Gaussian Process Regression for a Single Underlying Autocallable Security

Rebecca Herbert

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.

October 15, 2023

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 Cape Town. It has not been submitted before for any degree or examination in any other University.

October 15, 2023

Signed by candidate

Abstract

Traditionally in Quantitative Finance, in order to price exotic options, particularly with path dependency, time consuming Monte Carlo simulations are done. This dissertation considers the use of the machine learning technique Gaussian Process Regression (GPR) as a faster pricing alternative to Monte Carlo simulations. The speed of calculation is of interest since prices are linked to fast moving market variables. We focus on the pricing of a single underlying autocallable under GPR against its traditional pricing under a Stochastic Local Volatility (SLV) model. An autocallable is a structured product which allows for early redemption when the underlying meets certain barrier conditions. Due to its path dependency, autocallables are typically priced using Monte Carlo simulations under an SLV model which captures realistic market dynamics by allowing volatility to be modelled as stochastic rather than assumed constant, but also allows for more precise calibration by including a local volatility component. We find that a desired level of accuracy is achieved only for autocallable prices under the Schobel-Zhu SLV model, with computation speeds slightly slowler than Monte Carlo simulations.

Contents

Chapter 1

Introduction

Machine learning is a set of techniques which gives outputs in the form of classifications or predictions from input data without explicit programming. Given a limited data set of input-output pairs, the model is trained on this data set in such a way that it will predict new output with relative accuracy when given a new set of inputs, without the need for explicit programming, thus saving computation time. In recent years, machine learning has gained popularity in various areas of Quantitative Finance including risk management, asset management, option trading and market making (Gonzalvez *et al.* (2019)).

Traditionally in Quantitative Finance, in order to price exotic options, particularly with path dependency, time consuming Monte Carlo simulations are done whereby underlying risk factors e.g. asset price, interest rates etc. are simulated over multiple sample paths. These sample paths are then used to determine other measures such as derivative prices, hedge positions, and risk metrics. Recalculation is required with each market move and the information is useful for a limited time (De Spiegeleer *et al.* (2018)). Quantitative Finance in particular may benefit most from the use of Machine Learning techniques where many calculations need to be run frequently but often entail similar inputs (e.g. strike, current asset price, maturity, barrier) with a single output and thus can share a training process. Additionally, the speed of calculation is vital when determining values that are linked to fast moving market variables.

This research considers the use of a particular Machine Learning technique known as Gaussian Process Regression to improve the speed of derivative pricing, with particular focus on the pricing of a univariate autocallable in comparison to its traditional pricing under a Stochastic Local Volatility model. An autocallable is a structured product which allows for early redemption when the underlying meets certain barrier conditions. It is typically priced under a Stochastic Local Volatility

model as constant volatility models do not produce sufficiently realistic market dynamics.

We begin with an introduction to the statistical theory underpinning Gaussian Process Regression, then consider its existing applications in Quantitative Finance. Next we give context to autocallables and the Stochastic Local Volatility model, and finally test the use of GPR in the pricing of univariate autocallables.

Chapter 2

Gaussian Process Regression

2.1 Foundations of Gaussian Process Regression

In this section, Gaussian process regression (GPR) is introduced with the relevant references therein. We begin by introducing the Bayesian statistical framework under which GPR will be considered. Next, the statistical theory of a Gaussian Process is discussed, and then extended into Gaussian Process Regression. Finally the kernel selection and hyperparameter estimation of GPR is considered.

2.1.1 Bayesian Statistical Framework

To begin, we consider a data set of observations as outlined in Williams and Rasmussen (2006):

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, .., n\},$$

where $\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n$ in $\mathbb{R}^d$ are the d-dimensional input vectors aggregated in a $d \times n$-matrix X, and each y_i is the output corresponding to $\mathbf{x}_i$. In particular, we are concerned with a vector $\mathbf{x}_i$ containing market variables and model parameters e.g. interest rate, volatility, mean-reversion parameter etc. Then $\mathbf{x}_i$ through some payoff function $f(\mathbf{x}_i)$ will produce an option price y_i.

A simplistic approach to modelling the relationship between $\mathbf{x}_i$ and y_i would be in the form of linear regression:

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{x}^\intercal \mathbf{w}, \\ y &= f(\mathbf{x}) + \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2), \end{aligned}$$

where $\mathbf{w}$ is a vector of parameters. In a non-Bayesian machine learning context, an algorithm would attempt to learn exact values for $\mathbf{w}$ based on the given input-output pairs specified in $\mathcal{D}$. Then for some new input vector $\mathbf{x}_*$, predictions for the output are given by:

$$y_* = f(\mathbf{x}_*) + \epsilon.$$

In a Bayesian framework however, rather than learning exact values for $\mathbf{w}$, a distribution is specified over the parameters, known as a prior distribution $p(\mathbf{w})$, intended to express some prior belief about the parameters, without having observed the data set. Then using Bayes' Theorem, one can see that:

$$p(\mathbf{w} \mid y, X) = \frac{p(y \mid X, \mathbf{w})\ p(\mathbf{w})}{p(y \mid X)},$$

where $p(\mathbf{w})$ is the prior, $p(y \mid X, \mathbf{w})$ is the likelihood, $p(y \mid X)$ is the marginal likelihood, and $p(\mathbf{w} \mid y, X)$ is known as the posterior i.e.

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

The posterior distribution then incorporates information about the data set via the likelihood distribution, as well as prior beliefs via the prior distribution.

Williams and Rasmussen (2006) show that for some new input vector $\mathbf{x}_*$ and $f(\mathbf{x}_*) = f_*$, predictions for the output are given by:

$$p(f_* \mid \mathbf{x}_*, \mathbf{y}, X) = \int_w p(f_* \mid \mathbf{x}_*, \mathbf{w})\ p(\mathbf{w} \mid \mathbf{y}, X) dw.$$

If the likelihood and prior are assumed to be Gaussian, then the resulting predictive distribution is also Gaussian. The mean of the predictive distribution then represents the point prediction of the model, and the variance represents the uncertainty about that prediction. Further details of the posterior distribution are given in section 2.1.3.

In the context of path-dependent derivative pricing, one typically makes use of Monte Carlo simulations, whereby various parameters are used to simulate many possible paths of the underlying asset for certain time steps. The payoff of the derivative is then calculated at the relevant time steps and discounted to t_0. Finally, the mean of the discounted payoff is taken to calculate the derivative price. The parameters used in simulating the underlying are often market variables, which are likely to change, particularly in liquid markets. When these parameters change, the underlying's path will have to be simulated again in order to reprice the derivative. The use of GPR in derivative pricing attempts to lessen computation time by training the model over a large set of possible parameters X and the corresponding option prices y. GPR then obtains a posterior distribution for the derivative prices. When the parameters change (e.g. due to market movements), the posterior distribution gives an estimate for an option price taking into account the new parameters, without having to rerun the simulated paths.
The details of the posterior distribution in GPR are discussed in the section below.

2.1.2 Gaussian Process Regression

This section details how GPR arrives at a posterior distribution to make predictions about new sets of input data. An introduction is first given on Gaussian processes, and then the expansion into Gaussian Process Regression is discussed.

Gaussian Process Definition

To begin, a normally distributed random variable x has probability density function:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right),$$

where μ is the mean and σ^2 is the variance.

Further, a n-dimensional multivariate normal vector X has probability density function:

$$p(X) = \frac{1}{(2\pi)^{n/2}\det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(X-\boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(X-\boldsymbol{\mu})\right),$$

where $\boldsymbol{\mu}$ represents the mean vector, and the $\boldsymbol{\Sigma}$ the covariance matrix (Bishop and Nasrabadi (2006)).

If the vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ are jointly Gaussian, i.e. $\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}$ is multivariate normal, then their distributions are as follows:

$$\mathbf{x}_1 \sim \mathcal{N}(\mu_1, \Sigma_{11}),$$
$$\mathbf{x}_2 \sim \mathcal{N}(\mu_2, \Sigma_{22}),$$

$$\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right),$$

where $\Sigma_{12} = \Sigma_{21}{}^\intercal$, and the following result holds:

$$\mathbf{x}_1 \mid \mathbf{x}_2 \sim \mathcal{N}\left(\mu_1 + \Sigma_{12}{\Sigma_{22}}^{-1}(\mathbf{x}_2 - \mu_1), \Sigma_{11} - \Sigma_{12}{\Sigma_{22}}^{-1}\Sigma_{21}\right).$$

A Gaussian Process is defined as a collection of random variables, in which any finite number of these random variables have a joint Gaussian distribution (Williams and Rasmussen (2006)).

