Gaussian Process Regression for Option Pricing and Hedging

Ishani Patel

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.

May 3, 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.

May 3, 2022

Abstract

Recent literature in the field of quantitative finance has employed machine learning methods to speed up typical numerical calculations including derivative pricing, fitting Greek profiles, constructing volatility surfaces and modelling counterparty credit risk, to name a few. This dissertation aims to investigate the accuracy and efficiency of Gaussian process regression (GPR) compared to traditional quantitative pricing algorithms. The GPR algorithm is applied to pricing a down-and-out barrier call option. Notably, Crépey and Dixon (2019) propose an alternative method for computing the Gaussian process Greeks by directly differentiating the GPR option pricing model. Based on their approach, the GPR algorithm is further extended to compute the delta and vega of the option. Numerical experiments display that option pricing accuracy scores are within a tolerable range and demonstrate increased speed of considerable magnitudes with speed-up factors in the $1\ 000$s. Computing the Greeks convey favourable computational properties; however, the GPR model struggles to obtain accurate predictions for the delta and vega. The trade-off between accuracy and speed is further investigated, where the inclusion of additional GPR input parameters hinder performance metrics whilst a larger training data set improves model accuracy.

Acknowledgements

Firstly, I would like to convey my gratitude to my supervisor, Professor Peter Ouwehand, whose insights, guidance and unwavering support has been invaluable. A huge thank you must be extended to the entire AIFMRM team for affording me the opportunity to pursue this degree. Finally and most importantly, thank you to my friends, family and parents whose constant support has never faltered. You have been my source of strength and inspiration, for which I am deeply grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Calculations based on traditional methods in financial modelling are often computationally expensive. For example, though Monte Carlo (MC) simulations have the advantage that the order of error is independent of dimension, MC estimates tend to exhibit poor accuracy relative to other quantitative methods. This may be improved by increasing the sample size; however, it comes at the expense of increased computational burden and consequently requires variance reduction techniques. In contrast, finite difference schemes are accurate but time consuming when the dimension of the problem increases. Over time, pricing models and instruments have also become more complex and may be nontrivial to evaluate. Thus, in practice, MC simulations are often relied upon.

Through recent advances, the application of machine learning (ML) has been made possible across a broad spectrum of industries, not least the finance industry. ML has become an actively researched field where literature suggests that its implementation can address and improve the accuracy and efficiency of traditional quantitative problems (De Spiegeleer *et al.* (2018)). More specifically, ML techniques have been applied to computationally expensive and challenging problems such as modelling counterparty credit risk, constructing implied volatility surfaces, fitting sophisticated Greek profiles, as well as pricing derivatives (Dixon *et al.* (2020)).

The focus of this research paper is to investigate the efficacy of Gaussian process regression (GPR) applied to option pricing and estimating sensitivity measures. GPR, also referred to as "kriging", is a supervised non-parametric Bayesian ML technique. Unlike most ML methods which parameterise exact model values, the Bayesian approach infers a probability distribution over all possible values or functions hence, the name *non-parametric*. Furthermore, these methods seek to best fit a labelled training data set, known as supervised learning, by constructing a mapping or learning function whilst still maintaining its flexibility to enable the model to generalise unseen data.

Often, the motivation for implementing GPR, rather than frequentist ML methods, such as support vector machines or artificial neural networks, is that the regression model provides probabilistic bounds so that uncertainty in the predictions can be quantified. Additionally, GPR requires much less data for training when compared to alternative deep learning techniques (Rasmussen and Williams (2006)). Versatility in the selection of kernels also allows for effective hyperparameter optimisation. Like many ML algorithms, GPR can be efficiently implemented using readily available support structures and open-source ML libraries such as scikit-learn and GpyTorch (Dixon *et al.* (2020). Furthermore, markets are often modelled with a limited set of parameters; namely, volatility, dividend yields, and interest rates. In the context of ML, this is ideal since one of the main factors which hinders model efficiency are the number of input parameters. The training time scales poorly as the dimension of the training set increases (De Spiegeleer *et al.* (2018)).

The principal aim of this dissertation is to implement and evaluate the performance of GPR applied to option pricing and Greek estimation, and compare its results to classical quantitative methods namely, MC simulation. To demonstrate the efficacy of the ML algorithm, we price a computationally expensive option, specifically a down-and-out barrier call option, in the GPR framework. Traditionally, barrier options may prove to be computationally heavy using standard pricing algorithms such as MC simulation.

This dissertation begins with a brief overview of ML fundamentals. The Gaussian process regression framework is thoroughly investigated before a review of GPR as seen in recent literature is presented. To demonstrate and solidify its key theoretical concepts, a preliminary example of GPR applied to a vanilla call option in the Black-Scholes framework is discussed. The GPR pricing algorithm is investigated in both the Black-Scholes and the Heston model, where the requisite mathematical theory of computing option prices and estimating sensitivity measures are considered in both frameworks. Given that barrier options are particularly sensitive to volatility, pricing in a stochastic volatility model is more appropriate. Subsequently, to demonstrate the efficacy of the regression model, the results of a down-and-out barrier call option are analysed under the GPR framework and compared to traditional MC methods. Finally, concluding remarks are given.

Chapter 2

Gaussian Process Regression

This chapter aims to provide the necessary mathematical framework and theoretical background of GPR. First, to highlight it key concepts, a fundamental overview of machine learning is presented. Thereafter, the theory of GPR is adapted from the seminal book, *Gaussian Processes for Machine Learning* by Rasmussen and Williams (2006). Finally, applications of GPR in quantitative finance as seen in recent literature are reviewed, accompanied by preliminary examples to illustrate the theory presented in this chapter.

2.1 Machine Learning Fundamentals

Applications of ML have been employed across a range of industries, from typical classification, regression, anomaly detection, and speech recognition problems, to MRI registration and segmentation applications in the medical field.

ML algorithms typically require a training (in-sample) and a testing (out-of-sample) data set. A model *learns* from the training data whilst the testing set serves as a measure to validate the performance of the model. More specifically, the validation process allows one to quantify a model's ability to generalise new and unseen data. Broadly speaking, ML can be categorised into supervised and unsupervised learning. Supervised learning algorithms involve applications in which the training data set comprises an input feature matrix, X, along with corresponding target outputs, y (Bishop (2006)). However, in unsupervised learning problems, the training set is unlabelled, only containing the feature matrix, X.

The quality and quantity of data sets play an integral role in the performance of ML algorithms. The model will not perform well if it learns from a poor quality training set. Data which is non-representative, polluted with irrelevant features, has insufficient sample points, or contains many outliers and errors will result in an inadequate model (Géron (2019)).

The manner in which ML models fit the testing data gives rise to two key concepts namely, overfitting and underfitting. Overfitting refers to models which perform well on training data but cannot generalise unseen data, i.e. has a low training error and a high generalisation error. One of the most common ways to reduce the risk of overfitting is to ensure that the model trains on sufficient data. Additionally, regularisation techniques may be implemented to alleviate the effects of overfitting. Conversely, underfitting occurs when a model is too simple to learn the underlying data structure. This is evident when the ML algorithm exhibits both a high training and generalisation error (Géron (2019)). Figure 2.1 provides a simple example to illustrate the concepts of underfitting and overfitting.

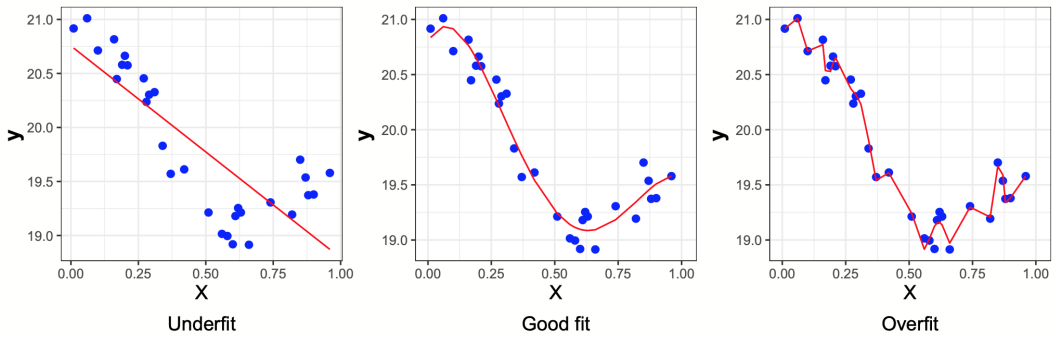

Fig. 2.1: Visual representation of an underfit, optimally fit and overfit ML model, reproduced from Badillo *et al.* (2020).

The free parameters of the algorithm control the learning process and are tuned to obtain an ideal set of hyperparameters. This is referred to as hyperparameter optimisation, the process of minimising the loss function. Common approaches include gradient descent, Bayesian optimisation and grid search algorithms. Considering the key points highlighted in this brief overview, the importance of data structure, appropriate model selection and effective hyperparameter optimisation cannot be overstated in developing a competent ML algorithm.

We end this section with the following quote which encapsulates the essence of what ML algorithms seek to achieve.

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

— Tom Mitchell, 1997

2.2 Rudimentary GPR Framework

A Gaussian process (GP) is a stochastic process defined as a collection of random variables, any finite number of which has a joint Gaussian distribution (Rasmussen and Williams (2006)). It is completely defined by its mean function, $m(x)$, and covariance function, $K(x, x')$, and can be expressed as

$$f(x) \sim \mathcal{GP}\big(m(x), K(x, x')\big),$$

where the covariance function or kernel, K, is restricted to be any symmetric positive definite function [1]. Unless otherwise specified, convention in literature is to assume a zero mean function hence, $f(x) \sim \mathcal{N}(0, K(x, x'))$.

Consider a labelled training dataset $\{(X, \boldsymbol{y}) = (\boldsymbol{x_i}, y_i) \mid i = 1, \ldots, n\}$ where n is the number of observations, $\boldsymbol{x_i}$ is a $d-$dimensional input vector and y_i is the corresponding target output. GPR ultimately seeks to find the relation between the inputs and outputs which is achieved by learning a model

$$y_i = f(\boldsymbol{x_i}) + \epsilon_i,$$

where $f(x)$ is a GP. The term $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$ represents additive i.i.d. Gaussian noise since it is typical to assume some noise in the observations, with known variance σ_n^2, to allow for more realistic modelling. Therefore, the idea of GPR is to induce a posterior distribution over all latent functions in the function space without parameterising a map (Dixon *et al.* (2020)).

