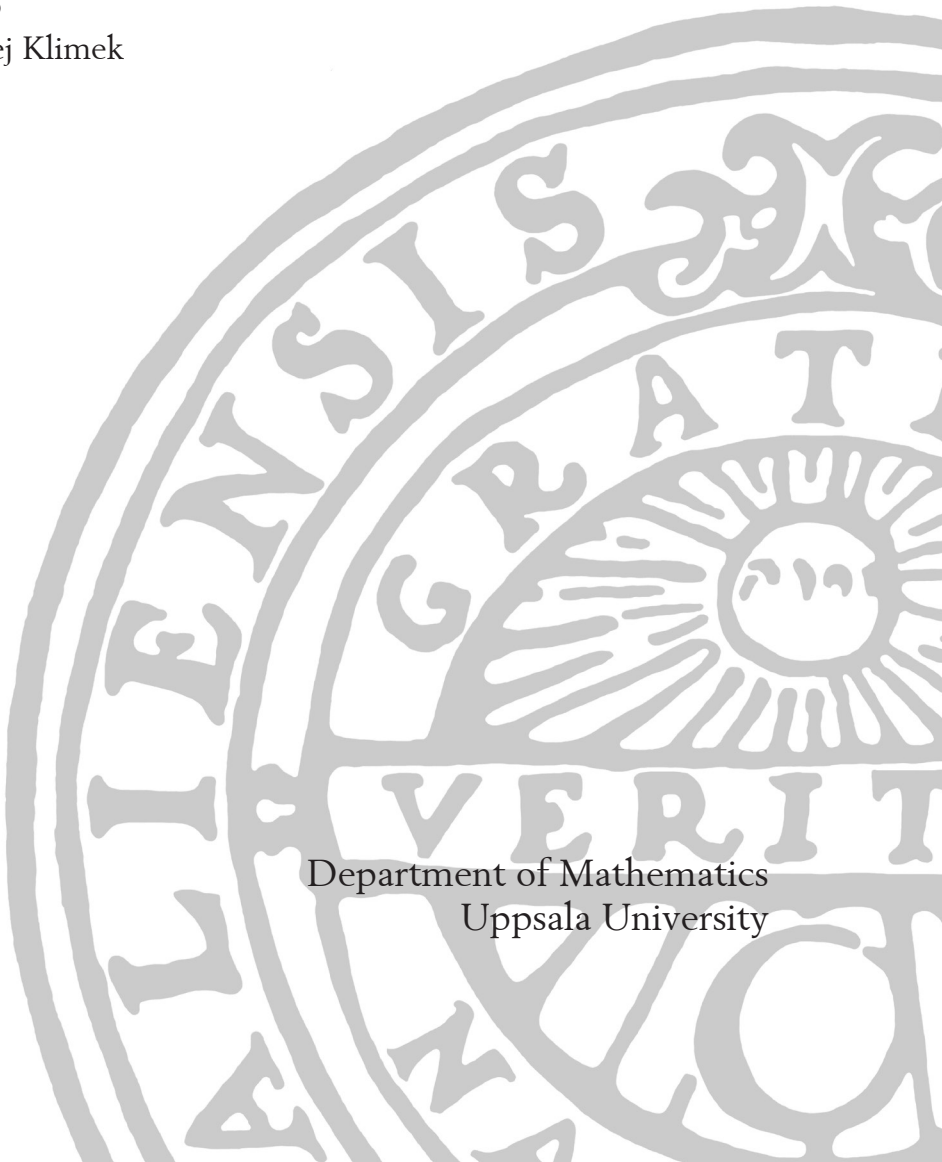
UPPSALA
UNIVERSITET

U.U.D.M. Project Report 2020:2

Gaussian Process Regression In Computational Finance

Hugues Herfurth

Examensarbete i matematik, 30 hp
Handledare och examinator: Maciej Klimek
Januari 2020



Department of Mathematics
Uppsala University

Abstract

Machine learning can be deployed not only in order to solve non trivial problems, but also faster than traditional implemented solutions. We illustrate several classical problems in the field of computational finance where it is possible to fit, with Gaussian process regressions, complex functions under and beyond the Black-Scholes model with high accuracy. The results from the regressions, in our examples, show speed-ups of several magnitudes compared to the classical implementations while keeping a precision well within the acceptable limits for practical use. The concrete examples consist in financial Greeks fitting, summarizing implied volatility surfaces, as well as estimating vanilla and exotic options while reducing the computation time.

Keywords – Machine learning; Gaussian process; Derivative pricing; Volatility surface

Acknowledgements

I would like to thank my supervisor, Professor Maciej Klimek, for his precious time and guidance during this project. I would also like to thank my family for their relentless support through all these years of studying.

