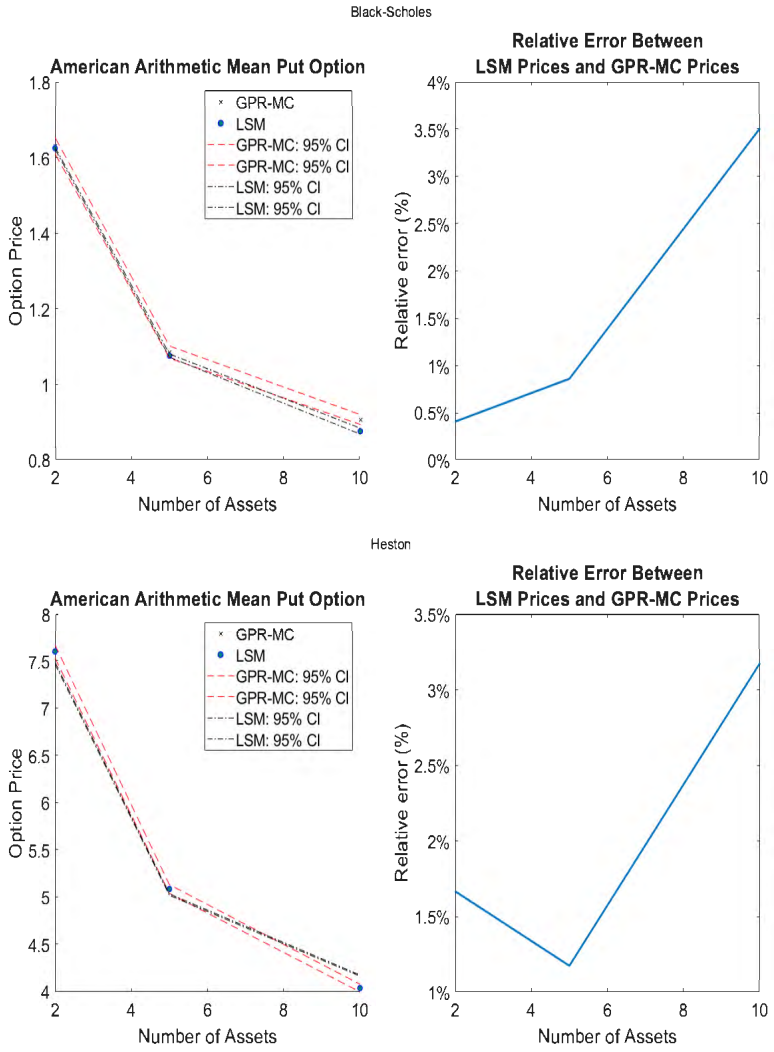


Fig. 5.2: [Left] GPR-MC & LSM American arithmetic mean basket put average option prices and 95% confidence intervals for 100 runs, under the Black-Scholes & Heston models. [Right] Relative error between average LSM & GPR-MC option prices.

sion method, which is more computationally efficient. Close form expressions for the regression coefficients which best fit the data exist. The only time consuming aspect is the matrix inversion involved in computing these optimal regression coefficients. As a result, the LSM algorithm is more efficient than GPR-MC at low dimensions since the number of basis functions is quite low and consequently, the complexity of the matrix inversion is low. At high dimensions, the number of basis functions becomes very large and hence the matrix inversion complexity increases significantly.

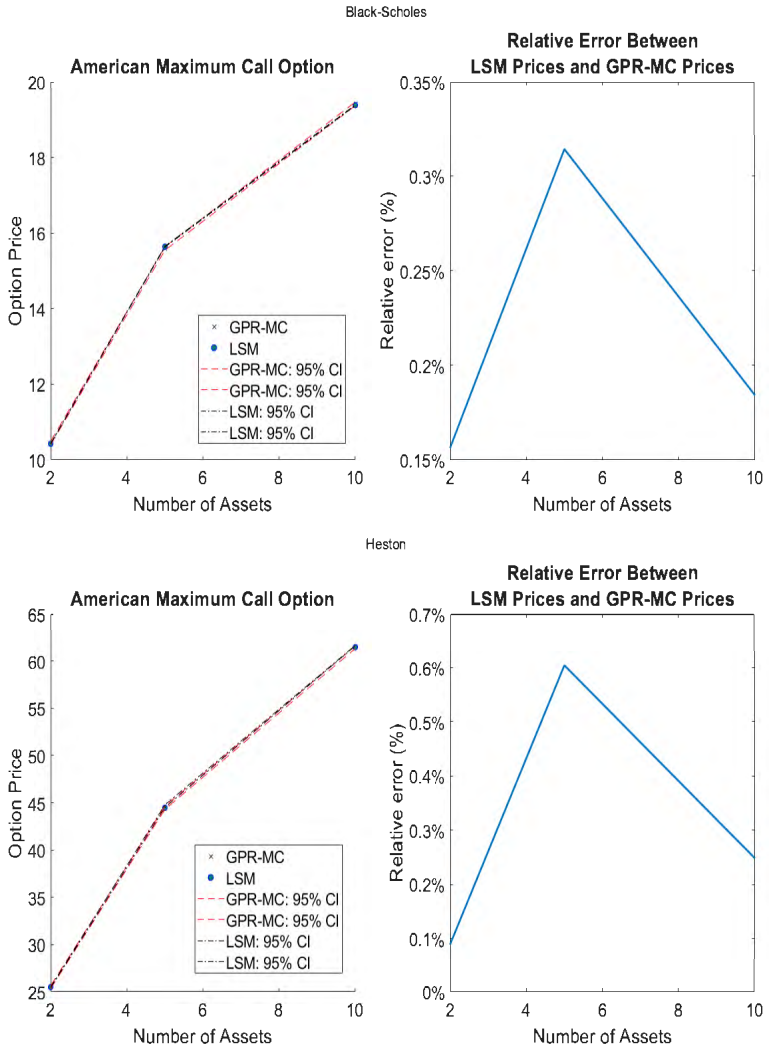


Fig. 5.3: [Left] GPR-MC & LSM American maximum-call average option prices and 95% confidence intervals for 100 runs, under the Black-Scholes & Heston models. [Right] Relative error between average LSM & GPR-MC option prices.

The efficiency of the LSM algorithm depends on the dimensionality of the pricing problem and the model assumptions of the underlying assets. This is expected because as the dimension increases, so does the number of basis functions. Moreover, more complex models which involve stochastic volatilities, for example, also increase the number of basis functions. Consequently, the matrix inversion becomes computationally expensive since the complexity of the matrix inversion increases quadratically with the number of basis functions. GPR-MC does not have the curse of dimensionality problem, in theory. The dimension of the pricing prob-

d	*Geometric Mean Put*		*Arithmetic Mean Put*		*Max-Call*	
	Average Option Value	Range	Average Option Value	Range	Average Option Value	Range
2	0.7859	[0.7129;0.9405]	0.7963	[0.7320;1.1937]	0.9289	[0.7518;1.1358]
5	5.1638	[5.0442;5.5637]	5.4411	[5.1241;7.0452]	6.9081	[6.4054;12.7634]
10	46.6414	[44.2643;48.0795]	48.6859	[46.0563;62.0447]	69.8186	[64.9996;87.5777]

Tab. 5.5: LSM Average computational times and range interval of computational times for 100 runs, under the Black-Scholes model.

d	*Geometric Mean Put*		*Arithmetic Mean Put*		*Max-Call*	
	Average Option Value	Range	Average Option Value	Range	Average Option Value	Range
2	1.6741	[1.4137;4.1607]	1.5701	[1.4335;3.8343]	1.9528	[1.6101;2.2841]
5	14.0464	[12.5234;20.7371]	13.4262	[12.7817;33.3177]	16.0055	[14.9732;17.5130]
10	112.7052	[102.1997;142.6500]	107.8830	[97.8944;247.8357]	140.1360	[138.1222;144.9202]