Now, consider a matrix of zero mean test inputs, X_*, and its corresponding test outputs, $\boldsymbol{f}_*$. The joint prior distribution of the training, $\boldsymbol{y}$, and test, $\boldsymbol{f_*}$, outputs are Gaussian and are given by

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

where $K(X, X)$ is the matrix with $(i, j)-$entry $k(x_i, x_j)$ etc. For completeness, note that $\boldsymbol{y} = \boldsymbol{f}$ when the observations are 'clean', i.e. when the noise variance $\sigma_n^2 = 0$ (De Spiegeleer *et al.* (2018)). Furthermore, $K(X, X_*)$ is an $n \times n_*$ matrix where n_* denotes the number of test points. Following the notation of Rasmussen and Williams (2006), the predictive distribution is obtained by conditioning the joint prior on the observations given by

$$\boldsymbol{f_*}|X, \boldsymbol{y}, X_* \sim \mathcal{N}\Big(\mathbb{E}\left[\boldsymbol{f_*}|X, \boldsymbol{y}, X_*\right], \mathrm{Var}\left[\boldsymbol{f_*}|X, \boldsymbol{y}, X_*\right]\Big), \tag{2.1}$$

[1] A function $k(x, x')$ is symmetric positive definite iff for any $x_1, \ldots x_n$, the matrix $(k(x_i, x_j))_{ij}$ is symmetric positive definite for any n.

where

$$\mathbb{E}\left[\boldsymbol{f_*}|X, \boldsymbol{y}, X_*\right] = K(X_*, X)\left[K(X, X) + \sigma_n^2 I\right]^{-1}\boldsymbol{y}, \tag{2.2}$$

$$\text{Var}\left[\boldsymbol{f_*}|X, \boldsymbol{y}, X_*\right] = K(X_*, X_*) - K(X_*, X)\left[K(X, X) + \sigma_n^2 I\right]^{-1}K(X, X_*). \tag{2.3}$$

Equation (2.2) represents the predicted values of $\boldsymbol{f}_*$ for inputs X_* while equation (2.3) quantifies the uncertainty in these predictions. Importantly, in the context of this dissertation, equation (2.2) represents the GP regressed option prices. The function values $\boldsymbol{f}_*$ can be sampled from the joint posterior distribution by evaluating the above mean and covariance functions and generating multivariate Gaussian samples (Rasmussen and Williams (2006)). Note that the posterior distribution is given by the well-known Bayes' Theorem

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}.$$

2.2.1 The Kernel Function

At the core of obtaining an effective GPR model is finding a suitable kernel function which models the covariance between distributed data points. A popular choice of covariance function is the squared exponential (SE) kernel which is an example of a radial basis function (RBF). In one dimension, the SE kernel is given by

$$K(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}||\boldsymbol{x} - \boldsymbol{x'}||^2\right) + \sigma_n^2 \delta_{\boldsymbol{x}\boldsymbol{x'}}, \tag{2.4}$$

where $\delta_{xx'}$ is the Kronecker delta. The free parameters of kernel function (2.4) are the signal variance σ_f^2, noise variance σ_n^2 and length-scale l. One may also combine various kernels [2] to create hybrid kernel functions. In addition to providing more realistic predictions, incorporating a strictly positive noise variance also ensures that matrix $K(X, X) + \sigma_n^2 I$ is non-singular (De Spiegeleer *et al.* (2018)). It prevents potential numerical issues during the learning process by ensuring that a positive definite matrix is formed.

Figure 2.2 provides an illustrative example of five random functions drawn from a GP prior and GP posterior. The prior distribution is conditioned on six noise-free observations and a SE kernel function is used. In each plot, the shaded regions correspond to a 97.5% confidence interval.

[2] Refer to Rasmussen and Williams (2006) for additional choices of kernel functions which include the Matérn, $\gamma-$Exponential and Rational Quadratic function.

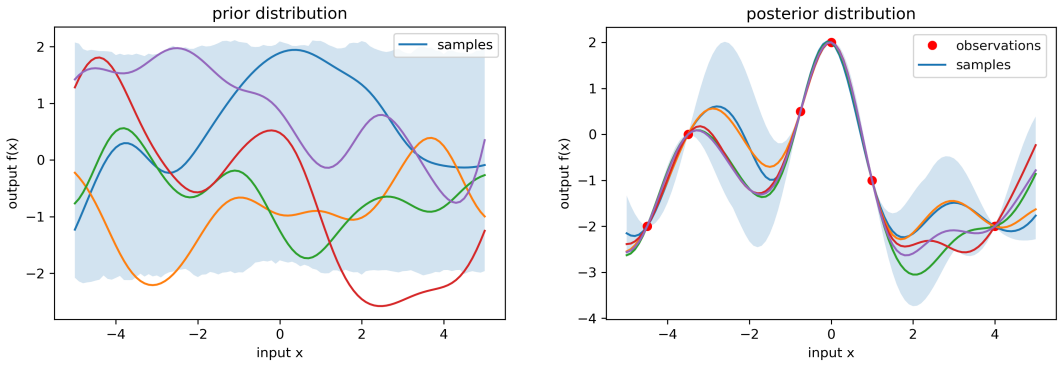

Fig. 2.2: Prior (left) and posterior (right) distributions.

2.2.2 Hyperparameter Tuning

Kernels such as the RBF are parameterised by varying the free parameters. Following Rasmussen and Williams (2006), these parameters can be calibrated to fit the data by optimising the log marginal likelihood of the data,

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

This expression is conditioned on the hyperparameters, $\boldsymbol{\theta}$, which is now explicitly shown with respect to the kernel function. Note that the hyperparameters for a RBF kernel are defined by $\boldsymbol{\theta} = [\sigma_f, \sigma_n, l]$. The first term, which contains the observed targets $\boldsymbol{y}$, can be interpreted as how well the model fits the data, the second as the complexity penalty and the third as the normalisation constant (Rasmussen and Williams (2006)). For simplicity, define $K := K(X,X) + \sigma_n^2 I$. Recall that K is the covariance matrix for noisy training targets $\boldsymbol{y}$. To estimate the parameters, the method of maximum likelihood estimation is used where the partial derivatives with respect to the hyperparameters, θ_j, is given by

$$\frac{\partial}{\partial\theta_j}\log p(\boldsymbol{y}|X,\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{y}^{\boldsymbol{T}}K^{-1}\frac{\partial K}{\partial\theta_j}K^{-1}\boldsymbol{y} - \frac{1}{2}\mathrm{tr}\left(K^{-1}\frac{\partial K}{\partial\theta_j}\right), \tag{2.5}$$

where tr is the trace of the square matrix. In practice, equation (2.5) is substituted into a gradient descent algorithm to minimise the negative log marginal likelihood. It is important to note that generally, $\log p(\boldsymbol{y}|X,\boldsymbol{\theta})$ is often a non-convex function and may suffer from multiple local minima (Bishop (2006)).

2.2.3 Computational Properties

The training time required for maximising the marginal likelihood scales poorly with the number of observations n since it requires the inversion of the $n \times n$ matrix K (Crépey and Dixon (2019)). This is one of the main sources of inefficiency

and computational complexity of GPR. As the number of observations increase, the data becomes more informative; however, this comes at the expense of escalated training time. Standard algorithms for inverting the $n \times n$ positive definite covariance matrix incurs $\mathcal{O}(n^3)$ complexity. However, once K^{-1} is known, real-time predictions are faster and are performed in $\mathcal{O}(n^2)$ (Rasmussen and Williams (2006)). Even though the training process is time-demanding, the model need only be trained once for a particular application. Thereafter, predictions for a new set of inputs are computed much faster.

2.3 Literature Review of GPR in Derivative Pricing

2.3.1 Option Pricing and Greek Estimation

GPR applications are particularly useful for estimating exotic type options which are computationally expensive and do not have analytical solutions. Additionally, given that accurate and efficient Greek computations play a vital role in effective risk management, GPR may also be applicable to estimating sensitivity measures. As markets move, there is a constant need for model calibration, hedge positions and risk metrics to be accurately and efficiently determined.

In the context of option pricing, input parameters to the GPR model can either be empirical market data (e.g. JSE/FTSE All Share Index) or standard quantitative pricing algorithms (e.g. MC estimates) which are simulated under a specified market model to obtain the target outputs. Each input parameter (e.g. strike, risk-free rate, volatility etc.) in the feature matrix X is a vector with a specified range of values. The vector $\mathbf{y}$ corresponds to target outputs which are obtained from a particular combination of input parameters. Once the GPR model is trained, $\mathbb{E}\left[\boldsymbol{f_*}|X, \boldsymbol{y}, X_*\right]$ in the predictive distribution expression (2.1) represents the GP regressed option prices and $\mathrm{Var}\left[\boldsymbol{f_*}|X, \boldsymbol{y}, X_*\right]$ quantifies the uncertainty in these predictions (Dixon *et al.* (2020)).

A study presented by De Spiegeleer *et al.* (2018) applies GPR to fast derivative pricing, fitting Greek profiles and summarising implied volatility surfaces. GPR is implemented to price European, American and barrier options where numerical experiments demonstrate speed-ups of several magnitudes relative to standard, classical pricing methods. The GPR algorithm is further applied to Greek estimation, in particular, fitting the gamma profile of a cliquet option under the Heston model where accuracy losses are observed to be within a tolerable range.

Crépey and Dixon (2019) develop an application of GPR for computing sensitivity measures which involves differentiating the kernel function. Therefore, it does not require any retraining of the algorithm as the GP Greeks are directly obtained

through differentiation. In particular, the authors investigate first order Greeks for a call option where the GP estimates are observed to closely track the Black-Scholes estimates. The analytical form of the GP Greeks proposed by Crépey and Dixon (2019) is expressed as

$$\partial_{X_*}\mathbb{E}\left[\boldsymbol{f_*}|X,\boldsymbol{y},X_*\right] = \partial_{X_*}\mu_{X_*} + \partial_{X_*}K(X_*,X)\left[K(X,X)+\sigma_n^2 I\right]^{-1}\boldsymbol{y}, \tag{2.6}$$

where the derivative is with respect to the test set variables. Recall that μ_{X_*} is usually set to zero therefore, $\partial_{X_*}\mu_{X_*}$ becomes null. Equation (2.6) can be differentiated once more with respect to X_* to obtain second order sensitivities. Importantly, the squared exponential kernel function, denoted by equation (2.4), is infinitely differentiable hence, its partial derivatives are easily computed. Regarding computational complexity, Crépey and Dixon (2019) note that calculating the Greeks imposes no significant additional computational burden since $\left[K(X,X)+\sigma_n^2 I\right]^{-1}$ was already computed during the GPR option pricing learning process.

2.3.2 Further Applications in Quantitative Finance

Thus far, emphasis was placed on the applications of GPR in the context of general option pricing and fitting Greek profiles. However, it is worth mentioning implementations of GPR in other areas of quantitative finance seen in recent literature.