Contents

List of Figures

List of Tables

1 Introduction

The concept of Gaussian Processes leads to a supervised learning method aimed at solving regression and probabilistic classification problems, and as such they are extensively used for for modeling dependent data. As they are statistical models, they interpolates the observations and induce probabilistic predictions which give empirical confidence intervals which are very practical in order to refine and adapt the fitting in the areas of interest. They are also extremely versatile as a diversity of kernels can be specified in order to reach desired functions characteristics such a smoothness.

The interest towards these processes resides in their inheritance properties. Thanks to this, they can be used on small samples and the regression process has implementation practicalities when fitting the model as one does not need to define higher than second order moments. Furthermore, this regression approach is non parametric, as it is theoretically possible to draw from an infinite set of functions, which is very convenient when adjusting kernels. Nevertheless, this method is not exempted of potential under or over fitting problems and has to be adjusted when the output is non Gaussian (e.g. binary). They can also suffer from a lack of accuracy when used in high dimensional spaces, typically when the number of features exceeds several dozens, but these limitations are not reached in our area of interest since the Heston model and binomial tree incorporate less than ten parameters.

In this thesis, I present the theory and numerical tools in order to implement such a regression in the field of computational finance. We discuss popular model and implementation for pricing several financial derivatives and compare their performance against a wisely fitted Gaussian process. The theoretical content is following the notation and logic from Rasmussen [see 12], and the diverse implementation followed the logic of De Spiegeleer [see 14]. The implementations were written in Python and Matlab, making use respectively of the sklearn library and the financial toolbox.

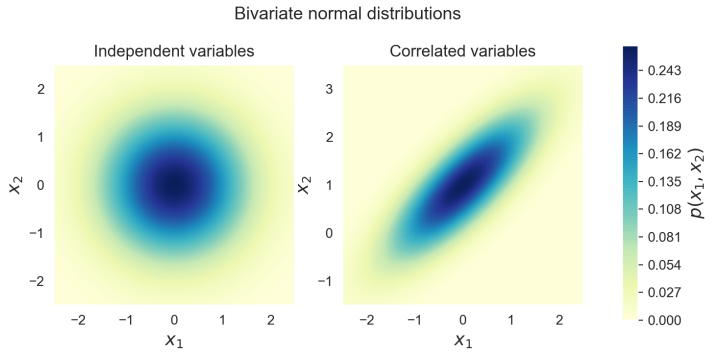


Figure 1.1: Bi-variate Gaussian

1.1 Multivariate Gaussian

The multivariate Gaussian distribution, or joint Gaussian distribution, is a multi-dimension generalization of the one-dimensional normal distribution. It represents the distribution of a multivariate random variable that is made up of multiple random variables that can be correlated with each other. An illustration can be seen in Figure 1.1 for a 2d distribution.

Like the normal distribution, the multivariate normal is defined by a set of parameters: the mean vector, which is the expected value of the distribution; and the covariance matrix, which measures how dependent two random variables are.

The joint probability density function for a distribution of dimentionality d is defined as:

$$f(x \mid \mu, \Sigma) = \frac{1}{\sqrt{\mid \Sigma \mid (2\pi)^d}} \exp\left(-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)'\right), \tag{1.1}$$

Where x is a random vector of size d, with μ as the mean vector and Σ is symmetric and semi-definite covariance matrix of size $d \times d$, and $\mid \Sigma \mid$ its corresponding determinant. Such a distribution of often denoted as $N(\mu, \Sigma)$.

This type of distribution has very important properties for the use of Gaussian processes:

- The sum of jointly Gaussians is Gaussian
- An affine transformation of a Gaussian is Gaussian
- The marginal distribution of a multivariate Gaussian is Gaussian
- The conditional distributions of a Gaussian are Gaussians

A graphical illustration of these two properties is shown in Figure 1.2.

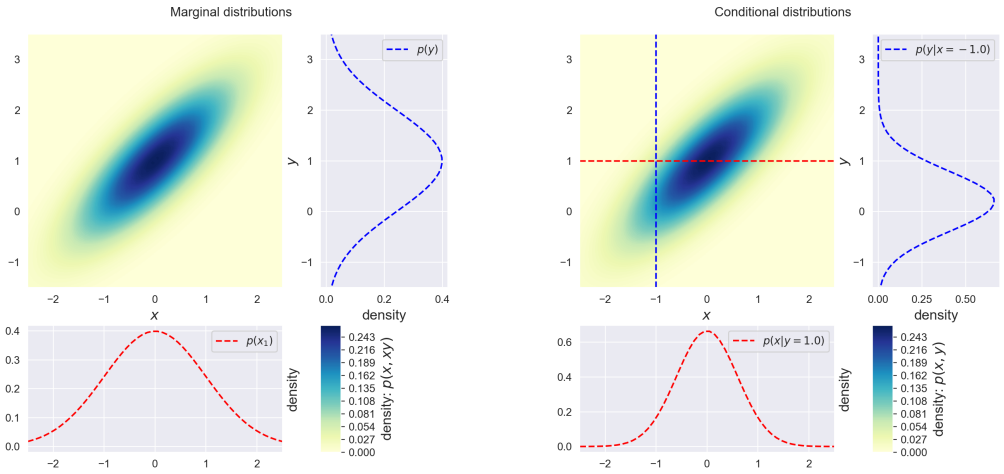


Figure 1.2: Bi-variate Gaussian marginal (a) and conditional (b) distributions

1.2 Bayesian Regression

In the context of Bayesian regression, we assume that we are in the case where we can describe the observed data $y = (y_1, ..., y_n)$ from the input $x = (x_1, ..., x_n)$ as:

$$y = f(x) + \epsilon, \tag{1.2}$$

where f is a Gaussian process and $\epsilon = (\epsilon_1, ..., \epsilon_n)$ are i.i.d random variables representing the noise in the data such as $\epsilon \sim N(0, \sigma_n^2)$.

1.2.1 The linear approach

If we were to take the linear Bayesian approach, we would have f as:

$$f(x) = x^T w, \tag{1.3}$$

with $w \sim N(0, \Sigma_p)$ a $n \times 1$ vector of prior weights. Then, using Bayes' rule we would get for the posterior weights, wiht X the vector of inputs:

$$p(w|y, X) = \frac{p(y|X, w)p(w)}{p(y|X)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{marginal likelihood}}, \tag{1.4}$$

with

$$\begin{aligned} p(y|X,w) &= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\left(-\frac{1}{2\sigma_n^2}|y - X^T w|^2\right) = N(X^T w, \sigma_n^2 I) \\ p(y|X) &= \int p(y|X,w)p(w)dw. \end{aligned} \tag{1.5}$$

This leads to a Gaussian for the posterior distribution of the form:

$$p(w|y,x) \sim N(\frac{1}{\sigma_n^2}A^{-1}xy, A^{-1}), \quad A = \sigma_n^2 xx^T + \Sigma_p^{-1} \tag{1.6}$$

Then, after obtaining the posterior weights, we can use them on non observed data x_* to get the predictive distribution of the output f_* which is also Gaussian:

$$p(f_*|x_*,x,y) = N(\frac{1}{\sigma_n^2}x_*^T A^{-1}xy, x_*^T A^{-1}x_*). \tag{1.7}$$

1.2.2 Linear to non linear

In order to express non linear effects that would be impossible to transcribe with the previous regression scheme, we can project the inputs into a feature space. This can be done with a set of basis functions, e.g. power functions which would lead to a polynomial regression. That is to say, with a set of basis functions represented by the column vector $\phi = (\phi_1, ..., \phi_n)$:

$$f(x) = \phi(x)^T w \tag{1.8}$$

With very similar steps as before, and by noting $\Phi(X)$ the aggregation of $\phi(x)$ for all cases in the training set, it leads to the following predictive distribution:

$$p(f_*|x_*,X,y) \sim N(\frac{1}{\sigma_n^2}\phi(x_*)^T A^{-1}\Phi(X)y, \phi(x_*)^T A^{-1}\phi(x_*)), \tag{1.9}$$

which, for later implementation performance, can be re-arranged using the Woodbury matrix identity property [15] and be written as:

$$p(f_*|x_*,X,y) \sim N(\phi_*^T\Sigma_p\Phi(K + \sigma_n^2 I)^{-1}y, \phi_*^T\Sigma_p\phi_* - \phi_*^T\Sigma_p\Phi(K + \sigma_n^2 I)^{-1}\Phi^T\Sigma_p\phi_*), \tag{1.10}$$

by expanding $A = \sigma_n^2 XX^T + \Sigma_p^{-1}$ and writing $\phi_* = \phi(x_*)$, $\phi(x) = \phi$, $\Phi = \Phi(X)$ and $K = \Phi\Sigma_p\Phi$.

1.2.3 The kernel trick

We saw that the terms in the predictive distribution are projected in the function space by matrices of the form $\phi(x)^T\Sigma_p\Phi(x')$ where x and x' are respectively observed and non observed inputs.

Having defined Σ_p as positive definite, we can rewrite the inner product by using singular value decomposition as:

$$\phi(x)^T\Sigma_p\Phi(x') = \Sigma_p^{1/2}\phi(x) \cdot \Sigma_p^{1/2}\phi(x'). \tag{1.11}$$

By defining $\psi(x) = \Sigma_p^{1/2}\phi(x)$, we can then define a covariance function, or rather kernel function k, such as:

$$k(x, x') = \psi(x) \cdot \psi(x') \tag{1.12}$$

This procedure reduces drastically the computational power required to compute the inner product, and enables us to work directly with inputs in the function space.