Denoted mathematically in Crépey and Dixon (2019) and De Spiegeleer *et al.* (2018), $f \sim \mathcal{GP}(m, k)$ if for any inputs $\mathbf{x_1}, \mathbf{x}_2, ...\mathbf{x}_n$ the vector of functions

$$[f(\mathbf{x_1}), f(\mathbf{x}_2), ...f(\mathbf{x}_n)] \sim \mathcal{N}(m(X), K(X,X)),$$

where

$$K(X,X) = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & ... & k(\mathbf{x}_1,\mathbf{x}_n)\\ k(\mathbf{x}_2,\mathbf{x}_1) & ... & k(\mathbf{x}_2,\mathbf{x}_n)\\ ... & ... & ...\\ k(\mathbf{x}_n,\mathbf{x}_1) & ... & k(\mathbf{x}_n,\mathbf{x}_n)\end{bmatrix}.$$

Such a process is characterised by its' mean and covariance functions:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})],$$

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= cov(f(\mathbf{x}), f(\mathbf{x}')) \\ &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \end{aligned}$$

In most of the literature, the mean function is taken to be zero for simplicity. The covariance function $k(\mathbf{x}, \mathbf{x}')$ is also known as the kernel, and can be specified by a variety of functions, the most common in Quantitative Finance being the Squared Exponential Kernel (SE). The kernel function can be chosen so as to impose certain desirable properties on the prior distribution, such as regularity, periodicity, or monotonicity, which will be discussed later.

The properties of Gaussian processes inform the way in which Gaussian Process Regression arrives at a predictive distribution, which is discussed below.

Gaussian Process Regression Prediction

In the context of GPR, there are two data sets of interest, called the training set and the test set. The training set refers to the set

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, .., n\},$$

used as inputs to train the model. Then for a new set of input parameters $\mathbf{x}_*$ we wish to predict y_*, referred to as the test set.

Williams and Rasmussen (2006) cover two different predictive cases, using noise-free and noisy observations, which are discussed below.

Noise-free Predictions

In the case of noise-free observations, it is assumed that:

$$y_i = f(\mathbf{x}_i).$$

Assuming a mean of zero,then the joint distribution of the training and test sets are:

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right),$$

and hence as per the properties of jointly Gaussian variables, the posterior distribution of f_* is:

$$\begin{aligned} f_* \mid X_*, X, f \sim \mathcal{N}\big(&K(X_*, X)K(X, X)^{-1}f, \\ &K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\big). \end{aligned}$$

Noisy Predictions

In the case of noisy observations, it is assumed that:

$$y_i = f(\mathbf{x}_i) + \epsilon, \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2).$$

As before, assuming a mean of zero, the joint distribution of y (determined from the training set), and the unknown vector of functions f_* is:

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

Once again the resulting posterior distribution of Gaussian random variables is:

$$f_* \mid X_*, X, y \sim \mathcal{N}\big(K(X_*,X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} y, \\ K(X_*,X_*) - K(X_*,X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} K(X,X_*)\big).$$

As previously stated, the mean of the posterior can now be used as a point prediction, and the variance as the uncertainity about that prediction. Note that setting $\sigma_n^2 = 0$ results in the noiseless case. (Williams and Rasmussen (2006)). Dixon *et al.* (2020) also notes the added benefit that should users want to retrain the GPR model, the previous posterior can potentially be used as the new prior.

2.1.3 Kernel selection and Hyperparameter estimation

A variety of kernel functions exist with various desirable properties, some of which we discuss below. The parameters specified in the kernel function are known as hyperparameters and we discuss their estimation later.

Note that covariance matricies are symmetric positive semi-definite and the Cholesky decomposition can be applied when inverting these matrices to lead to $O(n^3)$-operations (Gonzalvez *et al.* (2019)). A kernel is positive semi-definite if:

$$\int k(\mathbf{x},\mathbf{x}')\, f(\mathbf{x})\, f(\mathbf{x}')\, d\mu(\mathbf{x})\, d\mu(\mathbf{x}') \geq 0 \quad \forall \quad f \in \mathcal{GP}.$$

Linear Kernel

This is a very simple kernel of the form:

$$k(\mathbf{x},\mathbf{x}') = \mathbf{x}^\intercal \mathbf{x}',$$

with the resulting GPR being Bayesian linear regression, and multiplication with itself is equivalent to Bayesian polynominal regression (Wilson and Adams (2013)).

Squared Exponential kernel

As previously stated, the Squared Exponential is the most widely used kernel function in Quantitative Finance. Its representation is as follows:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{\|x - x'\|_2^2}{\ell^2}\right), \tag{2.1}$$

where ℓ is a positive constant that changes the characteristic length-scale of the process, allowing for closer fit to the observations. Informally, ℓ is described as "roughly the distance you have to move in input space before the function value can change significantly" (Williams and Rasmussen (2006)).

This kernel can also be expanded into an automatic relevance determination (ARD) kernel which is denoted instead as:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\intercal \Sigma (\mathbf{x} - \mathbf{x}')\right),$$

where taking $\Sigma = \text{diag}(\ell_1^2, ..., \ell_d^2)$ allows each of the inputs to be scaled individually. This means that when these parameters are tuned they can assert a certain relevance about the input e.g. $\ell_j = 0$ implies the j^{th} input is not relevant and eliminates it from the kernel.

Matern kernel class

This is a general class of covariance functions with two positive hyperparameters, ℓ and ν:

$$k(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\,\|\mathbf{x} - \mathbf{x}'\|_2}{\ell}\right)^\nu k_\nu\left(\frac{\sqrt{2\nu}\,\|\mathbf{x} - \mathbf{x}'\|_2}{\ell}\right),$$

where k_ν is the ν modified Bessel function of the second kind. This kernel simplifies to the squared exponential kernel when $\nu \to \infty$.

The addition of extra parameters allows for added flexibility when fitting to the training set.

The most commonly used Matern kernels are the Matern32 and Matern52, setting ν to $\frac{3}{2}$ and $\frac{5}{2}$ respectively. Setting ν to a multiple of $\frac{1}{2}$ simplifies the Matern kernel:

$$k_{32}(r) = \left(1 + \frac{\sqrt{3}r}{\ell}\right)\exp\left(-\frac{\sqrt{3}r}{\ell}\right),$$

$$k_{52}(r) = \left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right)\exp\left(-\frac{\sqrt{5}r}{\ell}\right).$$

Periodic Exponential Kernel

This kernel function allows for periodicity effects which may be useful when using regression to predict a time series (Gonzalvez *et al.* (2019)). The kernel function is:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2}\sum_{j=1}^{d}\frac{1}{\ell_j^2}\sin^2\left(\frac{\pi}{\lambda_j}\left(x_j - x_j'\right)\right)\right).$$

Spectral Mixture kernel

Wilson and Adams (2013) introduce a new kernel by establishing equivalence between stationary covariance kernels and their Fourier transforms, and creating Gaussian mixtures of spectral densities to make an extension of the squared exponential kernel function. They illustrate a mixture of n_m Gaussian densities with mean vectors $(\mu_1, ..., \mu_{n_m})$, covariance matrices $(\Sigma_1, .., \Sigma_{n_m})$, and weight ω_m for the m^{th} Gaussian distribution. When limiting the case to real-valued kernels, the resulting kernel is:

$$k(s) = \sum_{m=1}^{n_m}\omega_m \cos\left(2\pi s^\intercal \mu_m\right)\exp\left(-2\pi^2 s^\intercal \Sigma_m s\right).$$

The most useful feature of this kernel is that it can learn negative covariances, which is relevant when modelling mean-reverting processes.