In addition to pricing and Greeking, Crépey and Dixon (2019) consider a portfolio risk valuation problem where GPR is applied to estimating counterparty credit risk metrics, namely CVA. Modelling counterparty risk can be particularly challenging as it requires all trades associated with each counterparty to be simultaneously evaluated under both market and credit risk. The authors implement a multi-Gaussian process (multi-GP) regression approach which advances the work of De Spiegeleer *et al.* (2018) who limit their scope to single instrument pricing and do not consider portfolio aspects. The multi-GPs approach infers derivative prices in a portfolio of instruments. Numerical experiments demonstrate favourable accuracy and convergence properties for CVA computations when compared to a nested MC method.

Goudenège *et al.* (2019) employ GPR to price and compute the deltas for a Guaranteed Minimum Withdrawal Benefit (GMWB) Variable Annuity (VA) product under the Heston Hull-White model, which considers both stochastic volatility and stochastic interest rates. Numerical experiments indicate that GPR is effective for computing both price and sensitivity estimates. Furthermore, the authors conclude that even though their analysis was carried out for a GMWB VA, the methodology can generalise other insurance products, particularly other types of VA contracts.

Lastly, Goudenège *et al.* (2020) propose two techniques namely, GPR-Tree and GPR Exact Integration (GPR-EI), to price American basket options under the multi-dimensional Black-Scholes and rough Bergomi models. Option prices are computed on the exercise dates and is the maximum between the exercise and continuation values. The two GPR approaches are used to approximate the continuation value. Both methods are upgrades from the GPR-Monte Carlo approach where the MC based computations are replaced with a tree step and an exact integration step respectively. Experiments demonstrate that accuracy scores are reliable and computational efficiency is improved. The GPR-Tree method is efficient when the dimension or the number of assets $d \leq 10$ and is applicable for dimensions up to $d = 20$. The GPR-EI technique is particularly flexible and displayed favourable computational properties. Both methods address the issue of the curse of dimensionality.

2.4 Preliminary Examples

To solidify the theory discussed in this chapter, we present preliminary examples of GPR applied to pricing a simple European call (EC) option as well as the delta and vega in the Black-Scholes framework. Standard vanilla options play an important role in hedging and model calibration, and also form part of the building blocks in constructing structured products.

The GPR option pricing model is trained using an RBF kernel. We consider a two-dimensional case where the strike, K, and the volatility, σ, are the only input parameters which constitute the training and test sets. We begin by creating 600 samples of Black-Scholes call option prices using the well-known equation (2.7) given below

$$C = S\Phi(d_1) - Ke^{-rT}\Phi(d_2), \tag{2.7}$$

where

$$\begin{aligned} d_1 &= \frac{\log\left(\frac{S}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}, \\ d_2 &= d_1 - \sigma\sqrt{T}. \end{aligned}$$

The analytical formula for the Black-Scholes delta and vega are given by

$$\begin{aligned} \Delta &:= \frac{\partial C}{\partial S} = \Phi(d_1), \\ \nu &:= \frac{\partial C}{\partial \sigma} = S\phi(d_1)\sqrt{T}, \end{aligned} \tag{2.8}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the standard normal cumulative distribution and probability density function respectively. The data set comprises 500 samples of training data, which are uniformly sampled over a specified range, and 100 samples of testing data. Table 2.1 summarises the model input parameters. Figure 2.3 displays the corresponding results, accompanied by tables 2.2 and 2.3 which present the performance metrics for both in-sample and out-of-sample predictions.

S	1	n_{train}	500	K_{train}	$60\% \rightarrow 160\%$
r	3.5%	n_{test}	100	K_{test}	$70\% \rightarrow 150\%$
T	1			σ_{train}	$10\% \rightarrow 60\%$
				σ_{test}	$20\% \rightarrow 50\%$

Tab. 2.1: Summary of model input parameters.

The performance metrics summarised in table 2.2 and 2.3 quantify the mean absolute error (MAE), average absolute error (AAE), root mean square error (RMSE) and coefficient of determination (R^2) for the GP option price, delta and vega. Definitions for the aforementioned metrics can be found in section 4.1. Figure 2.3 illustrates the GP model results and its accompanying error plots. The GP call out-of-sample option prices display the most accurate results. The error term peaks when the strike is approximately 115%. The GP delta and option price display similar error profiles. Furthermore, they both demonstrate R^2 scores of approximately 1 which indicates that regression predictions almost perfectly fit the data. The GP vega exhibits the least accurate results, displaying that the error term peaks for lower volatility values. Given the results, one should bear in mind that the training set was only two dimensional and quite small, as S, r and T were kept constant. This was merely an illustrative example to demonstrate the mechanics of GPR applied to option pricing and Greeking. We reiterate that the GP sensitivities are computed directly from equation (2.6) and not by retraining the model on data comprised of the BS Greeks.

	EC Option Price	
	In-sample	Out-of-sample
MAE	1.777×10^{-2}	5.871×10^{-3}
AAE	2.873×10^{-3}	2.409×10^{-3}
RMSE	4.037×10^{-3}	2.910×10^{-3}
R^2	0.9999	0.9982

Tab. 2.2: GPR model performance metrics for the EC option.

	Δ	ν
MAE	1.305×10^{-2}	2.797×10^{-2}
AAE	6.399×10^{-3}	7.093×10^{-3}
RMSE	7.427×10^{-3}	1.025×10^{-2}
R^2	0.9988	0.9882

Tab. 2.3: Performance metrics for the GP Greeks: delta & vega.

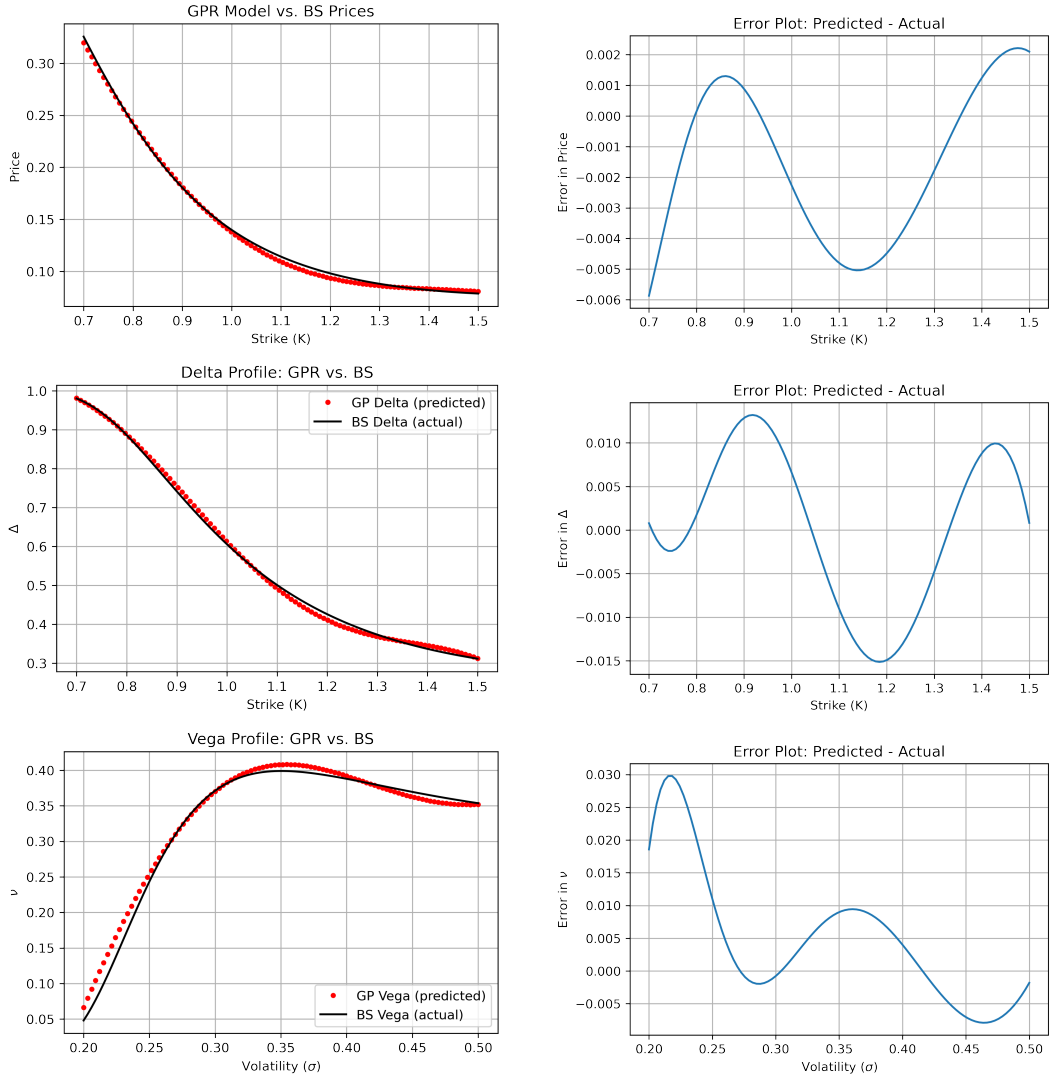

Fig. 2.3: GPR model out-of-sample results (left) and their accompanying error plots (right) for a BS call option price, delta (Δ) and vega (ν).

Chapter 3

Mathematical Preliminaries

3.1 The Models

The principal aim of this research is to implement and evaluate the performance of GPR applied to option pricing and Greek estimation, and compare results to traditional quantitative methods. In particular, this dissertation aims to highlight the efficacy of GPR when applied to computationally expensive quantitative finance problems, such as pricing and hedging path-dependent options. To demonstrate this, a down-and-out barrier call option (DOC) is priced in both the Black-Scholes and Heston models. The Black-Scholes model will serve to provide benchmark results since a closed-form solution exists within this framework. Furthermore, given that barrier options are particularly sensitive to volatility, pricing under a stochastic volatility model, such as Heston, is more appropriate. A more detailed discussion on barrier options is presented in section 3.2.1.

3.1.1 The Black-Scholes Model

In practice, the Black-Scholes model is the industry standard for option pricing. However, it is well-known that the model presents certain limitations; namely, the assumptions of constant volatility and constant short rate. Furthermore, stock prices are assumed to follow geometric Brownian motion (GBM). The risk-neutral dynamics are given by the stochastic differential equation (SDE)

$$dS_t = rS_t dt + \sigma S_t dW_t^{\mathbb{Q}},$$

where W_t is a standard Brownian motion under the risk-neutral measure $\mathbb{Q}$. The risk-free rate, r, and volatility, σ, are constants. The Black-Scholes framework is widely used in practice since prices can be easily computed. These may serve as a useful approximation to reality given that practitioners wholly understand the limitations which the model presents.