1.2.4 The choice of kernel

The kernel function can be any valid Mercer's kernel [see 10], i.e. a positive definite function, that is to say a function k for which any subset $x \in X$ leads to a positive semi-finite covariance matrix $K(x, x')$. That is to say, for any $v \in \mathbb{R}^D$:

$$v^T K(x, x')v = \sum_i \sum_j k(x_i, x_j)v_i v_j \geq 0. \tag{1.13}$$

A commonly used covariance function is the quadratic exponential kernel:

$$k(x, x', l) = \sigma^2 exp\left(\frac{-\mid x - x' \mid^2}{2l^2}\right), \tag{1.14}$$

with σ and l being hyper parameters. This can be equivalently described as a Bayesian regression with prior weights $w \sim N(0, \sigma_p^2 I)$ and an infinite set of basis functions of the form:

$$\phi(x, l) = \exp\left(-\frac{(x - c)^2}{2l^2}\right) \tag{1.15}$$

An example of covariance matrix from the quadratic exponential kernel (also known as the RBF kernel) covariance is shown in Figure 1.3. We see that this quadratic covariance decreases exponentially the further away the function values are from each-other.

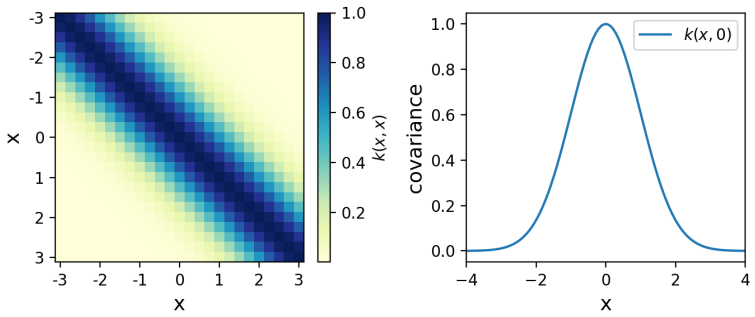


Figure 1.3: Example of exponential quadratic covariance matrix (a) and covariance $k(x, 0)$ (b)

1.3 Gaussian Process

A Gaussian process is a stochastic process where every finite subset of its collection of random variables has a multivariate normal distribution. That is to say, for an index set X, a real-valued stochastic process $\{f(x), x \in X\}$ is a Gaussian process if, for any subset $x = (x_1, ..., x_n) \in X$, $f(x)$ has a joint Gaussian distribution. It is then completely described by its mean function m and its covariance function k:

$$f \sim N(m(x), K(x, x')) \tag{1.16}$$

where $K(x, x')$ is the covariance matrix with entries $K_{i,j} = k(x_i, x'_j)$. In other terms, a Gaussian process can be understood as a multivariate Gaussian with an uncountable infinite number of random variables.

The mean function m can be any real-valued function, and is very often set to zero by subtracting the mean from the data. The kernel function k can be any valid Mercer's kernel, such as the exponential squared kernel shown before. This way, a Gaussian process is often written as:

$$f(x) \sim GP(m(x), k(x, x')). \tag{1.17}$$

To sample functions from the Gaussian process we just need to define the mean and covariance functions. The covariance function k models the joint variability of the Gaussian process random variables. It returns the modelled covariance between each pair of inputs. That is to say, with f and f_* being respectively the training and test outputs, we have the following joint distribution:

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \tag{1.18}$$

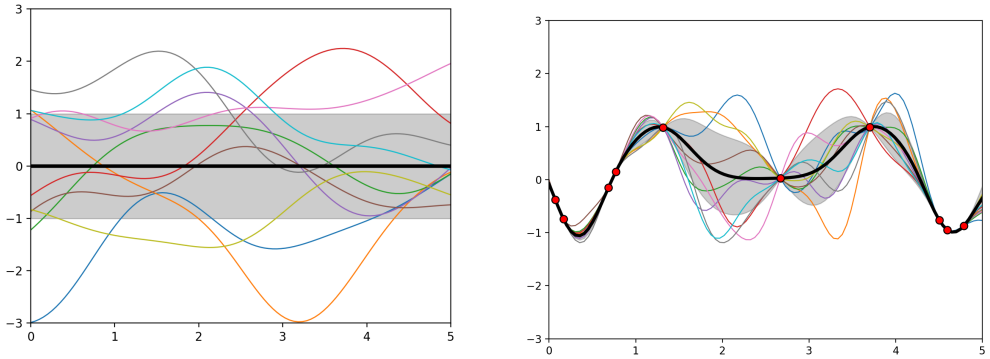


Figure 1.4: Example of GP functions sample for the prior (a) and post (b) distributions

As we have seen, the specification of this covariance function, the kernel function, implies a distribution over functions. By choosing a specific kernel function it is possible to set prior information on this distribution. We can sample function evaluations of a function drawn from a Gaussian process at a finite but arbitrary set of points. Thanks to the properties of Gaussian processes, we can evaluate the posterior by conditioning the joint Gaussian prior distribution on the observation:

$$\begin{aligned}(f_* \mid X_*, X, f) \sim N(&K(X_*, X)K(X, X)^{-1}f, \\ &K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)),\end{aligned} \tag{1.19}$$

we can go through the following steps to draw function samples:

- Find a matrix A such as $\Sigma = AA^T$, which is possible by using the Cholesky decomposition,
- Generate a vector $(Z_1, ..., Z_n)$ of independent standard variables,
- Let $X = \mu + AZ$, which has the desired distribution thanks to the affine transformation property.