Brownian Motion kernel

Although not as commonly used in Quantitative Finance GPR problems, it is useful to note that the kernel used to generate Brownian Motion paths against time is:

$$k(\mathbf{x}, \mathbf{x}') = min(\mathbf{x}, \mathbf{x}').$$

For illustrative purposes, in figure 2.1 we show five prior samples drawn from different kernels. These are generated firstly by evenly spacing the sample input in the range $[-5, 5]$. These inputs are then used to create a covariance matrix by inputting the vector in the relevant kernel function. Finally, a prior is drawn by generating a multivariate random normal vector, assuming a mean zero vector, and using the generated covariance matrix.

Further, we show a simple example of the prior and posterior distribution for the squared exponential kernel. We consider 5 observations as shown in table 2.1. In order to make predictions about $f(x)$, table 2.1 can be used as a training set, and a posterior distribution can be inferred. The posterior's mean then represents the point prediction, and the variance represents the uncertainty about that prediction. Figure 2.2 below shows five samples from the prior distribution of the Gaussian process, and figure 2.3 shows five samples from the posterior distribution as well as the mean function.

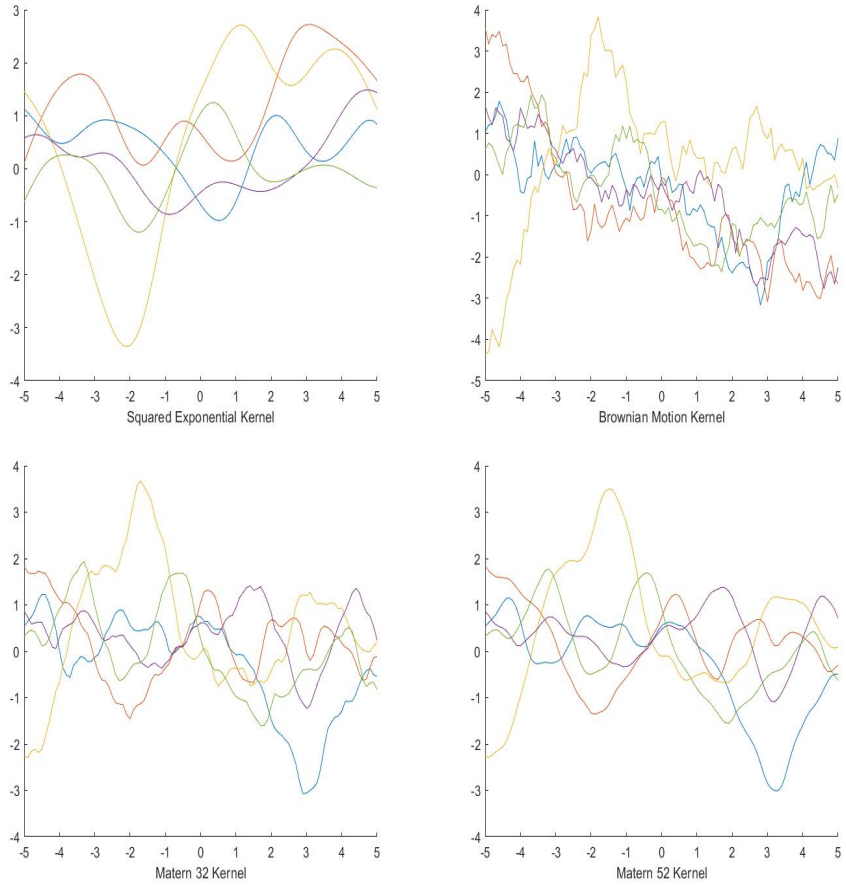

Fig. 2.1: Sample priors from different kernels.

x	-4.5	-3.5	-0.5	0	1
$f(x)$	-2	0	1	2	-1

Tab. 2.1: SE kernel prior observations.

Hyperparameter Estimation

Each kernel function's hyperparameters will need to be estimated from the training data once an appropriate kernel function is chosen (De Spiegeleer *et al.* (2018)). Parameter estimation serves as a means to fit the model to the data and is typically done by maximizing the marginal log-likelihood. In what follows, λ represents the vector of hyperparameters, for instance in the case of a squared exponential kernel, $\lambda = [\theta_k \quad \sigma]$ where $\theta_k = [\sigma_f \quad l]$ is the parameters of the kernel $K(X, X)$, and σ is the standard deviation of the noise (Crépey and Dixon (2019)).

The likelihood is defined as the marginal probability of the data given the model with respect to the learned kernel hyperparameters i.e. $L(\lambda) = p(y|X, \lambda)$ and

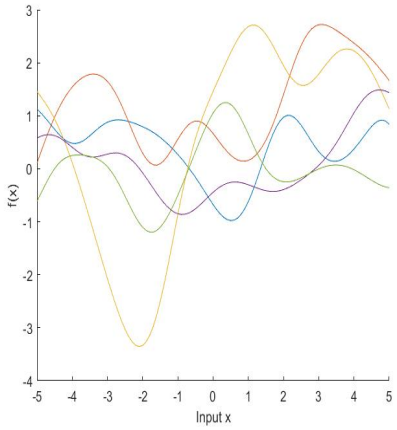

Fig. 2.2: Prior distribution of predictor function without learning.

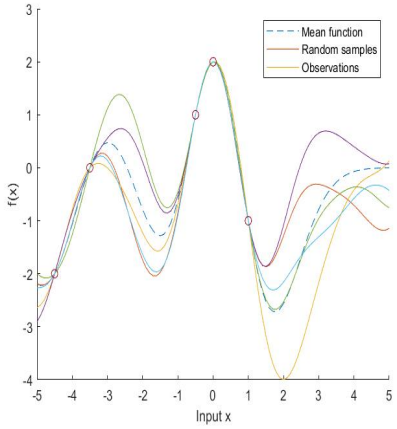

Fig. 2.3: Posterior distribution of predictor function after learning observations.

Williams and Rasmussen (2006) shows the marginal log-likelihood is:

$$\ell(\lambda) = \log\, p(y \mid X, \lambda) = -\big[y^T \left[K(X,X) + \sigma^2 I\right]^{-1} y \\ + \log(\det(K(X,X) + \sigma^2 I))\big] - \frac{n}{2}\log 2\pi.$$

Then we seek a λ_* such that:

$$\lambda_* = \arg\max_{\lambda} \log\, p(y \mid X, \lambda).$$

Note, Williams and Rasmussen (2006) describes the term $y^T \left[K(X,X) + \sigma^2 I\right]^{-1} y$ as a measure of model fit, and the term $\log(\det(K(X,X) + \sigma^2 I))$ as a pentaly term for complexity.

Methods such as gradient descent and Quasi-Newton are often used to determine

λ_* but suffer from the draw back that $\ell(\lambda)$ is potentially not convex, and return local maxima rather than global maxima (Bishop and Nasrabadi (2006)).

Alternatively, denoting $C_n = \left[K(X,X) + \sigma^2 I_n\right]$, taking the derivative of $\ell(\lambda)$ with respect to the hyperparameter i.e. λ_i, we obtain:

$$\frac{\partial}{\partial \lambda_i}\ell(\lambda) = -\frac{1}{2}\text{Tr}\left(C_n^{-1}\frac{\partial C_n}{\partial \lambda_i}\right) + \frac{1}{2}y' C_n^{-1}\frac{\partial C_n}{\partial \lambda_i}C_n^{-1}y.$$

Then if a prior is assumed over the observed data, a maximum can be obtained (Williams and Rasmussen (2006)).

2.2 Quantitative Finance applications of Gaussian Process Regression

Given the theoretical grounding above, we review the literature on existing applications of GPR to Quantitative Finance problems.