3.1.2 The Heston Model

To address the shortcomings presented by the Black-Scholes model, namely the unrealistic assumption that the volatility is known and constant, the Heston stochastic volatility model is used to describe the market. The risk-neutral dynamics of the Heston (1993) model are given by the following SDEs

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{\nu_t} S_t dW_t^{\mathbb{Q}(1)}, \\ d\nu_t &= \kappa(\theta - \nu_t)dt + \sigma\sqrt{\nu_t} dW_t^{\mathbb{Q}(2)}, \end{aligned} \tag{3.1}$$

where r is the risk-free rate, κ is the rate of mean-reversion of the variance to the mean reversion level θ, σ is the volatility of the variance and $W_t^{\mathbb{Q}(1)}$ and $W_t^{\mathbb{Q}(2)}$ are correlated Wiener processes with $dW_t^{\mathbb{Q}(1)} dW_t^{\mathbb{Q}(2)} = \rho dt$. The variance process, ν_t, mean reverts at a rate of $\kappa > 0$ to a mean reversion level $\theta > 0$. The model has an additional source of correlated randomness between the stock price process, S_t, and the volatility, ν_t, to generate the volatility skew or smile that is observed in practice (Heston (1993)).

3.2 Option Pricing

3.2.1 Analytical Solution of Barrier Options

Barrier options are a type of exotic path-dependent option, whose payoff depends on whether the underlying crosses a boundary level before expiration. They offer cheaper premiums than standard vanilla options, which contributes to their popularity. This is apparent from the in-out parity relationship[1] given by

$$\text{down-and-in} + \text{down-and-out} = \text{vanilla option}.$$

The four main types of single barrier options are down-and-in, down-and-out, up-and-in and up-and-out, each with either a call or put feature. This dissertation will focus on pricing DOC options which is a call option that becomes nullified if the price of the underlying falls below a certain boundary level at anytime before the expiry time T. The payoff of the option for some barrier $L < K$ is given by

$$C_T^{\text{DO}} = (S_T - K)^+ \mathbb{I}_{\{\inf_{t \le T} S_t > L\}}.$$

Merton (1973) derived an analytical solution for DOC options by risk-neutral valuation, from which the market for barrier options gained much traction. In the

[1] This parity relationship is true for both call and put options as well as up-and-in/out barrier options.

Black-Scholes model, the closed-form solution may be expressed as

$$C = S\Phi(d_1) - Ke^{-rT}\Phi\left(d_1 - \sigma\sqrt{T}\right) - S\Phi(d_2)\left(\frac{L}{S}\right)^{2(\mu+1)} + Ke^{-rT}\Phi\left(d_2 - \sigma\sqrt{T}\right)\left(\frac{L}{S}\right)^{2\mu},$$

where

$$\begin{aligned}
\mu &= \frac{r - \sigma^2/2}{\sigma^2}, \\
d_1 &= \frac{\log(S/K)}{\sigma\sqrt{T}} + (1+\mu)\sigma\sqrt{T}, \\
d_2 &= \frac{\log\left(L^2/(SK)\right)}{\sigma\sqrt{T}} + (1+\mu)\sigma\sqrt{T}.
\end{aligned}$$

Importantly, barrier options are particularly sensitive to volatility; a higher volatility will decrease the price of knock-out options since the underlying is more likely to dip below the boundary level. Conversely, an increase in volatility will result in an increase in the value of knock-in options as it is more likely that the underlying will reach the boundary level (Chiarella *et al.* (2012)). Therefore, it is more realistic to model barrier options in the presence of stochastic volatility. Hence, the DOC option will be priced in both the Heston model, to provide more realistic pricing and the Black-Scholes model, given that an analytical solution exists.

3.2.2 Monte Carlo Pricing

In theory, Monte Carlo methods rely on risk-neutral valuation. First, n independent random price paths of the underlying process $\{X_t\}_{0\leq t\leq T}$ are simulated. For each sample path, the associated payoff of the option, f, is computed. Finally, the discounted value of the risk-neutral expectation is calculated and the time $t = 0$ price of the option is given by

$$e^{-rT}\mathbb{E}_{\mathbb{Q}}\left[f\left(X\right)\right],$$

where $X \sim \omega$ is a sample path and ω is a probability density function. Hence, the MC estimate is computed as

$$\hat{Y} := \frac{1}{n}\sum_{i=1}^{n} f\left(X_i\right) \xrightarrow{a.s.} \mathbb{E}_{\mathbb{Q}}\left[f\left(X\right)\right] \quad \text{as} \quad n \to \infty,$$

where $X_i, \ldots, X_n$ are drawn from ω. The justification for the above estimator follows from the strong law of large numbers.

In the case where one is unable to directly simulate the process $\{X_t\}_{0\leq t\leq T}$, an approximation must be used. It is possible to apply Itô's Lemma to directly solve for S_t for the Black-Scholes SDE. However, in the Heston model, it is necessary

to utilise a numerical approximation to obtain paths for the SDEs. Following the methodology seen in Rouah (2013), this dissertation will focus on the Milstein discretisation scheme to approximate equation (3.1). Thus, the discretised stock process, $\{S_t\}_{0\leq t\leq T}$, and variance process, $\{\nu_t\}_{0\leq t\leq T}$, are given by

$$S_i = \begin{cases} S_0 & \text{if } i = 0, \\ S_{i-1}\exp\left(\left(r - \frac{1}{2}\nu_{i-1}\right)\Delta t + \sqrt{\nu_{i-1}}\sqrt{\Delta t}Z_{s,i}\right) & \text{if } i > 0, \end{cases}$$

where

$$\nu_i = \begin{cases} \nu_0 & \text{if } i = 0, \\ \left(\nu_{i-1} + \kappa\left(\theta - \nu_{i-1}\right)\Delta t + \sigma\sqrt{\nu_{i-1}}\sqrt{\Delta t}Z_{\nu,i} + \frac{1}{4}\sigma^2\left(Z_{\nu,i}^2 - 1\right)\Delta t\right)^+ & \text{if } i > 0. \end{cases}$$

and $Z_{s,i}$, $Z_{\nu,i}$ are standard normal random numbers with correlation ρ. The interval $[0, T]$ is partitioned into N subintervals of length $\Delta t = \frac{T}{N}$. Furthermore, a truncation scheme is used to ensure positive variance.

3.3 The Greeks

The ability to accurately and efficiently calculate the Greeks are arguably as important as option pricing given the key role it plays in effective risk management. MC methods for estimating sensitivity measures include finite-difference approximations, pathwise derivatives and likelihood-ratio methods, each of which present their own caveats. Though the pathwise method is an unbiased estimator, it requires the following interchange between expectation and derivative to hold

$$\frac{\partial}{\partial x}\mathbb{E}\left[f(\cdot)\right] = \mathbb{E}\left[\frac{\partial}{\partial x}f(\cdot)\right]$$

This interchange is only justified if the payoff function $f(\cdot)$ is Lipschitz continuous (Glasserman (2013)). The payoff of barrier options are dependent on whether the stock reaches a certain boundary level before maturity and are subject to knock-in or knock-out conditions. Therefore, barrier options have discontinuous payoffs and may first require its payoff to be approximated through a smoothing function before applying the pathwise method. Contrarily, likelihood-ratio methods are unaffected by the form of the payoff function; however, the estimator generally tends to display a higher variance than finite difference approximations (Glasserman (2013).

Therefore for ease of implementation, this dissertation will implement a finite-difference approximation, particularly a central-difference scheme, for estimating

the barrier option's delta and vega. The central-difference approximation of $f'(x)$ is given by

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \tag{3.2}$$

where h represents a perturbation or 'bump'. Furthermore, to reduce variance in the estimates, the method of common random numbers is employed. It should be noted that central-difference estimators require simulation at two different points, $x + h$ and $x - h$, whereas forward-difference estimators only require simulation at one point, $x + h$. Even though this poses additional computational overhead, the central-difference scheme yields superior accuracy when compared to forward-difference approximations (Glasserman (2013)).

For completeness, we provide the definitions of delta and vega for a DOC option below

$$\Delta = \frac{\partial \text{DOC}}{\partial S}, \tag{3.3}$$

$$\nu^{\text{Black-Scholes}} = \frac{\partial \text{DOC}}{\partial \sigma}, \tag{3.4}$$

$$\nu^{\text{Heston}} = \frac{\partial \text{DOC}}{\partial \nu_0}. \tag{3.5}$$

It is important to differentiate between the Black-Scholes and Heston model vega. In the Black-Scholes framework, the vega is computed with respect to the implied volatility, σ. Whereas the Heston model vega is computed with respect to the initial variance, ν_0. These are represented by equations (3.4) and (3.5) respectively.

Chapter 4

GPR Model Design

4.1 Option Pricing Implementation

Drawing from the theoretical concepts presented in Chapter 2, the GPR option pricing model can now be defined. A brief note on the general implementation of the model is outlined by the below steps. Note that Y_{bench} refers to target outputs obtained from standard quantitative finance pricing methods, whilst Y_{predict} refers to the GP regressed outputs.

1. Create a training and testing data set by simulating option prices using standard pricing methods under a specified market model. In particular, data sets are created by simulating option prices under both the Black-Scholes and Heston models.

2. Fit the GPR model to the training set (in-sample data) and evaluate the model performance on the testing set (out-of-sample data). Recall that GP regressed prices and their associated uncertainty in predictions are given by equation (2.1).

3. Analyse and compare performance metrics of standard option pricing methods with the GPR algorithm. Capturing and quantifying the quality of predictions are measured in terms of:

    - **Speed** – speed-ups are obtained by comparing CPU runtimes. Given that the GPR model need only be trained once, the CPU time of the GP predictions are compared with the CPU time of the standard MC pricing algorithm.
    - **Accuracy** – accuracy scores are measured in terms of maximum absolute error (MAE), average absolute error (AAE) and root mean square error (RMSE) given respectively by

$$\text{MAE} = \max\left\{\left|Y_{\text{bench}}(i) - Y_{\text{predict}}(i)\right|\right\}, \tag{4.1}$$

$$\text{AAE} = \frac{1}{n}\sum_{i=1}^{n}\left|Y_{\text{bench}}(i) - Y_{\text{predict}}(i)\right|, \tag{4.2}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}\left(Y_{\text{bench}}(i) - Y_{\text{predict}}(i)\right)^2}{n}}, \tag{4.3}$$

where $i = 1, ..., n$ is the number of observations. RMSE is a useful performance metric since the error terms are squared before averaged. This poses a higher penalty on larger errors. In addition to the above three metrics, the R^2 score, also known as the coefficient of determination, is a statistical measure of fit which allows us to quantify the percentage of variation. Scores can range from 0 to 1 where an R^2 of 1 indicates that the regression predictions perfectly fit the data. The formula is given by

$$R^2 = 1 - \frac{RSS}{TSS}, \tag{4.4}$$

where the residual sum of squares (RSS) and the total sum of squares (TSS) are given respectively by

$$RSS = \sum_{i=1}^{n}\left(Y_{\text{bench}}(i) - Y_{\text{predict}}(i)\right)^2,$$

$$TSS = \sum_{i=1}^{n}\left(Y_{\text{bench}}(i) - \bar{Y}_{\text{bench}}\right)^2.$$

Note that $\bar{Y}_{\text{bench}} = \frac{1}{n}\sum_{i=1}^{n} Y_{\text{bench}}(i)$ is the mean of the observed data.