In Figure 1.4 we can see 10 samples from the prior and posterior distributions, with the grey area and the black line being respectively the standard deviation and the mean. We see that a Gaussian process is the weighted averages of the observed variables.

1.4 Regression with noisy observations

The assumptions that we made before suggest that we receive observations without noise, that is to say draws from a noiseless distribution. This is observable in Figure 1.4 (b) as the variance in the posterior distribution is null on observation points. For a more realistic modelling, where observations always come with some noise or inaccuracy, is it better to assume that some white noise ϵ with variance σ_n^2 comes polluting the data such

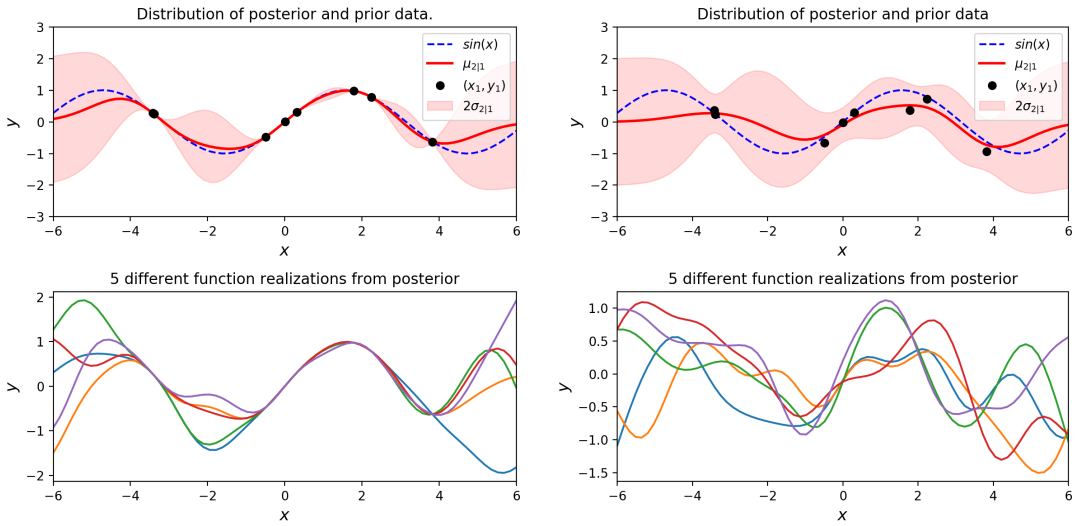


Figure 1.5: Example of noise free (a) and noise (b) regression on sinusiode

as we get $y = f(x) + \epsilon$. The prior on the noisy observation then becomes:

$$cov(y) = K(X, X) + \sigma_n^2 I_n. \tag{1.20}$$

Then, the joint prior that we saw before becomes:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I_n & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \tag{1.21}$$

and conditioning the joint Gaussian prior distribution on the observations:

$$\begin{aligned}(f_* \mid X_*, y, f) \sim N(&K(X_*, X)[K(X, X) + \sigma_n^2 I_n]^{-1} y, \\ &K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I_n]^{-1} K(X, X_*)),\end{aligned} \tag{1.22}$$

In Figure 1.5 we can see the effect of a noise assumption on the data. This is be done by adding it to the covariance kernel when modelling the observation.

1.5 Choice of the hyper parameters

One can to manually tune the hyper-parameters of the kernel function to reach e.g. a desired level of smoothness in the resulting regression. The effect of the length-scale l and variance σ^2 for the squared exponential are shown in Figure 1.6. However, it is possible