The methodology used to test GPR in computing derivative prices and hedge ratios typically consists of constructing a training set input matrix, X, by varying pricing parameters such as strike, interest rate, volatility etc. and the model specific parameters (e.g. mean-reversion level) and sampling uniformly randomly from these inputs. A training set output y (price or hedge ratio in this case) is then calculated using X as input under the relevant models for each derivative. In the case of path-dependent derivatives, y may be time consuming to calculate since a large number of Monte Carlo simulations may need to be run. As previously stated, X is a $d \times n$-matrix, where each d represents the number of input parameters per price y, hence y will be a vector of size n. X and y is then used to train the GPR model. Test set X_* and test outputs y_* are then constructed using the same procedure as for the training set, using different values than used under training. X_* is then used in the GPR to produce y_{GPR} which can be compared to y_* to evaluate the model's accuracy. The measurements typically used to measure accuracy are average absolute error, maximum absolute errors, and root mean square error:

$$\text{AAE} = \frac{1}{n_*}\sum_{i=1}^{n}|y_*(i) - y_{GPR}(i)|,$$

$$\text{MAE} = \max\{|y_*(i) - y_{GPR}(i)|, i = 1, 2, .., n_*\},$$

$$\text{RMSE} = \sqrt{\frac{1}{n_*}\sum_{i=1}^{n_*}(y_*(i) - y_{GPR}(i))^2}.$$

The other area of interest is the increased speed of calculations which is measured by comparing the CPU time for obtaining y_* versus the time for obtaining y_{GPR}.

2.2.1 Derivative Pricing

De Spiegeleer *et al.* (2018) demonstrates the use of GPR in pricing European, American and down-and-out put barrier options. The European option was priced under both the Heston and variance-Gamma model, using a Fast Fourier Transform for each. The American option was priced under a binomial tree model, and the barrier option was priced using a Monte Carlo procedure. All options were tested using a range of training set sizes. The largest MAE of European option was 0.0088 for a training set of 20 000, and the largest AAE was 5.8991e-04 for a training set of 10 000. The speed up of calculations ranged from GPR being 7-40 times faster. For the American option, the largest MAE and AAE were 0.0086 (10 000 samples), and 0.0012 (5000 samples) respectively, and speed-ups ranging from 33-137 times faster. Finally, for the barrier option the largest MAE and AAE were 0.0086 (10 000 samples), and 9.9330e-04 (5000 samples). The speed-up was most significant on the barrier ranging from 2172-11074 times faster. Overall De Spiegeleer *et al.* (2018) makes the case that pricing errors when using GPR are relatively small, but have significant upside in the improved computation time.

Ludkovski (2018) considers the pricing of early-exercise Bermudan put options. GPR is specifically considered in the optimal stopping time problem, and used in place of fixed kernels when estimating continuation values. Though GPR does not yield faster computation time, it decreases the amount of memory required, which Ludkovski notes is of commercial interest since speed is generally more scalable/parallelizable than memory.

Goudenege *et al.* (2020) builds on from Ludkovski (2018) by looking at American basket options using GPR Tree and GPR Exact Integration techniques to approximate continuation value. They conclude that both methods are reliable and speed up computation for large baskets of assets.

Muchabaiwa (2021) considers pricing for down-and-out barrier call options and Asian calls under the Heston model. The MAE and AAE of the barrier is 4.41×10^{-4}, and 1.80×10^{-3} respectively and 3.62×10^{-4} and 1.30×10^{-3} for the Asian, indicating the GPR is reliable for pricing these options. A key feature of GPR is that the improved computational time increases with the number of options in the portfolio. Muchabaiwa (2021) found increased speeds of 10 and 6 times on the barrier and Asian respectively, however for a portfolio of 100 assets the speed up becomes 2000 and 1000 respectively.

2.2.2 Greeks

As an illustrative example, De Spiegeleer *et al.* (2018) explores the use of GPR for the prediction of Gamma for a Cliquet option. Cliquets are typically priced under a Heston model due to their forward volatility sensitivity. In the absence of an analytical solution, calculating Gamma is computationally expensive when looking at multiple prices for the underlying. One solution employed is to calculate Gamma

overnight for a specific set of prices, and then interpolate Gamma for the intraday trading price, referred to as the grid method. Another alternative is to use a standard regression model to estimate Gamma. The analytical Gamma is compared to the predicted Gammas from the grid method, regression, and GPR. It can be seen that GPR produces the closest fit to the analytical solution. It is however noted that GPR performs worse on the boundaries and hence it is recommended that the training set bounds are made slightly larger than the expected range of market parameters.

Crépey and Dixon (2019) also examine the GPR learning of Delta and Vega for a call option against the analytical Black-Scholes, and demonstrates that they closely track. It is specifically noted that these sensitivities are learnt directly from the market parameters used as inputs to the model, and not by inputting the actual sensitivities.

Muchabaiwa (2021) also makes use of GPR to compute Delta and Vega for the down-and-out barrier call options and Asian calls under the Heston model. As previously discussed there were significant computational savings, however only the results for Delta were reasonably accurate, whilst Vega was not found to be reliably estimated via GPR.

2.2.3 Volatility Surfaces

De Spiegeleer *et al.* (2018) also considers GPR in predicting certain points on the volatility surface for various moneyness and maturities. 982 options from the SP 500 were used to train the model with 12 maturities. The resulting predictors were then compared to implied volatilities from the market. The MAE was 0.0077. It is added that what is of importance is also to back-test option trading strategies using the GPR predicted volatility surface, which De Spiegeleer *et al.* (2018) did on delta strangles, although the results here are slightly less promising.

2.2.4 CVA using GPR

Crépey and Dixon (2019) made use of a multi-response GPR in order to estimate the Credit Value at Risk on a client portfolio. xVA estimations require multi-response GPR since they are calculated at the portfolio level, entailing more than one asset. Results showed that CVA estimates were within 0.25% of the Nested Monte Carlo and computed significantly faster. Crépey and Dixon also note that the benefit of GPR in the CVA context is twofold. Firstly it allows for computational efficiency. Secondly, as GPR is a Bayesian approach, given an input point in testing that did not form part of the training set, the uncertainty about the prediction point is readily quantified.

2.2.5 Yield Curve Modelling

Gonzalvez *et al.* (2019) uses a GP-ARX method to predict the US yield curve. ARX is an Autoregressive with Extra Input model, and GP-ARX is the ARX model used in a Gaussian framework. The US yield curve is of particular interest due to its signalling effect across different markets. The curve prediction is tested for 2 year and 10 year predictions against predictions using a standard ARX model, and a random walk process. GP-ARX had the smallest mean-squared error for the 2 year tenor, however the random walk had the smallest mean-squared error for the 10 year tenor. The mean-squared error was not significantly different between each method, however Gonzalvez *et al.* (2019) notes that the added benefit of GP-ARX is that "GP-ARX is able to estimate the confidence interval, and this metric can be used as a trading signal".

Chapter 3

Applications

This dissertation will test the use of GPR in the pricing of an autocallable security. This section describes the features of an autocallable security and the model under which it is typically priced.

3.1 Introduction

Autocallables are structured products, written on one or more underlying assets. The product derives its name from the fact that it has predefined autocall dates, where if the underlying reaches a certain barrier on this date then the product is redeemed or 'called' early, paying out the nominal plus the coupons accumulated to that date (Reder (2005)). These coupons are typically large, and autocallables have gained popularity due to their ability to earn an above market-yield in low-yield environments (De Col and Kuppinger (2017)). Reder adds that the early redemption feature is also attractive to investors because if markets begin to rally, investors' capital becomes available to them to invest elsewhere. In the case of a single underlying (or univariate) autocallable, it is typically written on an index. In the case of multiple underlyings (or multivariate) autocallables, a popular autocallable is the "worst of", where the barrier levels are considered with respect to the worst performing asset in the basket. Typically these structures also contain a Knock-In (KI) feature, whereby if no redemption happens prior to maturity, then the investor will receive the full nominal if the underlying has not moved below the KI barrier throughout the life of the product. If the underlying has moved below the KI barrier then the investor receives relative performance of the underlying. Note however that since these are structured products, many of the features of the product can be customised. For instance coupons and barrier levels can be fixed or vary for different observation dates, or there may be different barrier levels for coupons and redemption.

Autocallables are typically priced under a Stochastic Local Volatility (SLV) model as it better captures the path dependency of the product (Sherif (2019)). Under this model it is difficult to derive an analytical pricing formula for the product. As autocallables require numerical techniques for pricing, it may benefit from the computational savings of GPR pricing technique for a given accuracy trade-off. In what follows, we first discuss the properties and payoff profile of autocallables.

Then we will discuss the Stochastic Local Volatility model and the existing Monte Carlo simulation techniques currently used when pricing under SLV models.