Further expanding on *Step 1* and following the notation used in Chapter 2, the training and testing data sets are constructed as input and corresponding target output pairs denoted by $(X, \boldsymbol{y})$ and $(X_*, \boldsymbol{f_*})$ respectively. In the Black-Scholes framework, the input matrix is given as

$$\begin{aligned} X &= [\boldsymbol{K}, \boldsymbol{T}, \boldsymbol{r}, \boldsymbol{\sigma}, \boldsymbol{L}] \\ &= \begin{bmatrix} K_1 & T_1 & r_1 & \sigma_1 & L_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ K_n & T_n & r_n & \sigma_n & L_n \end{bmatrix}. \end{aligned} \tag{4.5}$$

Similarly, under the Heston model, the input matrix is given as

$$X = [\boldsymbol{K}, \boldsymbol{T}, \boldsymbol{r}, \boldsymbol{\sigma}, \boldsymbol{L}, \boldsymbol{\nu_0}, \boldsymbol{\kappa}, \boldsymbol{\theta}, \boldsymbol{\rho}] \tag{4.6}$$
$$= \begin{bmatrix} K_1 & T_1 & r_1 & \sigma_1 & L_1 & v_{0_1} & \kappa_1 & \theta_1 & \rho_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ K_n & T_n & r_n & \sigma_n & L_n & v_{0_n} & \kappa_n & \theta_n & \rho_n \end{bmatrix}.$$

Expressions (4.5) and (4.6) demonstrate that X has dimension $n \times d$, where n is the number of observations and d denotes the number of input parameters. Random parameter combinations are sampled uniformly over a range for each input parameter. Every parameter combination has a corresponding target output (the option price) which is computed using a standard pricing algorithm namely, MC simulation as presented in Chapter 3. Given that GP predictions perform worse closer to the boundaries, the parameter ranges in the test set are specified to be slightly smaller relative to those in the training set (De Spiegeleer *et al.* (2018)). Note that expressions (4.5) and (4.6) do not contain S_0 as an input parameter, this is discussed in more detail in the next section.

4.2 Greek Implementation

In this short note, we restate the key computations of the GP Greek implementation highlighted in section 2.3. As previously mentioned, a popular choice of covariance function is the RBF kernel given by

$$K(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x - x')^2\right) + \sigma_n^2 \delta_{xx'}. \tag{4.7}$$

An advantage of the RBF kernel is that it is infinitely differentiable. This allows for the GP Greeks to be easily computed. Recall that predicted values for GP option prices are given by

$$\mathbb{E}\left[\boldsymbol{f_*}|X, \boldsymbol{y}, X_*\right] = \mu_{X_*} + K(X_*, X)\left[K(X, X) + \sigma_n^2 I\right]^{-1}\boldsymbol{y}. \tag{4.8}$$

By differentiating equation (4.8) with respect to X_*, we obtain a general analytical form of the GP Greeks, as formulated in Crépey and Dixon (2019), given by

$$\partial_{X_*}\mathbb{E}\left[\boldsymbol{f_*}|X, \boldsymbol{y}, X_*\right] = \partial_{X_*}\mu_{X_*} + \partial_{X_*}K(X_*, X)\left[K(X, X) + \sigma_n^2 I\right]^{-1}\boldsymbol{y}. \tag{4.9}$$

Substituting equation (4.7) into equation (4.9), assuming a zero mean in the test data set, i.e. $\mu_{X_*} = 0$, and simplifying yields

$$\partial_{X_*}\mathbb{E}\left[\boldsymbol{f_*}|X, \boldsymbol{y}, X_*\right] = \frac{1}{l^2}\left(X - X_*\right)K(X_*, X)\left[K(X, X) + \sigma_n^2 I\right]^{-1}\boldsymbol{y}. \tag{4.10}$$

We emphasise that the GP Greeks are obtained through equation (4.10) and not by retraining the GPR model on sensitivity measure data. It is important to note that $[K(X,X)+\sigma_n^2 I]^{-1}\boldsymbol{y}$ was already calculated during the option pricing training process which significantly reduces the total computational overhead needed to obtain the GP Greeks. Hence, the sensitivities of an option are directly obtained by differentiating the Gaussian Process pricing model.

This dissertation will particularly focus on the delta and vega of a DOC option. The GP vega estimate is obtained by differentiating with respect to σ in the Black-Scholes model and ν_0 in the Heston model. Given that an option's moneyness, S/K, is of importance and expressions (4.5) and (4.6) do not consider S_0 as an input parameter, the GP delta can instead be computed in terms of the strike, K, and the barrier level, L. To demonstrate this, first, consider a DOC option price in the Black-Scholes framework given by

$$DOC(S_0, K, L, T, r, \sigma).$$

This can be rewritten as

$$S_0 DOC(1, \frac{K}{S_0}, \frac{L}{S_0}, T, r, \sigma), \tag{4.11}$$

which follows from the homogeneity property of log-type models described by Joshi (2001), where the option price is an homogeneous function of S_0, K and L. The delta is computed by differentiating equation (4.11) with respect to S_0. Given that K and L are in terms of S_0, a combination of the product and chain rule is used to obtain an expression for the delta. For readability, we suppress the down-and-out option price: $DOC(1, \frac{K}{S_0}, \frac{L}{S_0}, T, r, \sigma) = DOC$. Therefore,

$$\begin{aligned}
\Delta_{GP} &= \frac{\partial}{\partial S_0}\left[S_0 DOC(1, \frac{K}{S_0}, \frac{L}{S_0}, T, r, \sigma)\right] \\
&= DOC + S_0 \frac{\partial}{\partial S_0} DOC \\
&= DOC + S_0 \frac{\partial}{\partial K} DOC \cdot -\frac{K}{S_0^2} + S_0 \frac{\partial}{\partial L} DOC \cdot -\frac{L}{S_0^2} \\
&= DOC - \frac{\partial}{\partial K} DOC \cdot \frac{K}{S_0} - \frac{\partial}{\partial L} DOC \cdot \frac{L}{S_0}
\end{aligned} \tag{4.12}$$

The first term in expression (4.12) represents predicted GPR DOC option prices whereas $\frac{\partial}{\partial K} DOC$ and $\frac{\partial}{\partial L} DOC$ denote the GP Greeks with respect to K and L. The homogeneity property holds in the Heston model as it can also be described as a log-type model (Joshi (2001)). Thus, the Heston GP delta may be obtained in the same manner. Equation (4.12) allows the GP delta to be computed without having S_0 as an additional input parameter in the training set X. This is desirable as the training time scales poorly as the number of model input parameters, d, increases.

4.3 Software Implementation

The GPR algorithm is implemented in Python using open-source *Scikit-Learn* packages. All computations are performed on a Windows 10 operating system with specifications of Intel i7-10610U quad core CPU processor at 1.8GHz−2.3GHz with 16GB RAM. Python code is implemented in Jupyter Notebook, an open-source, web-based computing environment. Key libraries imported for this dissertation include numpy, scipy, matplotlib.pyplot, pandas and sklearn. In particular, the GPR implementation follows from *Scikit-learn: Machine learning in Python* by Pedregosa *et al.* (2011). This implementation is based on the algorithm from the seminal book, *Gaussian Processes for Machine Learning* by Rasmussen and Williams (2006), as reviewed in Chapter 2. We briefly summarise the parameters for the GPR algorithm:

```
sklearn.gaussian_process.GaussianProcessRegressor(kernel,
        alpha, optimizer, n_restarts_optimizer,
        normalize_y, copy_X_train, random_state)
```

- `kernel` – Specified by the RBF kernel. It is important to note that the RBF kernel implementation in Scikit-Learn is only parameterised by the length scale, l. The signal variance, σ_f^2, can be represented by the `ConstantKernel` which is combined with the RBF to obtain the desired form of the kernel function expressed in equation (4.7).

- `alpha` – Can be interpreted as a regularisation parameter or as the noise variance σ_n^2. It ensures that $\left[K(X,X)+\sigma_n^2 I\right]$ is a positive definite matrix and prevents any numerical issues during the training process.

- `optimizer` – Optimises the kernel function's hyperparameters by minimising the negative log marginal likelihood.

- `n_restarts_optimizer` – Specifies the number of random restarts of the optimiser since the log marginal likelihood is often non-convex and may contain multiple local optima.

- `normalize_y` – Boolean argument defaulted to *False* as the GPR algorithm assumes that the data is normalised with zero mean.

- `copy_X_train` – Boolean argument defaulted to *True* where a persistent copy of the training data set is stored.

- `random_state` – Argument defaulted to *None* and specified by an integer if reproducible results are desired.

Chapter 5

Results and Analysis

5.1 Option Pricing

Training (in-sample) and testing (out-of-sample) data sets are created by uniformly sampling combinations from model parameter ranges. Input parameter ranges for the Black-Scholes and Heston models are displayed in table 5.1 and 5.2 respectively.

Parameter	Range (Training Set)	Range (Testing Set)
K	$70\% \rightarrow 140\%$	$80\% \rightarrow 130\%$
T	$1M \rightarrow 1Y$	$1M \rightarrow 1Y$
r	$3\% \rightarrow 5\%$	$3.5\% \rightarrow 4.5\%$
σ	$15\% \rightarrow 40\%$	$20\% \rightarrow 35\%$
L	$30\% \rightarrow 100\%$	$40\% \rightarrow 90\%$

Tab. 5.1: Black-Scholes model parameter ranges for training and testing data sets.

Parameter	Range (Training Set)	Range (Testing Set)
K	$70\% \rightarrow 140\%$	$80\% \rightarrow 130\%$
T	$1M \rightarrow 1Y$	$1M \rightarrow 1Y$
r	$3\% \rightarrow 5\%$	$3.5\% \rightarrow 4.5\%$
σ	$15\% \rightarrow 40\%$	$20\% \rightarrow 35\%$
ν_0	$0.01 \rightarrow 0.1$	$0.02 \rightarrow 0.09$
κ	$1.2 \rightarrow 2$	$1.3 \rightarrow 1.9$
θ	$0.1 \rightarrow 0.8$	$0.2 \rightarrow 0.7$
ρ	$-0.9 \rightarrow -0.5$	$-0.8 \rightarrow -0.6$
L	$30\% \rightarrow 100\%$	$40\% \rightarrow 90\%$

Tab. 5.2: Heston model parameter ranges for training and testing data sets.