Tab. 5.6: LSM Average computational times and range interval of computational times for 100 runs, under the Heston model.

d	*Geometric Mean Put*		*Arithmetic Mean Put*		*Max-Call*	
	Average Option Value	Range	Average Option Value	Range	Average Option Value	Range
2	35.2895	[28.7075;44.8589]	36.5203	[30.9952;43.3989]	33.6524	[28.6928;43.4881]
5	30.9370	[26.7962;39.2828]	31.9381	[26.9357;37.5387]	30.9545	[27.3669;44.0950]
10	32.1939	[28.0057;40.5890]	33.3030	[28.1041;40.3732]	33.0228	[29.9454;42.6285]

Tab. 5.7: GPR-MC Average computational times and range interval of computational times for 100 runs, under the Black-Scholes model.

lem only affects the evaluation of the covariance function, in particular, the evaluation of the norm. However, this can be done efficiently (for the Euclidean norm at least) using matrix multiplication. In the prediction stage, the matrix inversion only depends on the number of sample points in the training set and not on the

d	*Geometric Mean Put*		*Arithmetic Mean Put*		*Max-Call*	
	Average Option Value	Range	Average Option Value	Range	Average Option Value	Range
2	42.0615	[34.3509;51.9134]	51.8272	[39.3495;67.4334]	33.8474	[27.2237;42.7694]
5	29.8773	[26.8145;36.4550]	33.5314	[27.3523;49.9522]	27.1987	[25.7347;33.2431]
10	30.9721	[28.4309;39.6971]	36.8621	[29.8590;47.0663]	30.8495	[28.7434;36.9337]

Tab. 5.8: GPR-MC Average computational times and range interval of computational times for 100 runs, under the Heston model.

dimension of the problem. More complex model assumptions increase the dimensionality, however this does not affect GPR-MC.

GPR-MC generally yields overestimated option prices, where the error increases with the dimension of the problem. Moreover, for a more complex model assumption, i.e., the Heston model, it generally yields even greater errors. However, the magnitude of the errors is reasonable for the Max-Call option, with relative errors not exceeding 0.7%. For basket options, the errors are significant, with relative errors in the range $(0.4\%, 2\%)$ at low dimensions ($d \leq 5$) and slightly exceeding 3% at high dimensions ($d \geq 10$). Since these errors are less than 5%, this dissertation deems them reasonable.

This dissertation implements GPR-MC using a probabilistic grid, as opposed to a space-filling grid used by Goudenège *et al.* (2019). Therefore, there is a non-zero probability that there exist points in the prediction set (i.e. the inner simulation) whose values exceed those in the training set (i.e., the outer simulation) resulting in poor predictions. Unlike *fast derivative pricing*, where the prediction set can be chosen such that its values lie within the bounds of the training set, this is not guaranteed by GPR-MC. In GPR-MC, the training and prediction sets are generated using simulation schemes, thus introducing randomness in the sets. Consequently, there is a possibility that some points in the prediction set may lie beyond the bounds of the training set, as a matter of probability. The reader should refer to section 3.2 to observe the prediction accuracy of points that lie beyond the bounds of the training set.

It is important for the reader to note that the errors in the prices might also be due to a poor choice of a benchmarking algorithm. The LSM algorithm depends on the choice of basis functions, thus a different choice of basis function from those chosen in this paper may yield different results. Moreover, GPR-MC depends on P

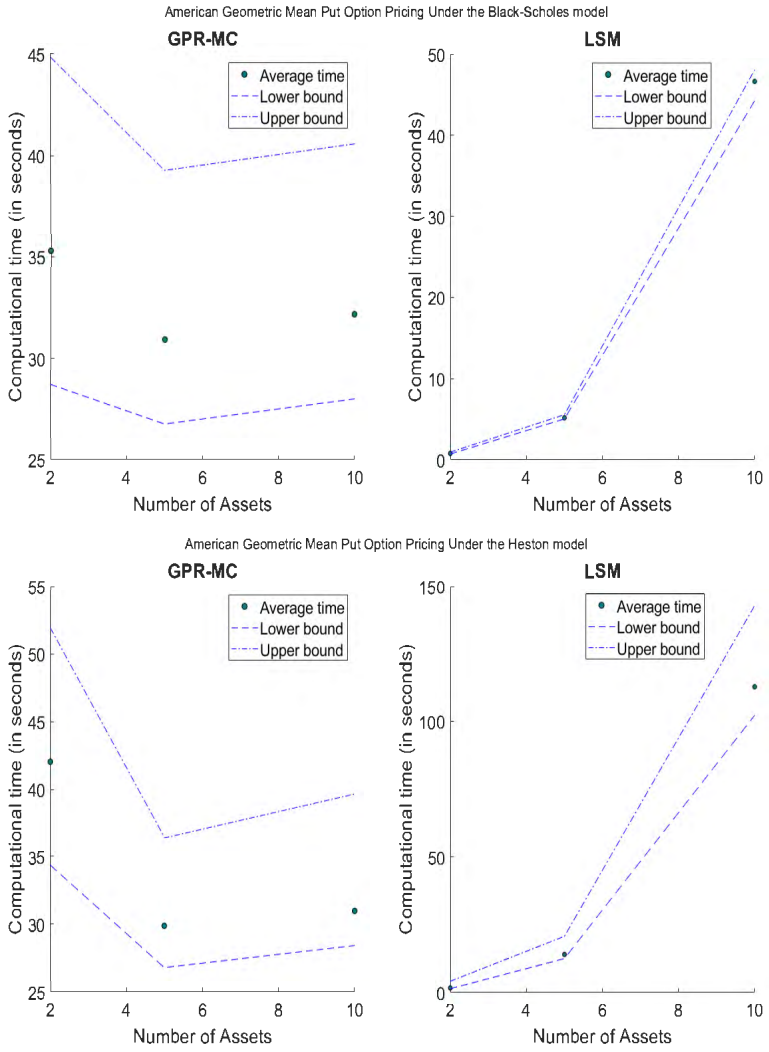


Fig. 5.4: American geometric mean basket put: GPR-MC & LSM Average computational times and range-interval of computational times for 100 runs.

and M, which are the number of paths for the outer simulation and inner simulation respectively. Due to memory considerations, this dissertation chose $P = 1000$ and $M = 200$, which may explain the accuracy achieved. Goudenège *et al.* (2019) used low discrepancy pseudo-random numbers and this resulted in more accurate prices, when benchmarked against those obtained using Monte Carlo simulation. This dissertation implements GPR-MC using random numbers and this allows for the generation of 95% confidence interval bounds.