3.2 Autocallables

For the purposes of GPR testing, we will consider a Single Underlying Standard Trigger (SUST) Autocallable as described in Reder (2005).

The SUST Autocallable has the following features:

$$\begin{aligned} t_0 &\quad \text{start date,} \\ t_1 < t_2 < ... < t_m &\quad \text{trigger dates,} \\ T_{final} > t_m &\quad \text{maturity,} \\ P(t_j) &\quad \text{underlying spot level at } t_j. \end{aligned}$$

At trigger date t_j, if relative spot is above or equal to trigger condition X_j, then the structure redeems at $1 + C_j$. We refer to the autocall date as τ, then more formally:

$$\tau = \min_{j \in \{1,...,m\}} \left\{ t_j : \frac{P(t_j)}{P(t_0)} \geq X_j \right\}.$$

If $\tau \neq \emptyset$, the autocallable's payoff A is:

$$A = 1 + C_\tau.$$

If $\tau = \emptyset$ i.e. no early redemption has occurred, then the autocallable payoff with Knock-In barrier B, and final coupon F is as follows:

$$A = F + \frac{P(T_{Final})}{P(t_0)} 1_{\left\{ \frac{P(T_{Final})}{P(t_0)} < B \right\}} + 1_{\left\{ B \leq \frac{P(T_{Final})}{P(t_0)} \right\}}. \tag{3.1}$$

Intuitively, Hansson (2012) describes these payoffs informally as zero-coupon bonds with stochastic maturity. Additionaly, Reder (2005) and Lee *et al.* (2019) obtain analytical solutions for a univariate autocallable, however they do so under the Black-Scholes model. We proceed in the pricing of an autocallable by generating sample paths of the underlying's spot level $P(t_j)$ using a stochastic local volaility model, and then taking the discounted payoff of each sample path according to the above payoff equations. Further details of the stochastic local volatility model are given in the next section.

3.3 Stochastic Local Volatility model

The local volatility (LV) model is a widely used pricing model, with its main advantage being precise calibration to any given set of arbitrage-free European vanilla option prices via the Dupire formula. This is achieved as the implied Black-Scholes

volatility surface is used as input to the model (Van der Stoep *et al.* (2014)). However, this model generates flattening implied forward volatilities, meaning that it will potentially misprice products that are sensitive to forward volatility.

An alternative class of model considered to address the shortcomings of the LV model is the class of stochastic volatility (SV) models. Two of the most common SV models are Heston and Schobel-Zhu. The SV models consider volatility as a separate process with its own stochastic differential equation. SV models however often cannot be calibrated to the same precision as LV models. The shortcomings of the LV and SV models informs the proposal of a hybrid model known as a stochastic local volatility (SLV) model.

The stochastic differential equations (SDEs) for the SLV model class, as characterised in Van der Stoep *et al.* (2014), are as follows:

$$\frac{dS(t)}{S(t)} = rdt + \sigma(t, S(t))\psi(V(t))dW_x(t),$$

$$dV(t) = a_\nu(t, V(t))dt + b_\nu(t, V(t))dW_\nu(t),$$

$$dW_x(t)dW_\nu(t) = \rho_{x,\nu}dt,$$

where $\rho_{x,\nu}$ is the correlation between the corresponding Brownian motions, $\sigma(t, S(t))$ is the local volatility component, $\psi(V(t))$ controls the stochastic volatility, $a_\nu(t, V(t))$ is the drift and $b_\nu(t, V(t))$ is the diffusion of the variance process, and r is a constant interest rate.

Note, if $\sigma(t, S(t)) = 1$ then the SLV model is equivalent to a pure stochastic volatility model. Conversely, if $b_\nu(t, V(t)) = 0$ then the SLV model is equivalent to a local volatility model.

The aforementioned stochastic volatility models have the following variance processes:

For the Heston SV model the variance process is driven by the CIR dynamics:

$$\psi(V(t)) = \sqrt{V(t)},$$

$$a_\nu(t, V(t)) = \kappa(\overline{v} - V(t)),$$

$$b_\nu(t, V(t)) = \gamma\sqrt{V(t)}.$$

For the Schobel-Zhu model:

$$\psi(V(t)) = V(t),$$

$$a_\nu(t, V(t)) = \kappa(\overline{v} - V(t)),$$

$$b_\nu(t, V(t)) = \gamma,$$

where κ is the mean reversion parameter, $\overline{v}$ is a long-term mean and γ is the volatility of the process $V(t)$.

The $\sigma(t, S(t))$ can have various forms. A common form is the constant elasticity of variance, which describes volatility movements in terms of the underlying asset as seen below:

$$\sigma(t, S(t)) = \hat{\sigma} S^{\beta}(t),$$

where when $\psi = 1$, $\beta = 1$ corresponds to Geometric Brownian motion, $\beta = 0.5$ corresponds to a square root process (similar to CIR process, without mean reversion), and $\beta = 0$ corresponds to a process similar to the Ornstein-Uhlenbeck process, without mean reversion.

A Euler discretization can be used to simulate the sample paths for the SLV model:

$$s_{i+1,j} = s_{i,j} + r s_{i,j} \triangle + \sigma(t_i, s_{i,j}) s_{i,j} \psi(\nu_{i,j}) \sqrt{\triangle} Z_{x,i,j}, \tag{3.2}$$

$$\nu_{i+1,j} = \nu_{i,j} + a_\nu(t_i, \nu_{i,j}) \triangle + b_\nu(t_i, \nu_{i,j}) \sqrt{\triangle} Z_{\nu,i,j}, \tag{3.3}$$

where:

$$s_{0,j} = S(t_0),$$
$$\nu_{0,j} = \nu(t_0),$$

$i = 1, .., M$ where M is the number of timesteps,
$j = 1, .., N$ where N is the number of Monte Carlo paths,

$\triangle$ is the equidistant time-step given by $\triangle = \dfrac{T}{M}$, and T is the maturity time.

$Z_{1,i,j}$ and $Z_{2,i,j}$ are independent standard normal variables for $i = 1 : M$, $j = 1 : N$. Then:

$$Z_{x,i,j} = Z_{1,i,j},$$

$$Z_{\nu,i,j} = \rho_{x,\nu} Z_{1,i,j} + \sqrt{(1 - \rho_{x,\nu}^2)} Z_{2,i,j}.$$

For the purpose of this research, the discretization is used to generate N sample paths over M timesteps of the underlying $s_{i,j}$ (which requires $\nu_{i,j}$ as an input). The payoff of the autocall derivative is then calculated at each timestep using $s_{i,j}$, and discounted to t_0, representing the derivative price for this particular sample path.

In the case of calculating the payoff for an autocall, this requires for a given path at each timestep first evaluating if the option has already been knocked in i.e. if the underlying has breached the barrier at previous timesteps, in which case the payoff will be 0 for that timestep. If the path has not previously been knocked in, then the underlying for the current timestep is evaluated against the barrier and if it has breached then the payoff is $1 + C_j$, else the payoff is 0. This evaluation will continue iteratively through each timestep, until the final timestep. If the option has not been knocked-in at the final timestep, the payoff will be as described in (3.1).

Using the same parameters as for simulation, a price can be found under GPR, and compared for accuracy and speed of calculation.

Chapter 4

Results

Results were obtained in MatLab R2020b, using a HP 15s Intel(R) Core(TM)i7 Laptop with CPU speed of 2.80GHz and 20GB of RAM.

All autocallable testing is parameterised according to Lee *et al.* (2022) with maturity $T = 3$, with call dates every 6 months and 6 autocall barriers $\{0.9\ 0.9\ 0.85\ 0.85\ 0.8\ 0.8\}$. The Knock-In barrier is 0.7, and coupon is 0.06. For ease of implementation, when testing under a Stochastic Local Volatility model we restrict to the simplest case, assuming a constant elasticity of variance local volatility process with $\beta = 1$ and $\psi = 1$, in order to correspond to a Geometric Brownian Motion. Additionally, calibrated market data for more complex dynamics were not readily available.