Given that an analytical solution for a DOC option exists in the Black-Scholes framework, we display GPR predictions vs. closed-form prices as a benchmark re-

sult. Speed-ups are not applicable in this scenario as indeed, the analytical solution will always yield the fastest results. An illustration of the GPR model results are displayed in figure 5.1 with performance metrics shown in table 5.3. Importantly, note that if GPR prices were perfect predictions of the analytical prices, all points would lie on the line $y = x$.

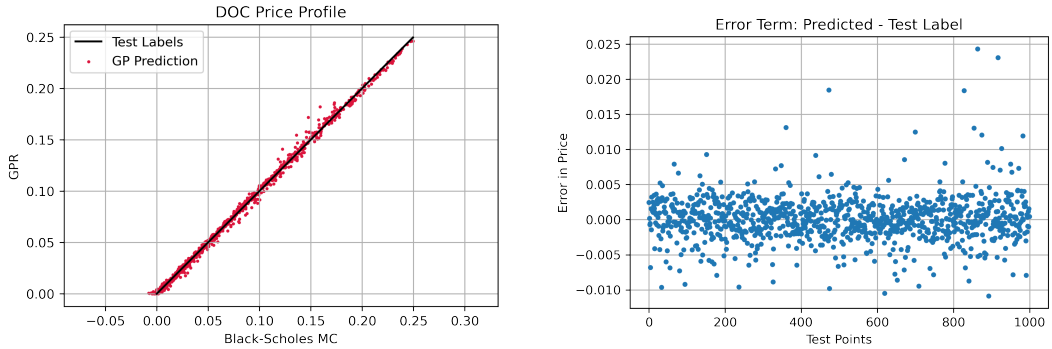

Fig. 5.1: 1 000 out-of-sample GP regressed option prices (left) trained on 10 000 Black-Scholes analytical points with the accompanying error plot (right).

	In-sample	Out-of-sample (Test Range)	Out-of-sample (Train Range)
MAE	4.421×10^{-2}	2.431×10^{-2}	4.842×10^{-2}
AAE	2.363×10^{-3}	2.226×10^{-3}	2.403×10^{-3}
RMSE	3.782×10^{-3}	3.213×10^{-3}	3.986×10^{-3}
R^2	0.9981	0.9973	0.9971

Tab. 5.3: GPR model performance metrics displayed for analytical Black-Scholes DOC option prices. Out-of-sample predictions correspond to 1 000 sample points.

In general, ML models display lower accuracy scores in the validation set compared to the training set. We emphasise that the performance metrics are better on a smaller test set which has been resized to exclude parameters closer to its boundaries. When the ranges are the same, the metrics behave as expected. For reference, we have provided out-of-sample predictions using a validation data set constructed from both the training and testing input parameter ranges. Note that in table 5.1 and 5.2, the test set, X_*, is constructed by sampling over smaller ranges since GPR typically performs worse closer to the boundaries (De Spiegeleer *et al.* (2018)). For this reason, the *Out-of-sample (Test Range)* predictions are more accurate than the *Out-of-sample (Train Range)* results which is evident in table 5.3.

Next, Black-Scholes MC option prices are obtained by simulating $50\ 000$ paths with 100 discrete time intervals. The training data set is varied in size to assess the trade-off between model accuracy and speed. Figure 5.2 graphically displays the performance and table 5.4 quantifies the accuracy scores and speed-ups. Similarly, Milstein MC estimates in the Heston model are computed by simulating $50\ 000$ paths with 100 updates over the time interval. Figure 5.3 displays the GP option pricing results as well as the accompanying error plots for varying training set sizes. Performance metrics are summarised in table 5.5.

Train Set Size	1 000		5 000		10 000	
	Train	Test	Train	Test	Train	Test
MAE	2.206×10^{-2}	1.864×10^{-2}	1.717×10^{-2}	1.351×10^{-2}	1.563×10^{-2}	1.267×10^{-2}
AAE	4.281×10^{-3}	3.427×10^{-3}	2.117×10^{-3}	1.591×10^{-3}	1.766×10^{-3}	1.448×10^{-3}
RMSE	5.533×10^{-3}	4.423×10^{-3}	2.950×10^{-3}	2.342×10^{-3}	2.538×10^{-3}	2.230×10^{-3}
R^2	0.9965	0.9956	0.9990	0.9988	0.9993	0.9991
Speed-up	$\times$ 927		$\times$ 892		$\times$ 459	

Tab. 5.4: GPR model performance metrics quantified for varying training data set sizes. Results correspond to Black-Scholes MC DOC option prices. All out-of-sample (test) results consist of $1\ 000$ points.

Train Set Size	1 000		5 000		10 000	
	Train	Test	Train	Test	Train	Test
MAE	3.312×10^{-2}	3.072×10^{-2}	2.526×10^{-2}	2.289×10^{-2}	2.107×10^{-2}	2.121×10^{-2}
AAE	4.537×10^{-3}	4.376×10^{-3}	2.993×10^{-3}	2.710×10^{-3}	2.638×10^{-3}	2.348×10^{-3}
RMSE	6.542×10^{-3}	6.297×10^{-3}	4.300×10^{-3}	3.946×10^{-3}	3.751×10^{-3}	3.458×10^{-3}
R^2	0.9955	0.9927	0.9981	0.9971	0.9985	0.9978
Speed-up	$\times$ 5134		$\times$ 4808		$\times$ 2469	

Tab. 5.5: GPR model performance metrics quantified for varying training data set sizes. Results correspond to Heston MC DOC option prices. All out-of-sample (test) results consist of $1\ 000$ points.

We note that the accuracy scores namely, MAE, AAE and RMSE, of the out-of-sample predictions exceed those of the training data set. Once again, this is attributed to the testing set having a smaller range than the training set.

5.1.1 GPR Option Pricing: Model Accuracy and Computational Properties

Model accuracy and computational efficiency are the two key metrics used to evaluate the performance of the GPR model. Training the model on varying data set sizes allows for the trade-off between model speed and accuracy to be quantified. The GPR option pricing model demonstrates considerable speed-ups of several orders of magnitude, particularly in the Heston model. This being said, it is important to note that as the number of training samples doubles from $5\ 000$ to $10\ 000$ samples, the speed-up measure approximately halves, demonstrating the compromise between speed and accuracy. The performance metrics convey that there are two prominent factors which impact model accuracy:

1. The size of the data set. As the number of input parameters (dimension d) increases, the lower the accuracy. This is apparent when we compare GPR results in the Black-Scholes and Heston models. GP regressed option prices trained on data simulated under the Heston model display lower accuracy scores owing to the additional input parameters where $d = 9$. In the Black-Scholes framework, however, $d = 5$ and the accuracy scores are higher. Furthermore, the performance metrics in table 5.4 and 5.5 demonstrate that as the number of training sample points (n) increase, the accuracy improves.

2. The width of the parameter ranges. Parameters sampled from smaller ranges result in denser training sets and yield improved results. This is the principal reason for constructing a test set where parameter ranges are sampled over a smaller range. GPR tends to exhibit poorer accuracies closer to the boundaries De Spiegeleer *et al.* (2018). Table 5.3 demonstrates this when we compare *Out-of-sample (Test Range)* with *Out-of-sample (Train Range)* accuracy scores.

Graphically, it is evident from figure 5.2 and 5.3 that the error terms are initially sparse but gradually become more concentrated as the size of the training set increases. This indicates that predicted GP regressed option prices deviate less from their corresponding test labels. Note that in a perfect estimation scenario, we would observe the predicted values to lie along the straight line, $y = x$, without any deviation.

As discussed, a larger training data set yields more accurate GPR model predictions. However, this comes at the expense of increased training time. Recall, the major source of inefficiency within the GPR framework is attributed to the Cholesky decomposition of $K := \left[K(X, X) + \sigma_n^2 I\right]$. This incurs $\mathcal{O}(n^3)$ complexity, where n is the number of observations. Clearly, the training time does not scale

well as the dimension of the training set, X, increases. Additionally, the computational complexity escalates when obtaining optimal hyperparameters through the method of maximum likelihood estimation. Once again, if K^{-1} is known, the partial derivative computations expressed in equation (2.5) are faster and require time $\mathcal{O}(n^2)$ per hyperparameter. Furthermore, given that the log-marginal-likelihood, $\log p(\boldsymbol{y}|X, \boldsymbol{\theta})$, is often non-convex, it may suffer from multiple optima. Therefore, the parameter `n_restarts_optimizer` is often set to a value > 1 which imposes additional computational overhead. Considering all of the above factors, the GPR training process can prove to be quite taxing. Fortunately, the model need only be trained once. Thereafter, real-time GP predictions are efficiently computed and are performed in $\mathcal{O}(n^2)$ with a matrix-vector multiplication for each test point.