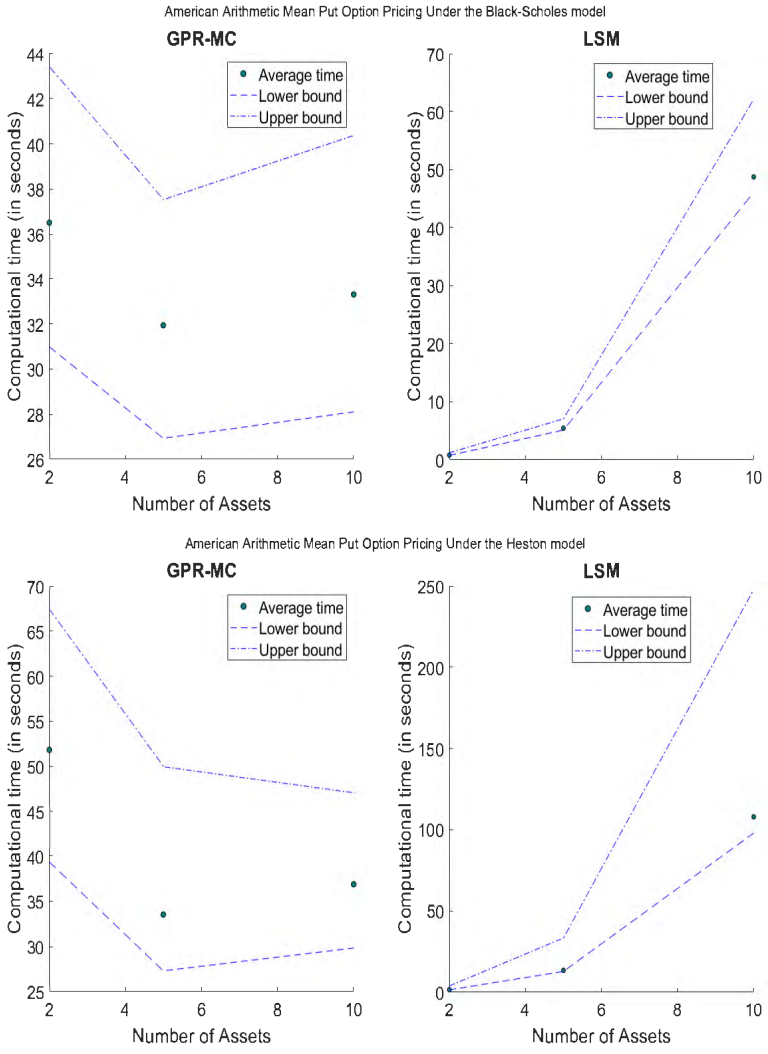


Fig. 5.5: American arithmetic mean basket put: GPR-MC & LSM Average computational times and range-interval of computational times for 100 runs.

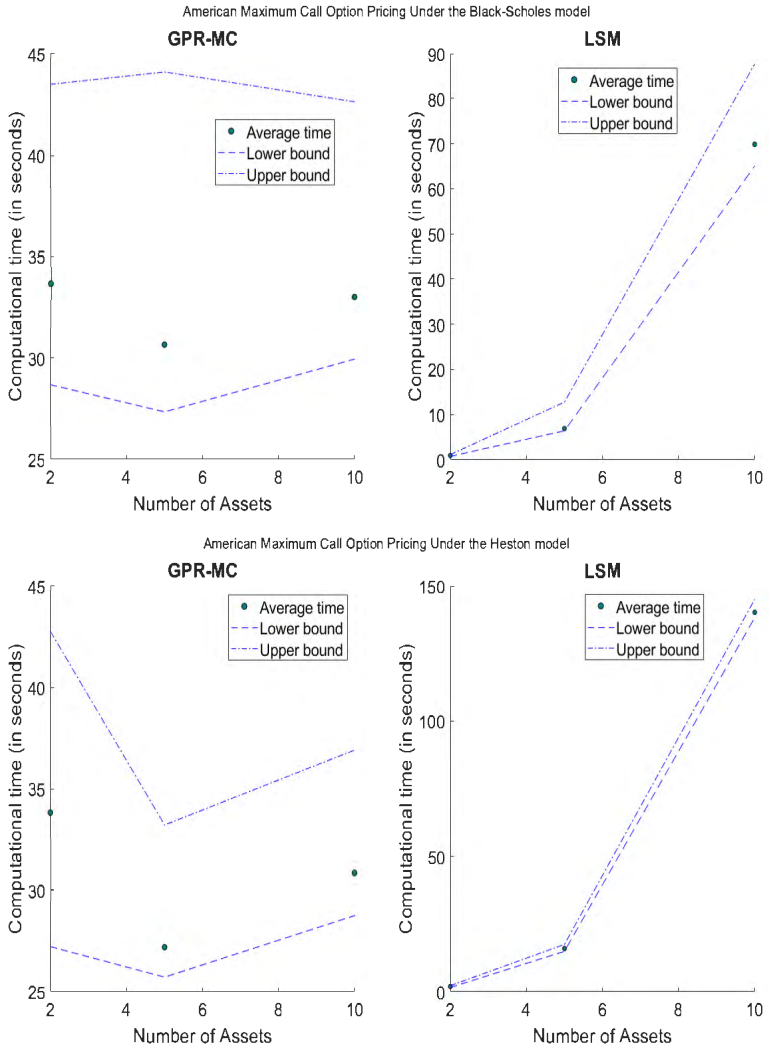


Fig. 5.6: American maximum-call: GPR-MC & LSM Average computational times and range-interval of computational times for 100 runs.

Chapter 6

Conclusions and Further Research

This dissertation explored the use of GPR-MC to price American options in high dimension. To this effect, options of various dimensions or number of underlying assets were priced using GPR-MC and LSM, with the latter scheme as the benchmark. The effect of the dimensionality of the pricing problem on the accuracy and efficiency of GPR-MC was studied. Simulated market data, i.e., stock prices (and stochastic volatilities) were used for pricing, under the multi-asset Black-Scholes and Heston models. In the simulation schemes, random numbers as opposed to low discrepancy sequences (and hence pseudo-random numbers) such as in Goudenège *et al.* (2019), were used. This may have contributed to the relative errors between GPR-MC option prices and LSM option prices. However, the use of random number enabled the generation of 95% confidence intervals.

The GPR-MC algorithm was found to be more efficient than the LSM algorithm at higher dimensions, i.e., $d \geq 10$. The computation times of GPR-MC included training times, given the nature of the algorithm and how it differs from *fast derivative pricing schemes*. Compared to LSM, it was shown that GPR-MC's computational times were independent of the number of underlying assets and the complexity of the model assumptions. LSM's computational times were observed to increase with the number of underlyings.

In this study, GPR-MC was found to overestimate the option price, with relative errors increasing with number of underlying assets. However, the magnitude of these errors was found to be reasonable.

Taking into account the accuracy and efficiency insights, it is evident that GPR-MC may be useful for pricing options in very high dimensions, were schemes such LSM may be infeasible to use due to the number of basis functions required. Moreover, at high dimensions, GPR-MC will be easier to implement as one has to only specify a covariance function. It is important for the reader to note that at very high dimensions, memory usage will be a significant constraint even for GPR-MC.

A possible extension of this dissertation is on improving the accuracy of the

GPR-MC algorithm. Firstly, the number of outer simulation paths, i.e., P, and the number of inner simulations, i.e., M, may be increased but computational cost will be incurred. Secondly, since GPR produces poor predictions for out-of-sample inputs, it may be worthwhile to include a check within the GPR-MC algorithm to ensure that all inner simulations at step n lie within the bounds of the outer simulations at step $n+1$. Any inner simulations that lie beyond the outer simulation bounds could be discarded and re-simulated until they lie within the bounds of the outer simulations. Lastly, GPR-MC may be benchmarked against other algorithms such as the Stochastic Grid Bundling Method (SGBM) of Jain and Oosterlee (2012) in addition to LSM, to reduce the possibility of a benchmarking bias in the results.

Bibliography

Abbas-Turki, L. and Lapeyre, B. (2011). American options based on Malliavin calculus and nonparametric variance reduction methods, *arXiv preprint arXiv:1104.5131* .

Andersen, L. B. (2007). Efficient simulation of the Heston stochastic volatility model, *Available at SSRN 946405* .

Björk, T. (2009). *Arbitrage theory in continuous time*, Oxford university press.