The kernel hyperparameters are tuned using Matlab's built-in GPR fitting function which maximises the log-likelihood function. The squared exponential kernel function, as seen in (2.1), is the most commonly used in Quantitative Finance, and for this reason is used throughout training and testing. In all cases the testing set is chosen to be within the training set bounds as GPR generally does not perform well near the boundary of the training set (De Spiegeleer *et al.* (2018)).

4.1 Autocallable under Black-Scholes Model

To begin, we test the efficacy of GPR in pricing an autocallable under Black-Scholes pricing assumptions.

The training set is specified by randomly sampling 10,000 points from the parameter ranges shown in table 4.1.

Parameter	Minimum	Maximum
r	0.018	0.025
σ	0.08	0.2

Tab. 4.1: Training parameters for autocallable under Black-Scholes.

The parameters are then used to create a set of 10,000 autocallable prices. Based on the training set, the hyperparameters are $l = 6.63381$ and $\sigma_f = 0.70895$.

The same procedure is then used to create a test set by randomly sampling from the parameter ranges shown in table 4.2.

Parameter	Minimum	Maximum
r	0.019	0.024
σ	0.1	0.19

Tab. 4.2: Testing parameters for autocallable under Black-Scholes.

The results for GPR against Black-Scholes testing are shown in table 4.3.

Sample Size	1,000	5,000	10,000	20,000
AAE	3.94×10^{-7}	4.01×10^{-7}	3.99×10^{-7}	3.98×10^{-7}
MAE	8.39×10^{-7}	8.55×10^{-7}	8.74×10^{-7}	8.58×10^{-7}
RMSE	4.32×10^{-7}	4.37×10^{-7}	4.36×10^{-7}	4.36×10^{-7}

Tab. 4.3: Test results for autocallable under Black-Scholes.

GPR predicts the autocall's price accurately, however as would be expected, there was no speed up in calculation observed by using GPR. A visual representation of the results is shown in figure 4.1 and figure 4.2 for the testing sample of 1,000.

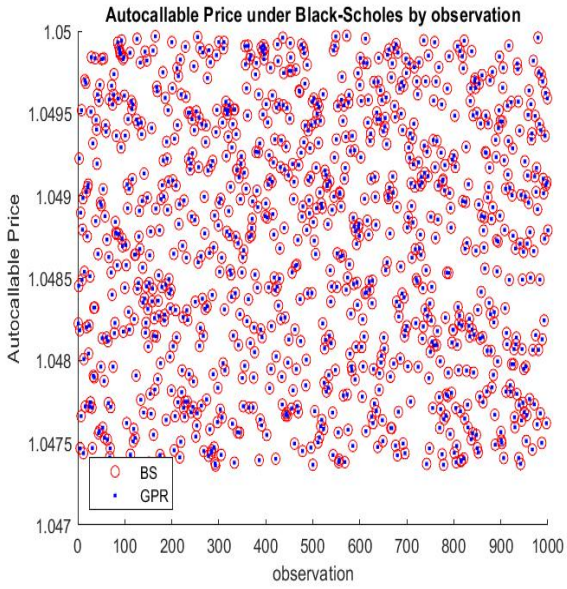

Fig. 4.1: Autocallable prices under Black-Scholes for 1000 test points.

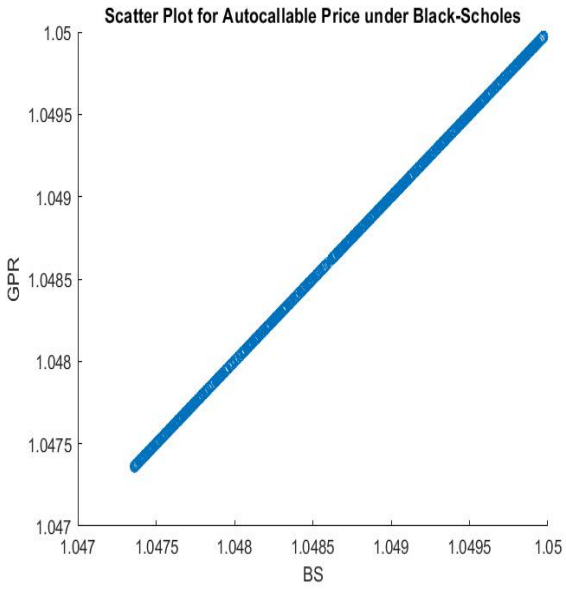

Fig. 4.2: Scatter plot of autocallable prices under Black-Scholes vs GPR.

4.2 Heston Stochastic Local Volatility Model

Engelmann *et al.* (2021) calibrates a Heston Stochastic Local Volatility model to DAX data as of 29/01/2020, The resulting parameters are shown in table 4.4

	v_0	$\overline{v}$	κ	γ	ρ
DAX	0.148	0.044	0.877	0.377	-0.778

Tab. 4.4: Heston SLV calibrated parameters.

Note that the above calibration and assumed parameters results in a pure Heston (i.e. Stochastic volatility) model, however the framework presented below allows for further testing of more complex dynamics beyond the scope of this research. Additionally, Engelmann *et al.* (2021) notes that the Feller condition is violated and states that this is typically the case when working with real-world date, however the set should still be used with care.

We train the GPR model by randomly sampling from a 20% range around the above calibrated parameters. The resulting training parameter ranges are in table 4.5.

A testing set is also constructed by randomly sampling from a 10% range around the calibrated parameters. The resulting testing parameter ranges are in table 4.6

10,000 samples paths are used to train the GPR model. The paths are created using

Parameter	Minimum	Maximum
v_0	0.1184	0.1776
$\overline{v}$	0.0352	0.0528
κ	0.7016	1.0524
γ	0.3016	0.4524
ρ	-0.9336	-0.6224
r	0.018	0.025

Tab. 4.5: Training parameters for Heston SLV.

Parameter	Minimum	Maximum
v_0	0.3393	0.4147
$\overline{v}$	0.0396	0.0484
κ	0.7893	0.9647
γ	0.3393	0.4147
ρ	-0.8558	-0.7002
r	0.019	0.024

Tab. 4.6: Testing parameters for Heston SLV.

the Euler discretization for the Heston SLV model as shown in (3.2) and (3.3).

4.2.1 European Call under Heston SLV model

In this section we consider GPR in pricing a European Call under the Heston Stochastic Local Volatility model. Based on the training set, the hyperparameters are $l = 8.238572$ and $\sigma_f = 5.649509$.

The results for GPR against SLV Monte Carlo testing for the European call are shown in table 4.7. A visual representation of the results is shown in figure 4.3 and figure 4.4 for the testing sample of 1,000. The resulting price predictions are poor, and no improvements in calculation time were observed under GPR.

Sample Size	1,000	5,000	10,000	20,000
AAE	0.018676918	0.018981114	0.019153206	0.018997121
MAE	0.131799827	0.212562508	0.179007337	0.254275841
RMSE	0.024465438	0.025179955	0.025210616	0.025131859

Tab. 4.7: Results for European call under Heston SLV.

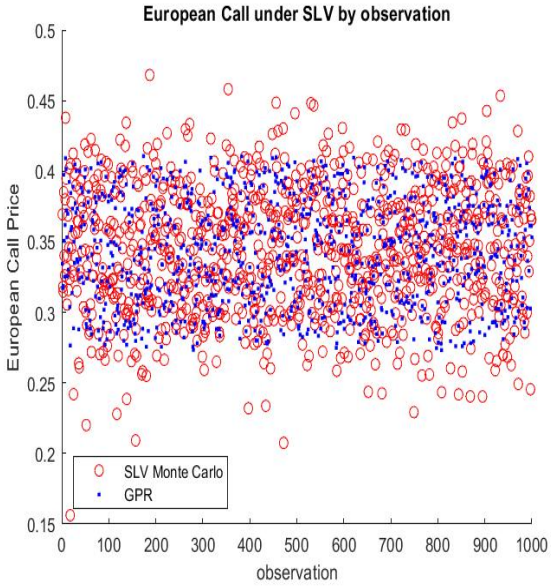

Fig. 4.3: European call prices under Heston SLV for 1000 test points.