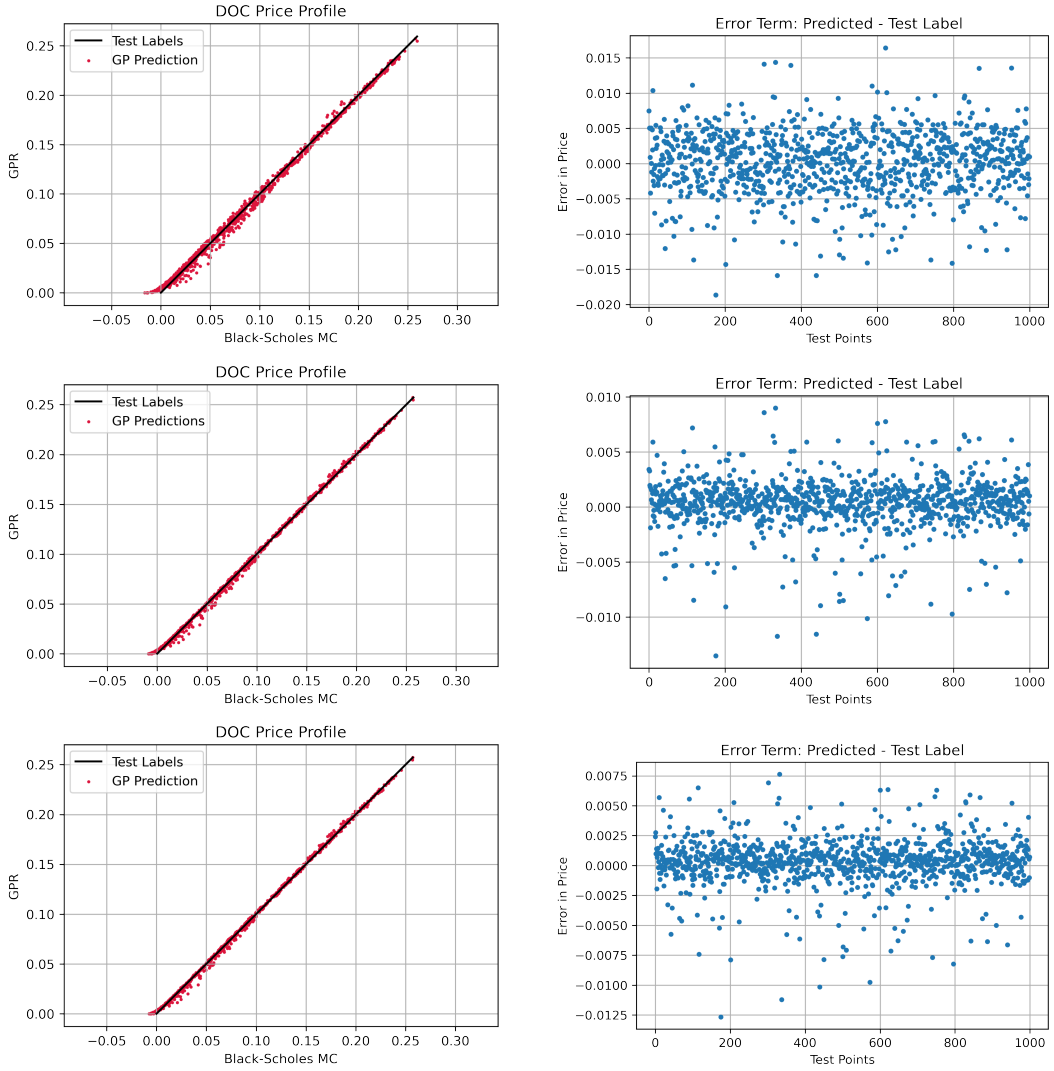

Fig. 5.2: 1 000 out-of-sample predictions (left) with the accompanying error plots (right) for a GPR model trained on 1 000, 5 000 and 10 000 Black-Scholes MC DOC option price samples.

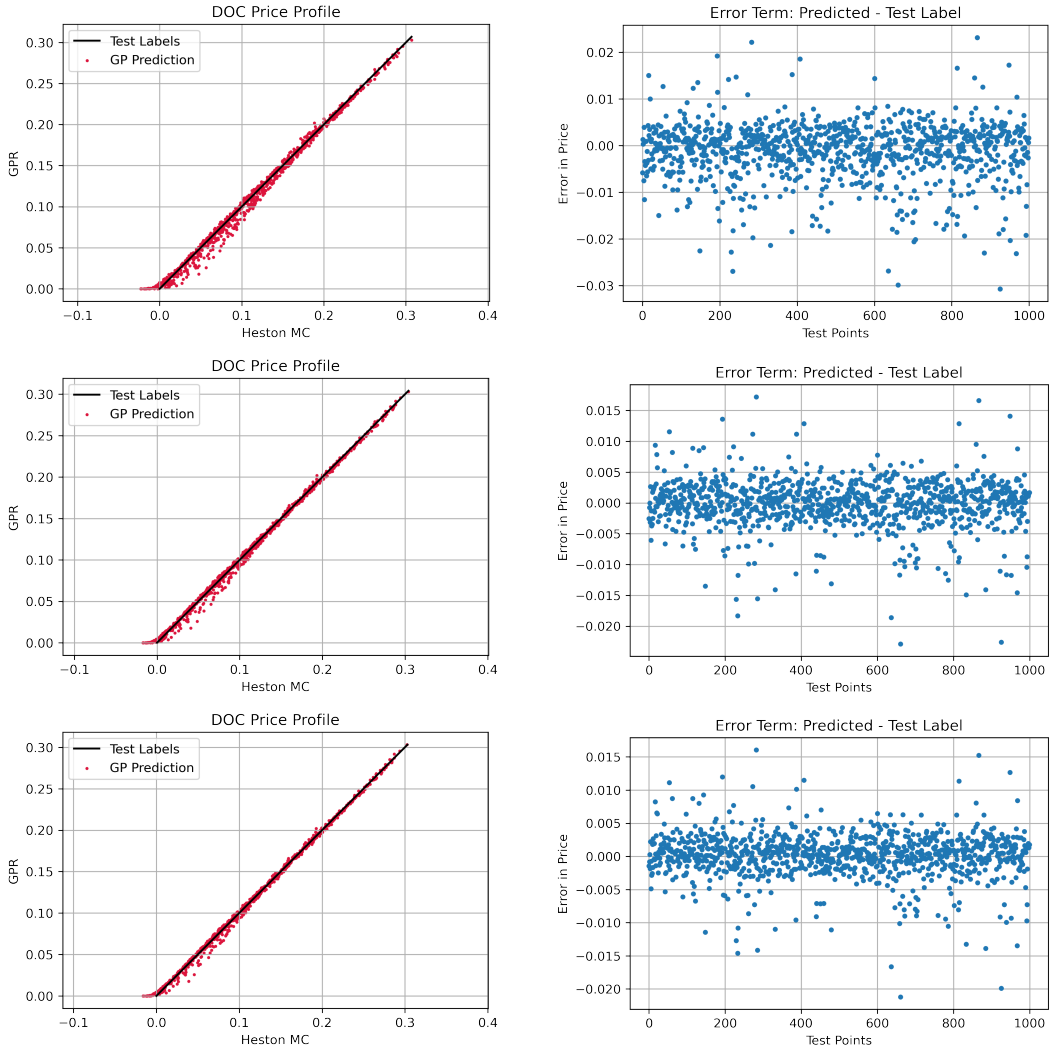

Fig. 5.3: 1 000 out-of-sample predictions (left) with the accompanying error plots (right) for a GPR model trained on 1 000, 5 000 and 10 000 Heston MC DOC option price samples.

5.2 The Greeks

The performance metrics for the GP Greeks are summarised in table 5.6 and figures 5.4 and 5.5 display the results. The GP Greeks are compared with Black-Scholes and Heston Greeks, which are computed via MC central-difference approximation using equation (3.2).

	Black-Scholes		Heston	
	Δ	ν	Δ	ν
MAE	0.110	4.671×10^{-2}	0.216	0.136
AAE	9.597×10^{-3}	8.502×10^{-3}	1.453×10^{-2}	1.685×10^{-2}
RMSE	1.674×10^{-2}	1.196×10^{-2}	2.741×10^{-2}	2.853×10^{-2}
R^2	0.9958	0.9852	0.9855	0.7914
Speed-up	$\times$ 3		$\times$ 7	

Tab. 5.6: Performance metrics for the GP Greeks vs. MC central finite-difference approximation.

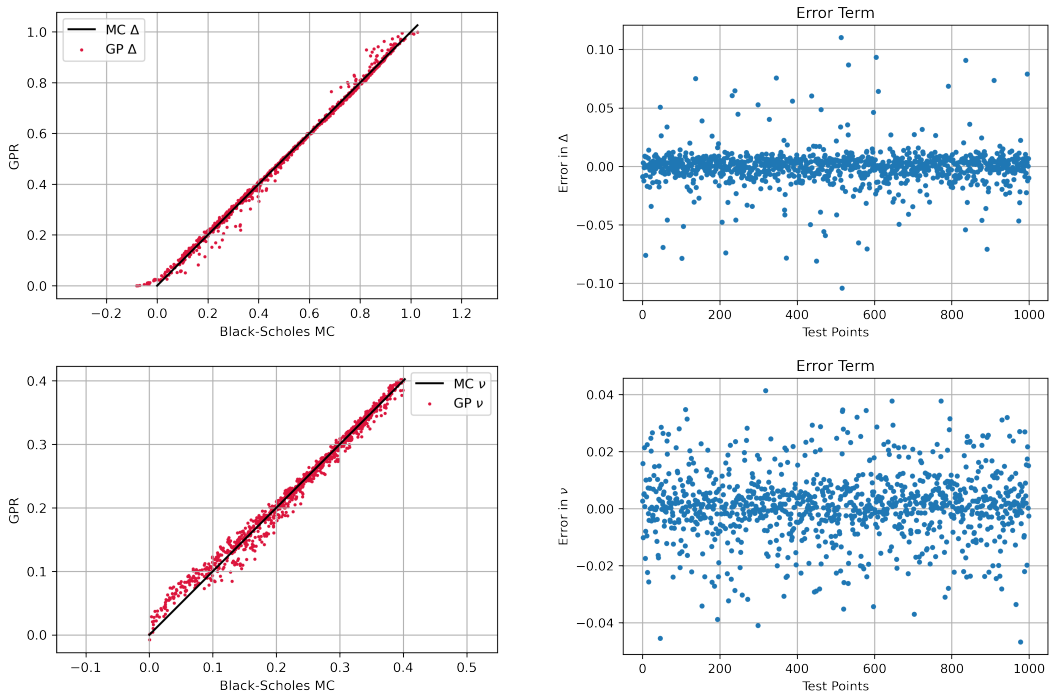

Fig. 5.4: 1 000 GP delta and vega predictions (left) with the accompanying error plots (right) for a GPR model trained on 10 000 Black-Scholes MC DOC option price samples.

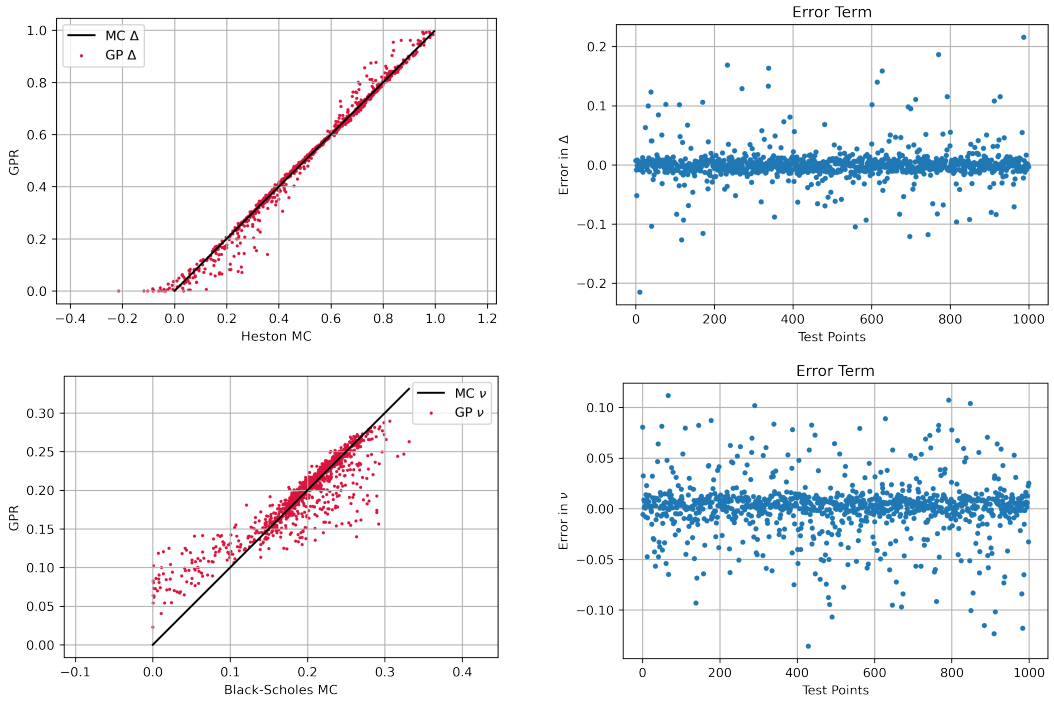

Fig. 5.5: 1 000 GP delta and vega predictions (left) with the accompanying error plots (right) for a GPR model trained on 10 000 Heston MC DOC option price samples.