Broadie, M. and Glasserman, P. (1997). Pricing American-style securities using simulation, *Journal of economic dynamics and control* **21**(8-9): 1323–1352.

Carriere, J. F. (1996). Valuation of the early-exercise price for options using simulations and nonparametric regression, *Insurance: mathematics and Economics* **19**(1): 19–30.

Cox, J. C., Ingersoll Jr, J. E. and Ross, S. A. (2005). A theory of the term structure of interest rates, *Theory of valuation*, World Scientific, pp. 129–164.

Crépey, S. and Dixon, M. (2019). Gaussian process regression for derivative portfolio modeling and application to cva computations, *arXiv preprint arXiv:1901.11081* .

de Innocentis, M., Lichters, R. and Trahe, M. (2016). Efficient simulation of the multi- asset Heston model, *Available at SSRN 2729475* .

De Spiegeleer, J., Madan, D. B., Reyners, S. and Schoutens, W. (2018). Machine learning for quantitative finance: fast derivative pricing, hedging and fitting, *Quantitative Finance* **18**(10): 1635–1643.

Glasserman, P. (2013). *Monte Carlo methods in financial engineering*, Vol. 53, Springer Science & Business Media.

Gonzalvez, J., Lezmi, E., Roncalli, T. and Xu, J. (2019). Financial applications of Gaussian processes and Bayesian optimization, *arXiv preprint arXiv:1903.04841* .

Goudenège, L., Molent, A. and Zanette, A. (2019). Variance reduction applied to machine learning for pricing Bermudan/ American options in high dimension, *arXiv preprint arXiv:1903.11275* .

Goudenège, L., Molent, A. and Zanette, A. (2020a). Gaussian process regression for pricing variable annuities with stochastic volatility and interest rate, *Decisions in Economics and Finance* pp. 1–16.

Goudenège, L., Molent, A. and Zanette, A. (2020b). Machine learning for pricing American options in high-dimensional Markovian and non-Markovian models, *Quantitative Finance* **20**(4): 573–591.

Harrison, J. M. and Kreps, D. M. (1979). Martingales and arbitrage in multiperiod securities markets, *Journal of Economic theory* **20**(3): 381–408.

Haugh, M. B. and Kogan, L. (2004). Pricing American options: a duality approach, *Operations Research* **52**(2): 258–270.

Herfurth, H. (2020). *Gaussian process regression in computational finance*, Master's thesis, Uppsala University, Uppsala.

Jain, S. and Oosterlee, C. W. (2012). Pricing high-dimensional Bermudan options using the stochastic grid method, *International Journal of Computer Mathematics* **89**(9): 1186–1211.

Longstaff, F. A. and Schwartz, E. S. (2001). Valuing American options by simulation: a simple least-squares approach, *The review of financial studies* **14**(1): 113–147.

Ludkovski, M. (2018). Kriging metamodels and experimental design for Bermudan option pricing, *Journal of Computational Finance* **22**(1).

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, USA.

Shen, Y. (2014). *Credit Value Adjustment for Multi-Asset Options*, PhD thesis, Delft University of Technology, Delft.

Szimayer, A., Dimitroff, G. and Lorenz, S. (2009). A parsimonious multi-asset Heston model: calibration and derivative pricing.

Tegnér, M., Roberts, S. and Oxford, O. (2017). A Bayesian take on option pricing with Gaussian processes, *Technical report*, Working paper.

Tong, Y. L. (2012). *The multivariate normal distribution*, Springer Science & Business Media.

Tsitsiklis, J. N. and Van Roy, B. (1999). Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives, *IEEE Transactions on Automatic Control* **44**(10): 1840–1851.

Wadman, W. S. (2010). *An advanced Monte Carlo method for the multi-asset heston model*, Master's thesis, Delft University of Technology, Delft.

Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian processes for regression, *Advances in Neural Information Processing Systems 8*, MIT press, pp. 514–520.

Appendix A

Simulation

In this section, schemes for simulating the multi-asset Black-Scholes and Heston models are presented. The proofs of the following results are omitted and can be found in Tong (2012).

Theorem A.1 (Cholesky Decomposition). *If $\boldsymbol{\Sigma} = (\sigma_{ij})$ is an $n \times n$ positive definite matrix, then there exist a unique lower triangular matrix $\boldsymbol{L} = (l_{ij})$ satisfying $\boldsymbol{L}\boldsymbol{L}^\top = \boldsymbol{\Sigma}$. Furthermore, the elements of $\boldsymbol{L}$ are given by*

$$
\begin{aligned}
&l_{ij} = 0 \qquad \text{for all } 1 \le i < j \le n,\\
&l_{11} = \sqrt{\sigma_{11}} \qquad \text{for } i = 2, ..., n,\\
&l_{jj} = \left(\sigma_{jj} - \sum_{r=1}^{j-1} l_{jr}^2\right)^{\frac{1}{2}} \qquad \text{for } j = 2, ..., n,\\
&l_{ij} = \frac{1}{l_{jj}}\left(\sigma_{ij} - \sum_{r=1}^{j-1} l_{ir} l_{jr}\right) \qquad \text{for } j < i \text{ and } i = 2, ..., n-1
\end{aligned}
$$

.

Theorem A.2. *Let $\boldsymbol{Y} := \boldsymbol{C}\boldsymbol{X} + \boldsymbol{b}$ where $\boldsymbol{X} \sim \mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$[1] and $\boldsymbol{C}$ is an $n \times n$ real matrix such that $det(\boldsymbol{C}) \neq 0$, then $\boldsymbol{Y} \sim \mathcal{N}_n\left(\boldsymbol{C}\boldsymbol{\mu} + \boldsymbol{b}, \boldsymbol{C}\boldsymbol{\Sigma}\boldsymbol{C}^\top\right)$.*

A.1 Multi-Asset Black-Scholes Model

Under the risk-neutral measure (or $\mathbb{Q}$-measure), the multi-asset Black-Scholes model has the following dynamics

$$
\begin{aligned}
&dS_i(t) = (r - \eta_i)S_i(t)\, dt + \sigma_i S_i(t)\, dW_i(t) \qquad i = 1, ..., d,\\
&d\langle W_i, W_j\rangle_t = \rho_{ij}\, dt, \quad |\rho_{ij}| \le 1.
\end{aligned} \tag{A.1}
$$

Eqn. (A.1) has the following analytical solution:

$$
S_i(t) = S_i(0) \exp\left(\left(r - \eta_i - \frac{1}{2}\sigma_i^2\right) t + \sigma_i W_i(t)\right). \tag{A.2}
$$

[1] $\boldsymbol{X}$ has an n-dimensional multivariate normal distribution with mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$ and a positive definite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$.

Equivalently,

$$S_i(t) = S_i(0)\exp\left(\left(r - \eta_i - \frac{1}{2}\sigma_i^2\right)t + \sigma_i\sqrt{t}\tilde{Z}_i\right) \qquad \text{where } \tilde{Z}_i \sim N(0,1) \qquad \text{(A.3)}$$

with $\mathbb{C}orr(\tilde{Z}_i, \tilde{Z}_j) = \rho_{ij}$ for $1 \leq i,\ j \leq d$.
Let

$$\boldsymbol{\Gamma} := \begin{bmatrix} 1 & \rho_{12} & \dots & \rho_{1d} \\ \rho_{21} & 1 & \dots & \rho_{2d} \\ \vdots & & \ddots & \vdots \\ \rho_{d1} & \dots & & 1 \end{bmatrix}.$$

Suppose that $\boldsymbol{L}$ is the Cholesky decomposition of $\boldsymbol{\Gamma}$, i.e., $\boldsymbol{\Gamma} = \boldsymbol{L}\boldsymbol{L}^\top$. Moreover, assume that $\boldsymbol{Z} \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{I}_d)$, with mean vector $\mathbf{0}^\top = (0, ..., 0)^\top$ and covariance matrix $\boldsymbol{I}_d$; the $d \times d$ identity matrix. It then follows by Theorem A.2 that $\boldsymbol{L}\boldsymbol{Z} \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{\Gamma})$. This implies the following simulation scheme:

1. Sample $\boldsymbol{Z}$ from $\mathcal{N}_d(\mathbf{0},\ \boldsymbol{I}_d)$.
2. Compute $\boldsymbol{L}$, the Cholesky decomposition of $\boldsymbol{\Gamma}$.
3. Compute $\tilde{\boldsymbol{Z}} := \boldsymbol{L}\boldsymbol{Z}$.