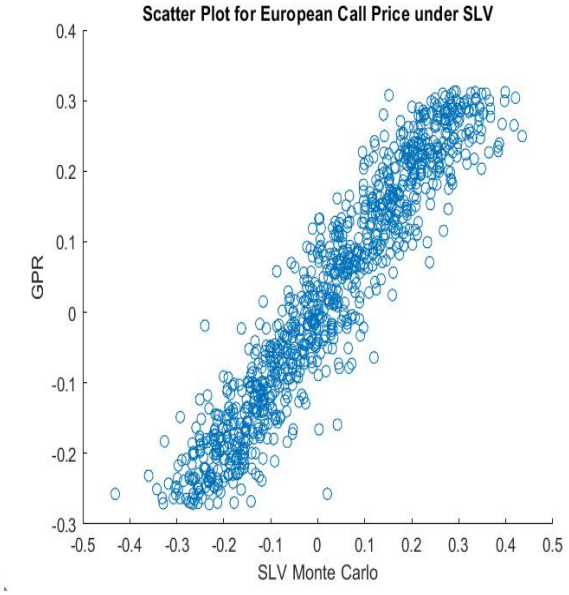

Fig. 4.4: Scatter plot of European call prices under Heston SLV vs GPR.

4.2.2 Autocallable under Heston SLV model

Now we consider the pricing of an autocallable under the Heston SLV model. Based on the training set, the hyperparameters are $l = 0.098687$ and $\sigma_f = 2.824079 \times 10^{-6}$.

The results for GPR against SLV Monte Carlo testing for the autocallable price are shown in table 4.8.

Sample Size	1,000	5,000	10,000	20,000
AAE	0.000774458	0.000776328	0.000762331	0.000766849
MAE	0.001855233	0.001857526	0.001857386	0.001857449
RMSE	0.000943243	0.000940082	0.000925509	0.000930788

Tab. 4.8: Results for autocallable under Heston SLV.

A visual representation of the results is shown in figure 4.5 and figure 4.6 for the testing sample of 1,000.

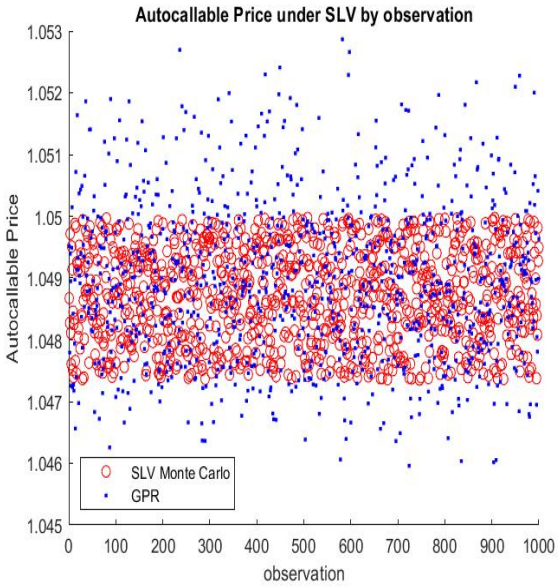

Fig. 4.5: Autocallable prices under SLV for 1000 test points.

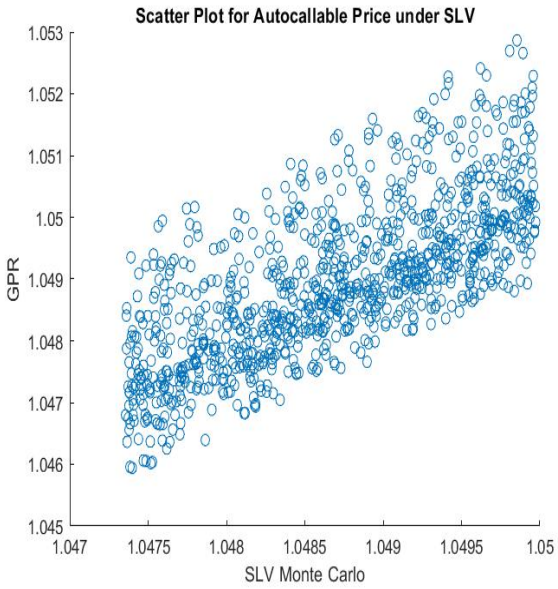

Fig. 4.6: Scatter plot of autocallable prices under SLV vs GPR.

Additionally, the computation times obtained are found in table 4.9.

Sample Size	1,000	5,000	10,000	20,000
SLV Monte Carlo	0.04652	0.280441	0.873927	3.298438
GPR	0.062289	0.350048	0.794245	1.535889

Tab. 4.9: Computation time for autocallable under Heston SLV.

GPR performs moderately in predicting the autocallable price under the Heston SLV model. The computation time performs well, outperforming Monte Carlo simulations at the $10,000$ sample size, but magnified to twice as fast at $20,000$ sample size.

4.3 Schobel-Zhu Stochastic Local Volatility Model

The same training and testing procedure, as well as the parameters seen in table 4.4 are used for the Schobel-Zhu SLV model, as was done for the Heston model. A limitation that should be noted is that it is not clear whether using the same calibrated parameters will produce realistic market dynamics, however a calibration of the Schobel-Zhu model was not readily available. Once again, the choice of parameters results in a reduced form of the model which simplifies the local volatility component. The Euler discretization is also used once again. The results are presented below.

4.3.1 European call under Schobel-Zhu SLV model

In this section we consider GPR in pricing a European call under the Schobel-Zhu Stochastic Local Volatility model. Based on the training set, the hyperparameters are $l = 3.818246$ and $\sigma_f = 1.402343$.

The results for GPR against SLV Monte Carlo testing for the European Call are shown in table 4.10. A visual representation of the results is shown in figure 4.7 and figure 4.8 for the testing sample of 1,000. GPR predictions are poor, and there was no improvement in computation time.

Sample Size	1,000	5,000	10,000	20,000
AAE	0.078840208	0.077799241	0.077634409	0.078392223
MAE	0.397141764	0.515024489	0.547492473	0.633817927
RMSE	0.103980626	0.104027089	0.103884214	0.104743907

Tab. 4.10: Results for European call under Schobel-Zhu SLV.

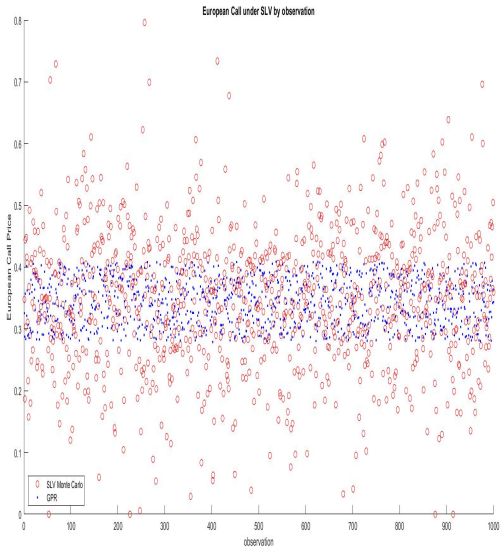

Fig. 4.7: European call prices under Schobel-Zhu SLV for 1000 test points.

4.3.2 Autocallable under Schobel-Zhu SLV model

Finally we consider the pricing of an autocallable under the Schobel-Zhu SLV model. Based on the training set, the hyperparameters are $l = 100.275091$ and $\sigma_f = 18.527187$. The results for autocallable prices under Schobel-Zhu SLV model for GPR against Monte Carlo are shown in table 4.11, and presented visually in figure 4.9 and figure 4.10. Additionally, the computation times obtained are found in table 4.12.

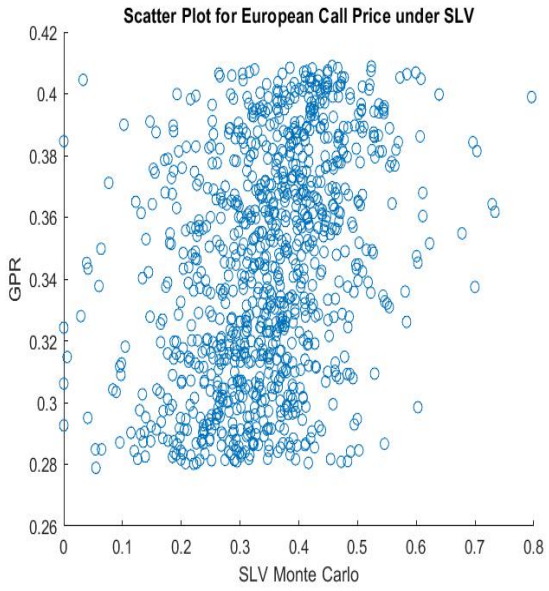

Fig. 4.8: Scatter plot of European call prices under Schobel-Zhu SLV vs GPR.