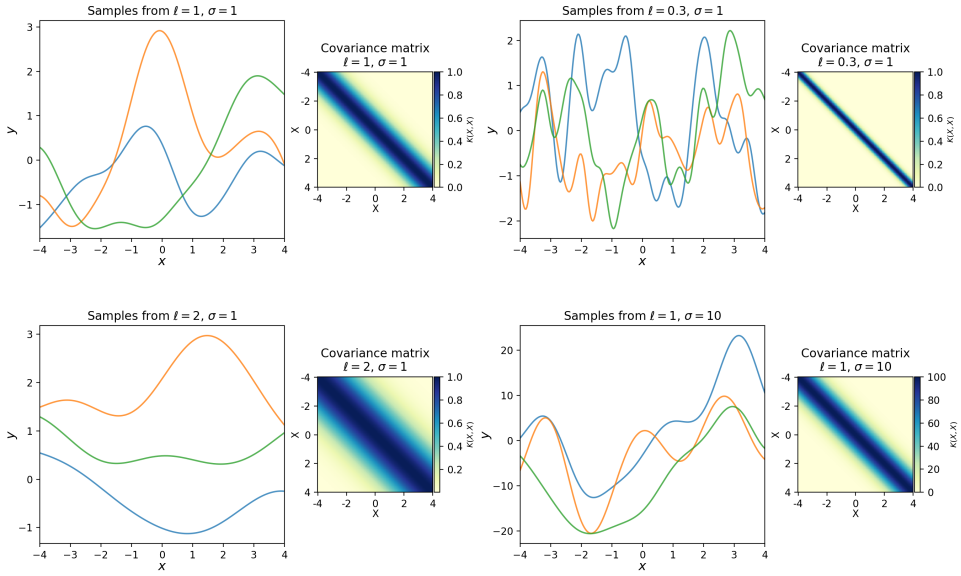


Figure 1.6: Example of different values of hyper-parameters for the quadratic exponential kernel

to obtain optimal parameters by maximizing the marginal likelihood which is defined as:

$$p(\mathbf{y} \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{y} - \mu)^\top \Sigma^{-1}(\mathbf{y} - \mu)\right), \tag{1.23}$$

with d being the dimension of the marginal. We can then transform the equation by observing the log-likelihood:

$$\log p(\mathbf{y} \mid \mu, \Sigma) = -\frac{1}{2}(\mathbf{y} - \mu)^\top \Sigma^{-1}(\mathbf{y} - \mu) - \frac{1}{2}\log|\Sigma| - \frac{d}{2}\log 2\pi \tag{1.24}$$

We see that the first term corresponds to a fit related to the data, whereas the rest corresponding to a complexity penalty. Then, the optimal parameters can be found by minimizing the log marginal likelihood by a gradient descent.

1.6 Incorporating a non zero mean function

When fitting a Gaussian process, one can choose a to use a zero mean function, but that is not necessary. The limitations of a zero mean are not extensive, as the posterior distribution will still take into account the mean of the input data.

If one were to implement a fixed mean function then the predictive mean would become:

$$\overline{f_*} = m(X_*) + K(X_*, X)K_y{-}1(y - m(X)), \tag{1.25}$$

with $K_y = K + \sigma_n^2$ and the predictive variance remaining unchanged from the one previously observed.

However, it is hard and unusual to define a deterministic mean. It is generally more convenient to specify a basis function whose parameters and coefficients β are determined from the input data. Let consider the following regression:

$$g(x) = f(x) + h(x)^\top \beta, \tag{1.26}$$

with $f(x) \sim GP(0, k(x, x'))$ is a zero mean Gaussian process, and $h(x)$ the column vector containing the basis functions. This way of expressing the regression shows that this process is similar to assimilating the data to a linear model with its residuals being captured by the Gaussian process. Then, when fitting the model, one should optimize jointly the parameters of the basis mean and the hyperparameters of the covariance. Additionally, if we assume a Gaussian distribution on the prior such as $\beta \sim N(b, B)$, we can describe the whole regression as the following Gaussian process:

$$g(x) \sim GP\left(h(x)^\top b, k(x, x') + h(x)^\top B h(x')\right) \tag{1.27}$$

We see that the probability distribution of the parameters of the mean affect the covariance. We can then obtain the predictive mean and covariance:

$$\begin{aligned} \overline{X_*} &= H_*^\top \overline{\beta} + K_*^\top K_y{-}1(y + H^\top \overline{\beta}) \\ &= \overline{f_*} + R^\top \overline{\beta} \\ cov(g_*) &= cov(f_*) + R^\top (B^{-1} + H K_y^{-1} H^\top){-}1R, \end{aligned} \tag{1.28}$$

with:

$$\begin{aligned} \overline{\beta} &= (B^{-1} + H K_y^{-1} H^\top)^{-1}(H K_y^{-1} y + B^{-1} b) \\ R &= H_* - H K_y^{-1} K_* \end{aligned} \tag{1.29}$$

We see that the mean is the combination of the linear output, which depends on the data input and the prior, and the Gaussian process model prediction of the residuals. The covariance ends up being the sum of the zero mean process and the contribution of the new introduced parameters.

1.7 Comparison with Kernel Ridge Regression

Both Gaussian Process and Kernel Ridge regression (KRR) learn a function space by using a kernel trick. The difference is that the GPR uses the kernel to define the covariance of a

prior distribution over the target functions and uses the observed training data to define a likelihood function whereas KRR learns a linear function, which is is chosen based on the mean-squared error loss with ridge regularization, in the space induced by the respective kernel which corresponds to a non-linear function in the original space. But both use the Bayes theorem, and define a Gaussian posterior distribution over target functions, whose mean is used for prediction.