   Note that $\tilde{\boldsymbol{Z}} = \begin{bmatrix} \tilde{Z}_1 \\ \vdots \\ \tilde{Z}_d \end{bmatrix}$.
4. $S_i(t) = S_i(0)\exp\left(\left(r - \eta_i - \frac{1}{2}\sigma_i^2\right)t + \sigma_i\sqrt{t}\tilde{Z}_i\right)$, $1 \leq i \leq d$, is a realisation of the i^{th} asset at time t.
5. To create a path over a time horizon $[0, T]$, proceed as follows
   - Partition the time horizon into $m+1$ time points $0 = t_0 < t_1 < ... < t_m = T$.
   - Sample $\boldsymbol{Z}_1,\ \boldsymbol{Z}_2,\ ...,\ \boldsymbol{Z}_m$ from $\mathcal{N}_d \sim (\mathbf{0}, \boldsymbol{I}_d)$.
   - Compute $\tilde{\boldsymbol{Z}}_1 := \boldsymbol{L}\boldsymbol{Z}_1,\ \tilde{\boldsymbol{Z}}_2 := \boldsymbol{L}\boldsymbol{Z}_2,\ ...,\ \tilde{\boldsymbol{Z}}_m := \boldsymbol{L}\boldsymbol{Z}_m$. It follows that $\tilde{\boldsymbol{Z}}_j = \begin{bmatrix} \tilde{Z}_{j,1} \\ \vdots \\ \tilde{Z}_{j,d} \end{bmatrix}$, $1 \leq j \leq m$.
   - Then, generate a path on time horizon $[0, T]$ using the following recursive relation:

$$\begin{aligned} S_i(t_{n+1}) =& S_i(t_n)\exp\left(\left(r - \eta_i - \frac{1}{2}\sigma_i^2\right)\Delta t_n + \sigma_i\sqrt{\Delta t_n}\tilde{Z}_{i,n+1}\right), \qquad \text{(A.4)} \\ & \Delta t_n := t_{n+1} - t_n,\ 0 = t_0 < t_1 < ... < t_m = T, \\ & n = 0, ..., m-1, \\ & i = 0, ..., d. \end{aligned}$$

Equation (A.4) is the exact discretisation of equation (A.1). Figure A.1 depicts a simulated path of a 10-dimensional Black-Scholes model.

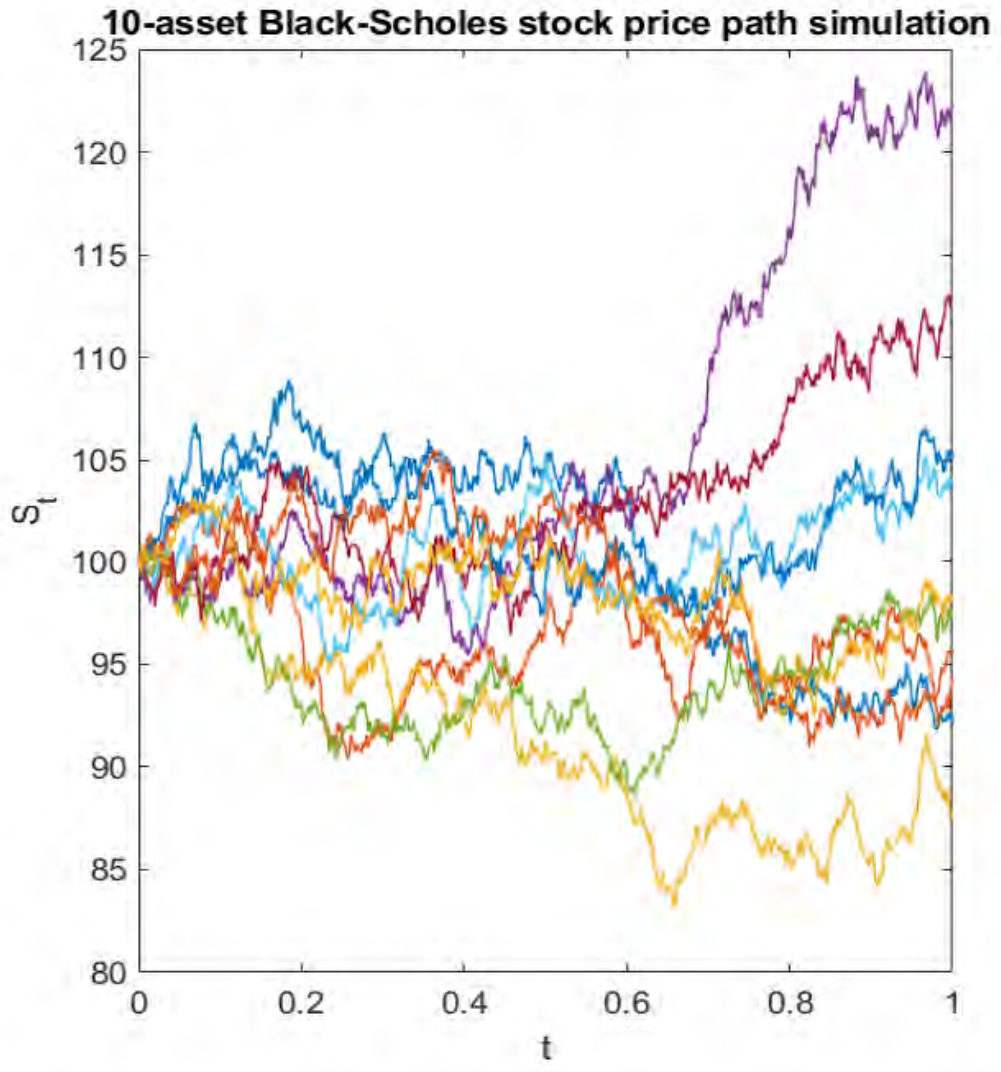


Fig. A.1: 10-dimensional Black Scholes stock price path simulation. ($S_i(0) = 100,\ \sigma_i = 0.1,\ \eta_i = 0.0,\ \rho_{ij} = 0.2\ (i \neq j),\ r = 0.05,\ N = 1000$ time steps)