Sample Size	1,000	5,000	10,000	20,000
AAE	5.845×10^{-7}	5.860×10^{-7}	5.873×10^{-7}	5.848×10^{-7}
MAE	1.253×10^{-6}	1.253×10^{-6}	1.262×10^{-6}	1.263×10^{-6}
RMSE	6.488×10^{-7}	6.434×10^{-7}	6.461×10^{-7}	6.450×10^{-7}

Tab. 4.11: Results for autocallable under Schobel-Zhu SLV.

Sample Size	1,000	5,000	10,000	20,000
SLV Monte Carlo	0.011999	0.054279	0.108279	0.15404
GPR	0.019281	0.06217	0.135738	0.227917

Tab. 4.12: Computation time for autocallable under Schobel-Zhu SLV.

GPR predictions for the autocallable under Schobel-Zhu are accurate. The computation time for the autocallable under the Schobel-Zhu SLV is however slightly slower than the Monte Carlo simulations, but closely competitive.

Of interest is the improved accuracy under Schobel-Zhu over Heston. A key difference between the two models is the inclusion of the $\sqrt{(V(t))}$ term in Heston, ensuring that $V(t)$ is strictly positive. Given that the Feller condition was violated in the calibration set, the parameters may have led to better performance of Schobel-Zhu which allows for negative $V(t)$, though this does not represent realistic market dynamics. GPR under Heston may have performed better under calibration set where the Feller condition was met.

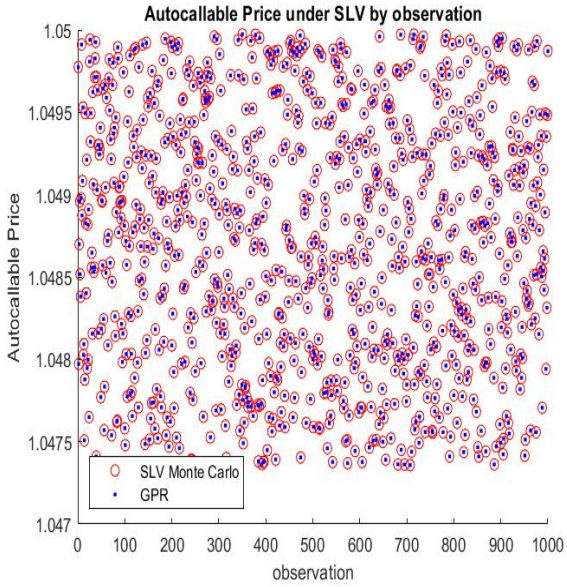

Fig. 4.9: Autocallable prices under Schobel-Zhu SLV for 1000 test points.

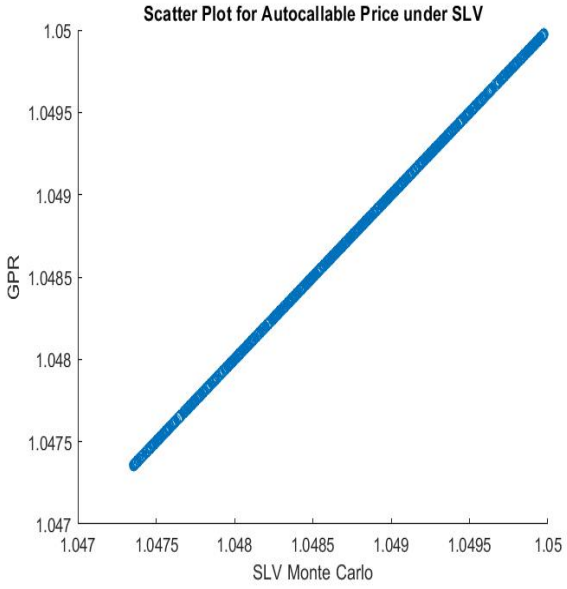

Fig. 4.10: Scatter plot of autocallable call prices under Schobel-Zhu SLV vs GPR.

The improved computation time for the autocallable under the Heston SLV and not on the Schobel-Zhu SLV, is likely due to a more time consuming evaluation of the volatility control (ψ) and diffusion variance (b_ν) functions.

Chapter 5

Conclusion

This dissertation implemented GPR with a squared-exponential kernel to price a Single Underlying Security Autocallable under the Black-Scholes model, the Heston Stochastic Local Volatility model, and the Schobel-Zhu Stochastic Local Volatility model. It was also implemented in pricing a European call option under the Heston Stochastic Local Volatility model and the Schobel-Zhu Stochastic Local Volatility model.

GPR performs best in pricing the Black-Scholes autocallable, however has no computation savings as expected. GPR also performs well in pricing an autocallable under Schobel-Zhu SLV model, however is computationally slightly slower. GPR performs poorly in pricing the European call under both Heston and Schobel-Zhu SLV models. GPR performs moderately in pricing an autocallable under the Heston SLV model. Although computation time was improved under Heston SLV, the accuracy is not likely high enough to feasibly be implemented in reality.

Further consideration should be given to testing GPR using calibrated parameters sets where the Feller condition is met as this may lead to more accurate results, particularly under the Heston model. Additionally, this research considered the one-dimensional GPR case, however the introduction of a second underlying stochastic process may warrant the use of two-dimensional GPR. Other avenues for improved accuracy could also be explored, such as alternate kernel functions, or larger training sets. Given the accuracy of the autocallable under Schobel-Zhu, a further possible avenue of research may be the implementation of GPR in pricing multivariate autocallables where the computation time may outperform Monte Carlo simulations.

Bibliography

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, Vol. 4, Springer.

Crépey, S. and Dixon, M. (2019). Gaussian Process Regression for derivative portfolio modeling and application to CVA computations, *arXiv preprint arXiv:1901.11081* .

De Col, A. and Kuppinger, P. (2017). The interplay between stochastic volatility and correlations in equity autocallables, *Available at SSRN 3228065* .

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*, Springer.

Engelmann, B., Koster, F. and Oeltz, D. (2021). Calibration of the Heston stochastic local volatility model: A finite volume scheme, *International Journal of Financial Engineering* **8**(01): 2050048.

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

Goudenege, 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.

Hansson, F. (2012). *A pricing and performance study on auto-callable structured products*, Master's thesis, KTH Royal Institute of Technology Ecoledes Ponts Paris Tech.

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

Lee, H., Ahn, S. and Ko, B. (2019). Generalizing the reflection principle of Brownian motion, and closed-form pricing of barrier options and autocallable investments, *The North American Journal of Economics and Finance* **50**: 101014.

Lee, H., Lee, M. and Ko, B. (2022). A semi-analytic valuation of two-asset barrier options and autocallable products using Brownian bridge, *The North American Journal of Economics and Finance* **61**: 101704.

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

Muchabaiwa, T. M. (2021). *Applications of Gaussian Process Regression to the pricing and hedging of exotic derivatives*, Master's thesis, Faculty of Commerce.

Reder, R. (2005). *Auto trigger securities: Closed-form solutions and applications*, Diplomarbeit, Technische Universität München.

Schöbel, R. and Zhu, J. (1999). Stochastic volatility with an Ornstein–Uhlenbeck process: an extension, *Review of Finance* **3**(1): 23–46.

Sherif, N. (2019). Calling out autocallable pricing.
URL: *https://www.risk.net/our-take/6348336/calling-out-autocallable-pricing*

Van der Stoep, A. W., Grzelak, L. A. and Oosterlee, C. W. (2014). The Heston stochastic-local volatility model: efficient Monte Carlo simulation, *International Journal of Theoretical and Applied Finance* **17**(07): 1450045.

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, Vol. 2, MIT press Cambridge, MA.

Wilson, A. and Adams, R. (2013). Gaussian process kernels for pattern discovery and extrapolation, *International conference on machine learning*, pp. 1067–1075.