An implementation difference is that while performing a regression with a Gaussian process, the the kernel's hyper-parameters are optimized during a gradient-ascent on the marginal likelihood function. For a kernel ridge regression, one needs to perform a grid search on a cross-validated loss function such as mean-squared error loss. An other major difference resides in that a Gaussian process is a probabilistic model and therefore can provide confidence intervals and posterior samples along with the predictions whereas a kernel ridge regression can only provides predictions. It is interesting to note that the mean of the predictive distribution of a GPR is equal to the prediction of a KRR when using the same kernel and hyper-parameters.

2 Financial Greek Example - Gamma

To start with the performance in terms of graph fitting of the GPR, we will first observe the Gamma curve and surface of a stock option priced with the Black-Scholes model. European options are options contracts which can only be executed at their expiration dates. Here we will focus our interest on vanilla European call options, which are financial contracts in which buyer has the right, but not the obligation, to buy a stock, bond, commodity or other asset or instrument at a specified price. Such an option is defined by a strike price K and a time to maturity T. With S_t the underlying asset price at time t, we have the corresponding payoff function:

$$\text{Payoff} = \max(S_T - K, 0) \tag{2.1}$$

Under the Black-Scholes model and for a non-dividend-paying asset, the price of a call option [see 2] is:

$$C_T = N(d_1)S_t - N(d_2)Ke^{-r(T-t)}, \tag{2.2}$$

with N the cumulative distribution function of the standard normal distribution and:

$$\begin{aligned} d_1 &= \frac{\log\left(\frac{S_0}{K}\right) + T\left(r + \frac{\sigma^2}{2}\right)}{\sigma\sqrt{T}} \\ d_2 &= d_1 - \sigma\sqrt{T-t} \end{aligned} \tag{2.3}$$

with S_0 the initial value of the underlying asset,K the strike price, r the risk free rate, σ the implied volatility, T the time to maturity.

For hedging purposes, one can be interested to measure the corresponding Gamma of a portfolio containing this options. Gamma is defined as the second derivative of the value function with respect to the underlying price:

$$\Gamma = \frac{\partial C_T}{\partial S}, \tag{2.4}$$

which under this model satisfies the following equation:

$$\Gamma = \frac{1}{S_0 \sigma \sqrt{T}} \frac{e^{\frac{-d_1^2}{2}}}{\sqrt{2\pi}}. \tag{2.5}$$

We can see that estimating such a function is not trivial and that getting an accurate non-linear regression over the five parameters is definitely very time consuming.

2.1 Curve Fitting

Let consider a long position in a call option. As getting a smooth representation for a range of stock price values would require a too large computational workload, simple alternative can be used such as a grid or a low order polynomial interpolation. This way, the grid can be computed during closed market time and then using some interpolation intraday when needed. To illustrate the performance of the GPR against other classical regressions, we can see in Figure 2.1 the comparison against a grid and a polynomial interpolation of the analytical value of the Gamma. We can see that the GPR can achieve a good accuracy even with a low number of data points. Using a grid with a higher density of points obviously increases the accuracy, but also comes with additional computational cost.

2.2 Surface Fitting

To further illustrate the fitting performance of the GPR, we can observe the fit of the regression on the Gamma surface for an increasing amount of input data points. In Figure 2.2 we can see the plot of the Gamma curve and the corresponding out of sample prediction error for a 50×50 input grid. The prices being shown in dollar units, we see that the error is extremely small for a limited amount of point compared to the surface. We also can notice that the performance of GPRs is reduced at extremities of the data input, therefore, to reach a better accuracy, one can extend the input training grid beyond

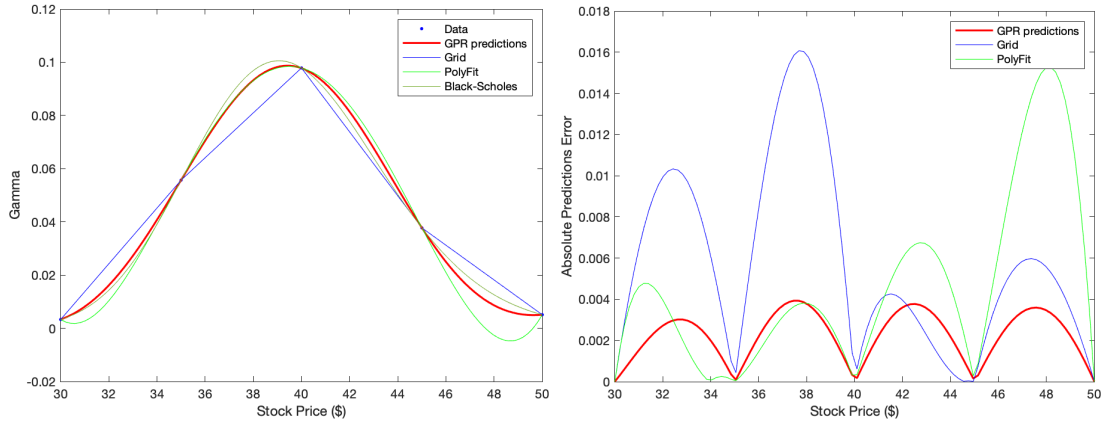


Figure 2.1: Predictions curves (a) and predictions errors (b) for GPR, Grid and polynomial interpolation