A.2 Multi-Asset Heston Model

Suppose $(S_i)_{t\geq 0}$ asset price process follows the Heston dynamics

$$dS_i(t) = (r - \eta_i)S_i(t)\,dt + \sqrt{v_i(t)}S_i(t)\,dW_i(t), \tag{A.5}$$

$$dv_i(t) = \kappa_i(\theta_i - v_i(t))\,dt + \sigma_i\sqrt{v_i(t)}\,d\tilde{W}_i(t), \tag{A.6}$$

such that

$$
\begin{aligned}
&d\langle W_i, W_j\rangle_t = \rho_{ij}^{SS} dt,\\
&d\langle W_i, \tilde{W}_j\rangle_t = \rho_{ij}^{SV} dt\\
&d\langle \tilde{W}_i, \tilde{W}_j\rangle_t = \rho_{ij}^{VV} dt,\\
&\theta_i, \kappa_i, \sigma_i > 0,\\
&\rho_{ij}^{SS}, \rho_{ij}^{SV}, \rho_{ij}^{VV} \in [-1, 1],
\end{aligned}
$$

for $1 \leq i, j \leq d$.
Unlike the Black-Scholes model, the SDEs representing the Heston model have no analytical solution. In the single asset case, various numerical schemes have been developed. In the multi-asset case, there is very little literature on the efficient simulation of the multi-asset Heston model with a general correlation structure (de Innocentis *et al.*, 2016). Szimayer *et al.* (2009) proposed a parsimonious multi-asset Heston model with a restricted correlation structure, and then they used Euler discretisation to simulate the model. Wadman (2010) and de Innocentis *et al.* (2016) developed the Multi-Quadratic Exponential (MQE) scheme which is a multi-asset extension of the Quadratic Exponential (QE) scheme by Andersen (2007). In this paper, the MQE scheme as developed in Wadman (2010) is used. As such, this paper also assumes that

$$\rho_{ij}^{VV} = \begin{cases} 1 & i = j, \\ 0 & i \neq j \end{cases}, \tag{A.7}$$

$$\rho_{ij}^{SV} = \begin{cases} \rho_i & i = j, \\ 0 & i \neq j \end{cases}. \tag{A.8}$$

No restrictions are imposed on $\rho_{ij}^{SS}\ \forall\ i, j$. Let $X_i(t) := \log(S_i(t))$, then by Itô's formula, equations (A.5) and A.6 become

$$dX_i(t) = \left(r - \eta_i - \frac{1}{2} v_i(t)\right)\ dt + \sqrt{v_i(t)}\ dW_i(t), \tag{A.9}$$

$$dv_i(t) = \kappa_i\left(\theta_i - v_i(t)\right)\ dt + \sigma_i \sqrt{v_i(t)}\ d\tilde{W}_i(t), \tag{A.10}$$

The correlation structure of the model implies that $(v_i(t))_{t\geq 0}\ \forall i$ can be simulated independently. Equation (A.10) is a mean-reverting square-root process whose dynamics are the same as the CIR model[2]. Its simulation depends on the following result found in Cox *et al.* (2005):

Theorem A.3. *Let $\Delta > 0$ and $F_{\chi^2}(x, \delta, \lambda)$ be the cumulative distribution for the non-central chi-squared distribution with δ degrees of freedom and non-centrality parameter λ. Then*

$$\mathbb{P}\left[v_i(t+\Delta) < x | v_i(t)\right] = F_{\chi^2}\left(\frac{n_i(t,\Delta)x}{\exp(-\kappa_i\Delta)}, d_i, n_i(t,\Delta)v_i(t)\right), \tag{A.11}$$

with $d_i = \frac{4\kappa_i\theta_i}{\sigma_i^2}$ and $n_i(t,\Delta) = \frac{4\kappa_i \exp(-\kappa_i\Delta)}{\sigma_i^2(1-\exp(-\kappa_i\Delta))}$

[2] Cox-Ingersoll-Ross model

Theorem A.3 implies that

$$\frac{n_i(t,\Delta)}{\exp(-\kappa_i\Delta)}v_i(t+\Delta) \sim \chi^2\left(d_i, n_i(t,\Delta)v_i(t)\right), \tag{A.12}$$

where Z is a standard normal random variable. Sampling from this distribution is rather time consuming (de Innocentis *et al.*, 2016). To make simulation more efficient, Andersen (2007) proposed the following approximations;

$$v_i(t+\Delta) \sim a_i(b_i+Z)^2, \tag{A.13}$$

for high non-centrality parameters, and sampling from the distribution

$$G(x,p_i,\beta_i) = \mathbb{P}\left[v_i(t+\Delta)\leq x\right] = p_i + (1-p_i)(1-\exp(-\beta_i x)), \tag{A.14}$$

whose inverse distribution is

$$v_i(t+\Delta) = G^{-1}(U,p_i,\beta_i) = \begin{cases} 0, & \text{if } 0\leq U\leq p_i, \\ \beta_i^{-1}\log\left(\frac{1-p_i}{1-U}\right), & \text{if } p_i\leq U\leq 1, \end{cases} \tag{A.15}$$

where $U\sim Uniform(0,1)$, for small non-centrality parameters (de Innocentis *et al.*, 2016). The expressions for $a_i,\ b_i,\ \beta_i,\ p_i\ (\forall i=1,...,d)$ are computed using moment matching (de Innocentis *et al.*, 2016) and said expressions are found in *Appendix B*. Let $t_{n+1} := t_n + \Delta$. Given $v_i(t_n)\ \forall i=1,...,d$, simulate $v_i(t_{n+1})$ as follows;

1. Let
$$\Psi_i := \frac{Var\left[v_i(t_{n+1})|v_i(t_n)\right]}{\mathbb{E}\left[v_i(t_{n+1})|v_i(t_n)\right]^2}\ \text{[3]} \quad \text{for } i=1,...,d. \tag{A.16}$$

2. If $\Psi_i\leq\Psi_c$ for Ψ_c[4], then use equation (A.15). If $\Psi_i\geq\Psi_c$, then use equation (A.13) for $i=1,...,d$.

Let

$$\boldsymbol{W}_{XV} := \begin{bmatrix} \tilde{W}_1 \\ \vdots \\ \tilde{W}_d \\ W_1 \\ \vdots \\ W_d \end{bmatrix},$$

be a $2d$ Brownian motion, with correlation matrix $\boldsymbol{\Sigma}\in\mathbb{R}^{2d\times 2d}$. Then our assumptions yield

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{I}_d & \boldsymbol{0} \\ \boldsymbol{\Sigma}_{XV} & \boldsymbol{\Sigma}_X \end{bmatrix},$$

[3] Refer to *Appendix B* for analytical expressions
[4] Note that $\Psi_c\in[1,2]$. Andersen (2007) used $\Psi_c = 1.5$

where $\boldsymbol{I}_d$ is the $d \times d$ identity matrix, $\boldsymbol{\Sigma}_{XV} = diag(\rho_1, ..., \rho_d)$ is the variance-stock correlation matrix, and $\boldsymbol{\Sigma}_X$ is the $d \times d$ stock-stock correlation matrix. It is shown in Wadman (2010) that the Cholesky decomposition of $\boldsymbol{\Sigma}$ is $\boldsymbol{L}\boldsymbol{L}^\top$, where

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{I}_d & \boldsymbol{0} \\ \boldsymbol{\Sigma}_{XV} & \boldsymbol{L}^* \end{bmatrix}, \tag{A.17}$$

where $\boldsymbol{0}$ is the $d \times d$ zero matrix and $\boldsymbol{L}^*$ is a $d \times d$ lower triangular matrix.
Let $\bar{\boldsymbol{B}}^\top = \left(\bar{B}_1, \bar{B}_2, ..., \bar{B}_{2d}\right)$ be $2d$-dimensional standard Brownian motion with independent components. Set

$$d\tilde{\boldsymbol{W}}(t) := \begin{bmatrix} d\tilde{W}_1(t) \\ \vdots \\ d\tilde{W}_d(t) \end{bmatrix} = \begin{bmatrix} d\bar{B}_1(t) \\ \vdots \\ d\bar{B}_d(t) \end{bmatrix}, \tag{A.18}$$

$$d\boldsymbol{W}(t) = \begin{bmatrix} dW_1(t) \\ \vdots \\ dW_d(t) \end{bmatrix} = \boldsymbol{\Sigma}_{XV}\, d\tilde{\boldsymbol{W}}(t) + \boldsymbol{L}^* \begin{bmatrix} d\bar{B}_{d+1}(t) \\ \vdots \\ d\bar{B}_{2d}(t) \end{bmatrix}. \tag{A.19}$$