5.2.1 GP Delta and Vega: Model Accuracy and Computational Properties

The performance metrics in table 5.6 convey that the Heston model displays less accurate results than the Black-Scholes model. As per the GPR option pricing model, a reason for this may be attributed to the Heston model having a larger input matrix dimension where $d = 9$. In particular, the Heston GP vega displays the least accurate R^2 score of 0.7914, indicating a lower degree of correlation between the regression model and the corresponding Heston MC samples.

Recall, the GP delta is computed via equation (4.12). In the context of ML, this form is desirable since the train and test data sets (X and X_*) do not require the initial stock price, S_0, as an additional input parameter. Therefore, computations are more efficient since the GP delta is computed in terms of the predicted option price and the GP derivative of K and L respectively. Fundamentally, equation (4.10) describes the gradient of the GPR pricing model with respect to a particular parameter in X_*. It can be viewed as an analytical solution where predictions for the GP

Greeks are directly computed from equation (4.10) without the need to retrain the initial GPR model. Once again, recall that $\left[K(X,X)+\sigma_n^2 I\right]^{-1}\boldsymbol{y}$ was already calculated during the GPR option pricing training process and therefore, computing the GP Greeks does not impose any significant computational overhead. Notably, even though table 5.6 demonstrates speed-ups relative to a standard MC simulation, they are not as significant when compared to the speed-ups seen in the GPR option pricing framework.

5.3 Further GPR Model Analysis

5.3.1 Comparability of GPR with MC Prices

Though computing various accuracy metrics are useful in evaluating the performance of the GPR model, it is also informative to quantify how GPR estimates compare with MC prices. Given that GPR can ultimately be described as a an interpolation function, the accuracy of the model is largely dependent on the accuracy of the MC estimates it trains on. Importantly, option prices obtained by MC simulation are not necessarily "ground truth", since rerunning the MC will yield slightly different results. Therefore, MC prices are approximations and the GPR algorithm essentially smooths these. For this reason, we may simulate MC estimates for a specific set of input parameters, obtain a distribution, compute a 95% confidence interval and inspect whether the predicted GPR price lies within these bounds. This allows us to quantify GPR price errors and compare these to its MC counterparts. Table (5.7) summarises these findings.

	Black-Scholes	Heston
Number of MC simulations	100	
Number of random parameter combinations	50	
% GPR estimates within MC confidence interval	68%	64%

Tab. 5.7: % of GPR estimates which lie within a 95% confidence interval of the MC prices. The confidence interval is computed based on 100 MC samples for each of the 50 random combinations (i.e. this totals to $5\ 000$ MC prices). The input parameter combinations were sampled uniformly over the test set range for the Black-Scholes and Heston models.

We note that more accurate results can be achieved by simulating an increased number of MC sample paths, yielding MC prices which are much closer to its "ground-truth" value. Additionally, training the GPR model on a larger data set

size will increase model accuracy. Again, we note that this boils down to the trade-off between model speed and accuracy.

5.3.2 Limitations of Computing the GP Greeks and Possible Recommendations

The performance metrics discussed in section 5.2 convey that the GP Greeks, particularly the Heston GP vega, yield less accurate results than its Black-Scholes counterpart. The GP Greeks, in general, also display poorer accuracies relative to the GP option pricing model. This could be attributed to MC option prices exhibiting "bouncy" behaviour causing the GP derivative function to not approximate well. Furthermore, given that the Heston model has almost double the number of input parameters, this may be a contributing factor as to why the model struggles to compute the Greeks, displaying larger errors when compared with Black-Scholes estimates. In this short note, we provide possible suggestions that can address the shortcomings of the GP Greeks, with particular reference to the limitations observed for the Heston GP vega.

As a first recommendation, one can follow a similar approach to the GPR option pricing process where the GPR model is directly trained on simulated vega estimates. Alternatively, a regularisation technique can be employed. Regularisation involves imposing additional constraints on the optimisation function to achieve a desired optimal result. In the context of GPR, we can incorporate the error terms of Δ and ν in the likelihood function. Recall, the model aims to optimise this function. In its most general form, the updated likelihood function can be expressed as

$$\min_{\theta_1,\ldots,\theta_n} f(X,\theta_1,\ldots,\theta_n) + \alpha \sum_{i=1}^{n} \Big(Y_{\text{bench}_\Delta} - Y_{\text{predict}_\Delta}(X,\theta_1,\ldots,\theta_n) \Big)^p + \beta \sum_{i=1}^{n} \Big(Y_{\text{bench}_\nu} - Y_{\text{predict}_\nu}(X,\theta_1,\ldots,\theta_n) \Big)^q, \tag{5.1}$$

where α, β, p and q are tuning parameters which control the impact of variance and bias in the regularisation technique. Roughly speaking, expression (5.1) ultimately forces the outcomes of Δ and ν to be closer to model predictions. However, this technique introduces certain caveats. Given that the GPR model will now try to optimise Δ and ν in addition to option prices, one needs to consider the implication of this on the regressed option prices. Though more reasonable GP vega estimates may be attained, the accuracy of option prices will suffer.

The above were offered as suggestions to improve the accuracy of GP vega estimates. Having offered possible solutions, we note that they are not in the spirit of

what this dissertation aims to demonstrate, particularly with respect to computing the GP Greeks. By retraining the model on vega estimates, equation (4.10) is no longer necessary. It will also require the training of additional, separate models to obtain option price predictions and each of the GP Greeks, which will introduce significant, additional computational overhead. These suggestions hinder the initial desire to compute the GP Greeks via its analytical form as expressed in equation (4.10) which provides an overall more efficient alternative for computing the GP Greeks.

Chapter 6

Conclusion

This dissertation aimed to examine the efficacy of GPR, a supervised, non-parametric Bayesian machine learning technique, applied to derivative pricing and hedging. GPR model performance metrics were measured against traditional quantitative pricing techniques, namely MC simulation, to evaluate relative speed-ups and accuracy scores. Notably, barrier options are particularly computationally expensive to compute via MC methods. Therefore, we investigated the accuracy and efficiency of GPR applied to pricing a DOC option compared to a standard MC pricing technique. The implementation of the GPR algorithm closely followed the methodology presented by De Spiegeleer *et al.* (2018) and Crépey and Dixon (2019). Results were obtained and compared in both the Black-Scholes model and Heston stochastic volatility model. Furthermore, the analytical Black-Scholes solution of the DOC option was used as benchmark result. In addition to option pricing, the GP delta and vega were computed following the implementation derived by Crépey and Dixon (2019), where the GP Greeks are directly obtained by applying equation (4.10).

Option price predictions indicate that accuracy scores are within a tolerable range and demonstrate increased speeds of considerable magnitudes with speed-up factors in the $1\ 000$s. Therefore, GPR proves to be a viable solution for fast derivative pricing when considering path-dependent, computationally expensive options, even under advanced models beyond Black-Scholes such as the Heston model. The Black-Scholes GP Greeks displayed more accurate results compared to those computed in the Heston framework. In particular, the Heston GP vega presented limitations where the GPR model struggled to obtain accurate predictions.

As expected, the largest training data set, consisting of $10\ 000$ samples, yielded the most accurate results. Though the training procedure is time consuming, the GPR model need only be trained once. Thereafter, the model can obtain significantly faster predictions. This is referred to as offline learning where after the initial training phase is complete, the model does not change its approximation of

the predictive function. In the context of obtaining an effective machine learning model, the choice of an appropriate kernel function and effective hyperparameter optimisation cannot be overstated. Importantly, the effective computation of the GP Greeks was enabled by the property of the RBF kernel being infinitely differentiable. Second order Greeks may be obtained by differentiating equation (4.10) once more.

The trade-off between model accuracy and speed was investigated by varying the number of training samples. Overall, experiments demonstrated that key factors which affected the accuracy and efficiency of the GPR pricing framework are the number of training samples (n), the number of input model parameters (dimension d) and the width of the chosen parameter ranges.

Moreover, we quantified GPR price errors and discussed the comparability to its MC counterparts. Importantly, it should be noted that MC prices are not necessarily ”ground truth” since simulating MC estimates multiple times will yield different results. An alternate approach to obtain more accurate GPR estimates can be to train the model on prices which are much closer to its ”ground truth” value by simulating a larger number of MC sample paths. As a consequence, this will also lead to substantial speed-ups.

Bibliography

Badillo, S., Banfai, B., Birzele, F., Davydov, I. I., Hutchinson, L., Kam-Thong, T., Siebourg-Polster, J., Steiert, B. and Zhang, J. D. (2020). An introduction to machine learning, *Clinical Pharmacology & Therapeutics* **107**(4): 871–885.

Bishop, C. M. (2006). *Pattern recognition and machine learning*, Springer.

Chiarella, C., Kang, B. and Meyer, G. H. (2012). The evaluation of barrier option prices under stochastic volatility, *Computers & Mathematics with Applications* **64**(6): 2034–2048.

Crépey, S. and Dixon, M. (2019). Gaussian Process Regression for Derivative Portfolio Modeling and Application to CVA Computations, *arXiv:1901.11081* .

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.

Dixon, M. F., Halperin, I. and Bilokon, P. (2020). *Machine Learning in Finance: From Theory to Practice*, Springer International Publishing, Cham.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*, O'Reilly Media.

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

Goudenège, L., Molent, A. and Zanette, A. (2019). 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. (2020). Machine Learning for Pricing American Options in High-Dimensional Markovian and non-Markovian models, *Quantitative Finance* **20**(4): 573–591.

Heston, S. L. (1993). A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options, *Review of Financial Studies* **6**(2): 327–343.

Joshi, M. (2001). Log-type models, homogeneity of option prices and convexity, *QUARC, Royal Bank of Scotland working paper* .

Merton, R. C. (1973). Theory of rational option pricing, *The Bell Journal of economics and management science* pp. 141–183.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**: 2825–2830.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*, Adaptive computation and machine learning, MIT Press, Cambridge, Mass.

Rouah, F. D. (2013). *The Heston model and its extensions in Matlab and C*, John Wiley & Sons.