Clearly,

$$d\boldsymbol{W}_{XV}(t) = \begin{bmatrix} d\tilde{\boldsymbol{W}}(t) \\ d\boldsymbol{W}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_d & \boldsymbol{0} \\ \boldsymbol{\Sigma}_{XV} & \boldsymbol{L}^* \end{bmatrix} d\bar{\boldsymbol{B}}(t) = \boldsymbol{L}\, d\bar{\boldsymbol{B}}(t),$$

which imposes the desire correlation structure. It follows that (i^{th} row of $d\boldsymbol{W}(t)$)

$$dW_i(t) = \rho_i\, d\bar{B}_i(t) + \sum_{j=1}^{i} L^*_{ij}\, d\bar{B}_{d+j} = \rho_i\, d\tilde{W}_i(t) + \sum_{j=1}^{i} L^*_{ij}\, d\bar{B}_{d+j}. \tag{A.20}$$

Substitute equation (A.20) in equation (A.9) and the express in integral form to obtain

$$\begin{aligned} X_i(t_{n+1}) =& X_i(t_n) + \int_{t_n}^{t_{n+1}} r - \eta_i - \frac{1}{2} v_i(s)\, ds + \rho_i \int_{t_n}^{t_{n+1}} \sqrt{v_i(s)}\, d\tilde{W}_i(s) \\ &+ \sum_{j=1}^{i} L^*_{ij} \int_{t_n}^{t_{n+1}} \sqrt{v_i(s)}\, d\bar{B}_{d+j}. \end{aligned} \tag{A.21}$$

Equation (A.10) can be rearranged and expressed in integral form to show that

$$\int_{t_n}^{t_{n+1}} \sqrt{v_i(s)}\, d\tilde{W}_i(s) = \sigma_i^{-1} \left(v_i(t_{n+1}) - v_i(t_n) - \kappa_i \theta_i \Delta + \kappa_i \int_{t_n}^{t_{n+1}} v_i(s)\, ds \right) \forall i = 1, ..., d. \tag{A.22}$$

Moreover,

$$\int_{t_n}^{t_{n+1}} v_i(s)\, ds \approx \Delta \left[\gamma_1 v_i(t_n) + \gamma_2 v_i(t_{n+1}) \right], \tag{A.23}$$

where $\gamma_1 = \gamma_2 = \frac{1}{2}$ (Wadman, 2010). Finally, using Itô's Isometry (and normality of Itô's integrals)

$$\sum_{j=1}^{i} L^*_{ij} \int_{t_n}^{t_{n+1}} \sqrt{v_i(s)}\, d\bar{B}_{d+j} = \sum_{j=1}^{i} L^*_{ij} \cdot Z_{j,n+1} \sqrt{\int_{t_n}^{t_{n+1}} v_i(s)\, ds}, \tag{A.24}$$

where $Z_{j,n+1} \sim \mathcal{N}(0,1)\ \forall 1 \leq j \leq i \leq d$. Define

$$f_i := \sigma_i^{-1}\left(v_i(t_{n+1}) - v_i(t_n) - \kappa_i\theta_i\Delta + \kappa_i \int_{t_n}^{t_{n+1}} v_i(s)\, ds\right),$$
$$s_i := \sqrt{\int_{t_n}^{t_{n+1}} v_i(s)\, ds}.$$

Then,

$$X_i(t_{n+1}) = X_i(t_n) + (r - \eta_i)\Delta - \frac{1}{2}\int_{t_n}^{t_{n+1}} v_i(s)\, ds + \rho_i f_i + s_i \sum_{j=1}^{i} L_{ij}^* \cdot Z_{j,n+1}. \tag{A.25}$$

MQE scheme

Let $t_{n+1} := t_n + \Delta$. Given $v_i(t_n)\ \forall i = 1, ..., d$, simulate $v_i(t_{n+1})$ as follows;

1. Let
$$\Psi_i := \frac{Var\left[v_i(t_{n+1})|v_i(t_n)\right]}{\mathbb{E}\left[v_i(t_{n+1})|v_i(t_n)\right]^2} \quad \text{for } i = 1, ..., d. \tag{A.26}$$
2. If $\Psi_i \leq \Psi_c$ for Ψ_c, then use equation (A.15). If $\Psi_i \geq \Psi_c$, then use equation (A.13) for $i = 1, ..., d$.

Now, given both $v_i(t_n)$ & $v_i(t_n)$, and $S_i(t_n)$, simulate $S_i(t_{n+1})$ as follows ($\forall i = 1, ..., d$);

1. Compute $X_i(t_n) = \log(S_i(t_n))$.
2. Compute equation (A.23).
3. Sample $\boldsymbol{Z}_{n+1}^\top = (Z_{1,n+1}, ..., Z_{i,n+1}) \sim \mathcal{N}_i(\boldsymbol{0}, \boldsymbol{I}_i)$.
4. Use equation (A.25) to compute $X_i(t_{n+1})$.
5. Compute $S_i(t_{n+1}) = \exp(X_i(t_{n+1}))$.

Figure A.2 depicts a simulated path of a 10-dimensional Heston model.

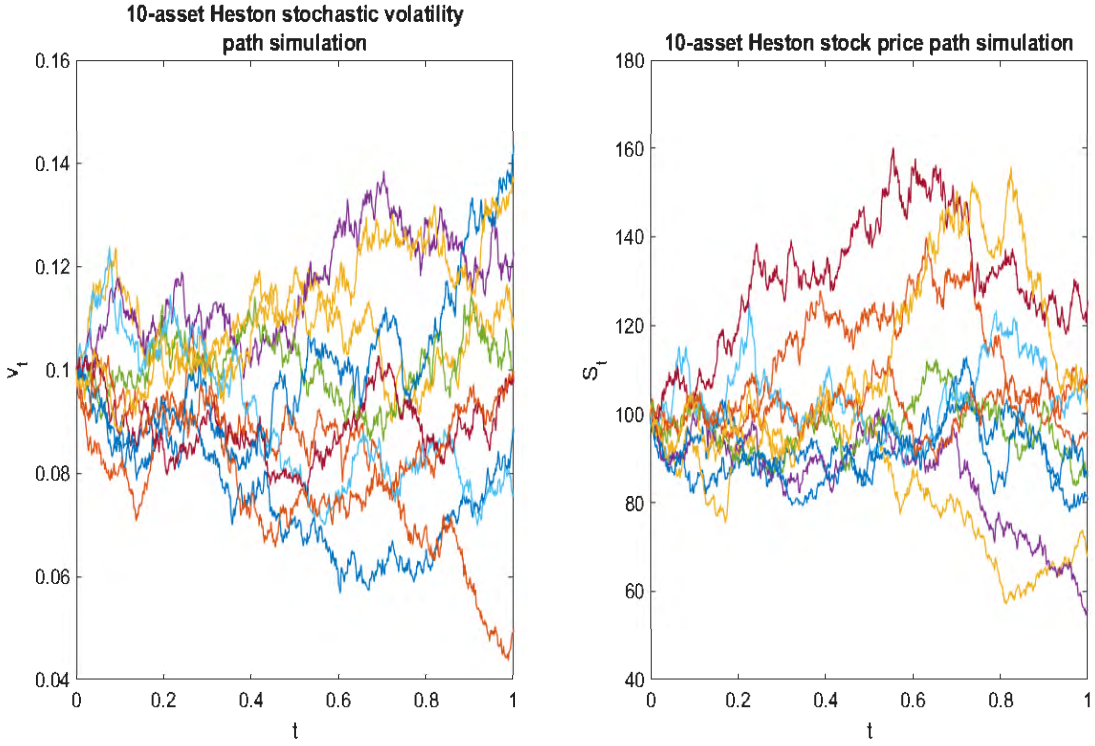


Fig. A.2: 10-dimensional Heston stochastic volatility and stock price path simulation. ($S_i(0) = 100$, $v_i(0) = 0.1$, $\kappa_i = 0.1$, $\theta_i = 0.15$, $\sigma_i = 0.1$, $\eta_i = 0.0$, $\rho_{ij}^{SS} = 0.1\mathbb{I}_{\{i\neq j\}} + \mathbb{I}_{\{i=j\}}$, $\rho_{ij}^{VV} = \mathbb{I}_{\{i=j\}}$, $\rho_{ij}^{SV} = 0.2\mathbb{I}_{\{i\neq j\}} + \mathbb{I}_{\{i=j\}}$, $r = 0.05$, $N = 1000$ time steps)

Appendix B

Quadratic Exponential (QE) Algorithm Parameters

Consider the following SDE

$$dv(t) = \kappa(\theta - v(t))\, dt + \sigma\sqrt{v(t)}\, dW(t). \tag{B.1}$$

The following results found in Andersen (2007) are important for the development of the QE algorithm and ultimately the MQE algorithm.

Theorem B.1. *Let $\Delta > 0$ and $F_{\chi^2}(x, \delta, \lambda)$ be the cumulative distribution for the non-central chi-squared distribution with δ degrees of freedom and non-centrality parameter λ. Then*

$$\mathbb{P}\left[v(t+\Delta) < x | v(t)\right] = F_{\chi^2}\left(\frac{n(t,\Delta)x}{\exp(-\kappa\Delta)}, d, n(t,\Delta)v(t)\right), \tag{B.2}$$

with $d = \frac{4\kappa\theta}{\sigma^2}$ and $n(t,\Delta) = \frac{4\kappa\exp(-\kappa\Delta)}{\sigma^2(1-\exp(-\kappa\Delta))}$

Corollary B.2. *Let $\Delta > 0$. Conditional on $v(t)$, the mean and variance of $v(t+\Delta)$ are as follows:*

$$\mathbb{E}\left[v(t+\Delta)|v(t)\right] = \theta + (v(t) - \theta)\, e^{-\kappa\Delta} \tag{B.3}$$

$$Var\left[v(t+\Delta)|v(t)\right] = \frac{v(t)\sigma^2 e^{-\kappa\Delta}}{\kappa}\left(1 - e^{-\kappa\Delta}\right) + \frac{\theta\sigma^2}{2\kappa}\left(1 - e^{-\kappa\Delta}\right)^2 \tag{B.4}$$

Consider the following definitions

$$m = \theta + (\hat{v}(t) - \theta)\, e^{-\kappa\Delta} \tag{B.5}$$

$$s^2 = \frac{\hat{v}(t)\sigma^2 e^{-\kappa\Delta}}{\kappa}\left(1 - e^{-\kappa\Delta}\right) + \frac{\theta\sigma^2}{2\kappa}\left(1 - e^{-\kappa\Delta}\right)^2 \tag{B.6}$$

Proposition B.3. *Let m and s be defined as in equations (B.5) and (B.6), and set $\psi = \frac{s^2}{m^2}$. Moreover, define $\hat{v}(t+\Delta) = a(b+Z)^2$ where Z is a standard normal random variable. Suppose $\psi \leq 2$, set*

$$b^2 = \frac{2}{\psi} - 1 + \sqrt{\frac{2}{\psi}}\sqrt{\frac{2}{\psi} - 1} \geq 0, \tag{B.7}$$

and,

$$a = \frac{m}{1+b^2}, \tag{B.8}$$

then $\mathbb{E}\left[\hat{v}(t+\Delta)\right] = m$ and $Var[\hat{v}(t+\Delta)] = s^2$.

Proof. From $\hat{v}(t+\Delta) := a(b+Z),\ Z \sim \mathcal{N}(0,1)$, it follows that $\hat{v}(t+\Delta)$ is a random variable which is a times a non-central chi-square random variable with one degree of freedom with non-centrality parameter b^2. Using the properties of a non-central chi-square distribution, it follows that,
$\mathbb{E}\left[\hat{v}(t+\Delta)\right] = a(1+b^2)$ and $\text{Var}[\hat{v}(t+\Delta)] = 2a^2(1+2b^2)$.
Match these with the moments of the square-root process- m and s^2, i.e.,
$m = a(1+b^2)$ and $s^2 = 2a^2(1+2b^2)$.
To solve this system of equations, set $x = b^2$ and $\psi = \frac{s^2}{m^2}$. Eliminate a to obtain,

$$x^2 + 2x\left(1 - \frac{2}{\psi}\right) + 1 - \frac{2}{\psi} = 0$$

The discriminant of this quadratic equation in x shows that a real solution is possible only when $\psi \leq 2$. Using this constraint, it follows that the solution for $x = b^2$ is (B.7). □

Low values of $\hat{v}(t)$ correspond with large values of ψ, which violate the constraint $\psi \leq 2$. Thus, for low values of $\hat{v}(t)$, the moment-maching fails.
To generate $\hat{v}(t+\Delta)$ when $\hat{v}(t)$ is very low requires the following approximation

$$\mathbb{P}\left(\hat{v}(t+\Delta) \in [x, x+dx]\right) \approx \left(p\delta(0) + \beta(1-p)e^{-\beta x}\right) \tag{B.9}$$

where δ is the Dirac delta-function and $p, \beta > 0$. It follows that

$$\Psi(x) := \mathbb{P}\left(\hat{v}(t+\Delta) \leq x\right) = p + (1-p)\left(1 - e^{-\beta x}\right) \tag{B.10}$$

Using the Probability Integral Transform, it follows that $\hat{v}(t+\Delta)$ can be generated using

$$\hat{v}(t+\Delta) = \Psi^{-1}(U) = \begin{cases} 0 & 0 \leq U \leq p \\ \beta^{-1}\log\left(\frac{1-p}{1-U}\right) & p < U \leq 1 \end{cases} \tag{B.11}$$

where $U \sim Uniform(0,1)$. p and β are determined by moment-matching.

Proposition B.4. *Let $m,\ s$, and ψ be defined as above. Assume that $\psi \geq 1$ and set*

$$p = \frac{\psi - 1}{\psi + 1} \in [0, 1) \tag{B.12}$$

and

$$\beta = \frac{1-p}{m} = \frac{2}{m(\psi+1)} > 0. \tag{B.13}$$

Moreover, let $\hat{v}(t+\Delta)$ be defined as in equation (B.11); then $\mathbb{E}\left[\hat{v}(t+\Delta)\right] = m$ and $Var[\hat{v}(t+\Delta)] = s^2$.

Proof. From equation (B.9), it follows by integration that

$$\mathbb{E}\left[\hat{v}(t+\Delta)\right] = \frac{1-p}{\beta}; \quad \text{Var}\left[\hat{v}(t+\Delta)\right] = \frac{1-p^2}{\beta^2}.$$

Use moment matching to obtain

$$m = \frac{1-p}{\beta}; \quad s^2 = \frac{1-p^2}{\beta^2}.$$

Eliminate β to obtain

$$(1+\psi)p^2 - 2\psi p + \psi - 1 = 0.$$

The system of equations will have one solution when $p < 1$ as in (B.12). Equation (B.13) follows. Equations (B.12) and (B.13) require $p \geq 0$ to make sense, i.e., $\psi \geq 1$. □