Grid Size	MAE	AAE
25×25	3.2e-03	2.8e-06
50×50	2.6e-03	7.7e-07
75×75	2.3e-03	3.4e-07
100×100	2.1e-03	1.9e-07

Table 2.1: Gamma surface fit performance for different grid sizes

the scope of the testing grid. In Table 2.1 we can see the corresponding maximum (MAE) and average (AAE) absolute errors for different sizes of input grids.

3 Summarizing Volatility Surface

The volatility surface is a three dimensional plot of the implied volatility of a financial asset. It is often described as an extended graphical representation of the volatility smile. The interest behind this graph resides in the fact that there exist disparities in the market price of stock options and that market behavior differs depending on the moneyness and

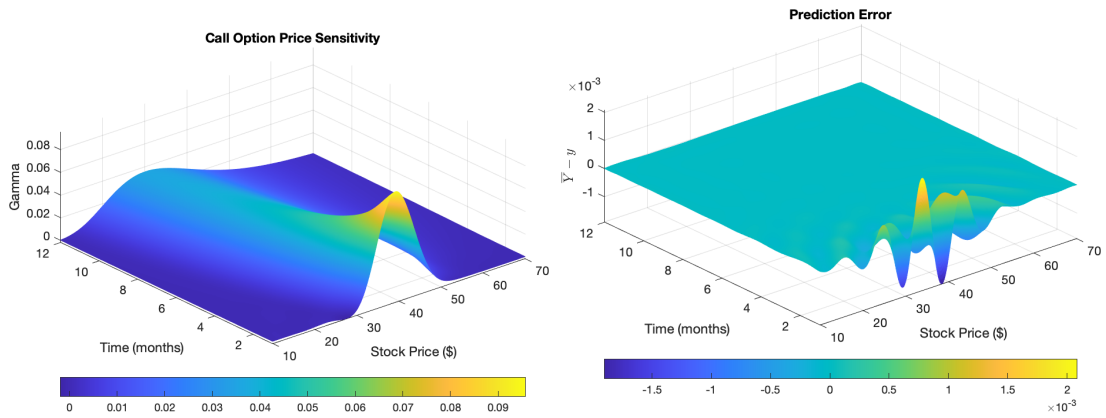


Figure 2.2: Gamma surface fit (a) and predictions errors (b) for GPR

the tenor of the option.

In simpler pricing models, such as the standard Black-Scholes model, the implied volatility surface across strike prices and time to maturity is often presumed constant and would lead to a flat graph. But we see that in practice, this is hardly the case. Indeed, we observe that out-of-the-money strike prices tend to have higher implied volatilities than at-the-money strike prices, giving place to a volatility smirk, hence the need of a graphical representation to illustrate the market tendencies. Additionally, as the time to maturity increases, volatilities across strike prices tend to converge to a relatively constant level. Though, we usually observe a volatility smirk across the range of tenors; options with shorter time to maturity often have a higher volatility than options with longer maturities. This observation is seen to be even more pronounced in periods of high market stress. This being said, surfaces can differ drastically between options and option chains, and therefore the representation of the volatility surface can lead to slightly wavy rather than strictly convex graph.

With a relatively large amount of strike prices but fewer tenors, the volatility surface can then help to price options. With interpolation, we can infer market's implied volatilities for a larger range of strikes and tenors than for the ones actively traded.

In Figure 3.1 is shown a volatility surface fit from freely available market data of a call option chain of S&P 500. Since the points are the last traded value and that the liquidity of the asset is far from other derivatives such as interest rate swaps, optimizing the noise parameter in the input data is crucial. One can also note that the option chain evolves in two time scales, evolving from a weekly basis, to a monthly and then yearly time to maturity. Therefore, the non-uniform sparsity of the data makes manual adjustment needed when optimizing parameters, in order both deal with the 'shaky' or wavy appearance of the input data.

4 Derivative Pricing

On the derivative pricing market, a huge variety of models are used, since specific models can be more explanatory regarding the characteristics and behavior of the underlying asset. Here we will delve into a frequently used model more advanced than the Black-Scholes, the Heston model and two different pricing techniques, the binomial tree and path simulating with Monte Carlo simulations.

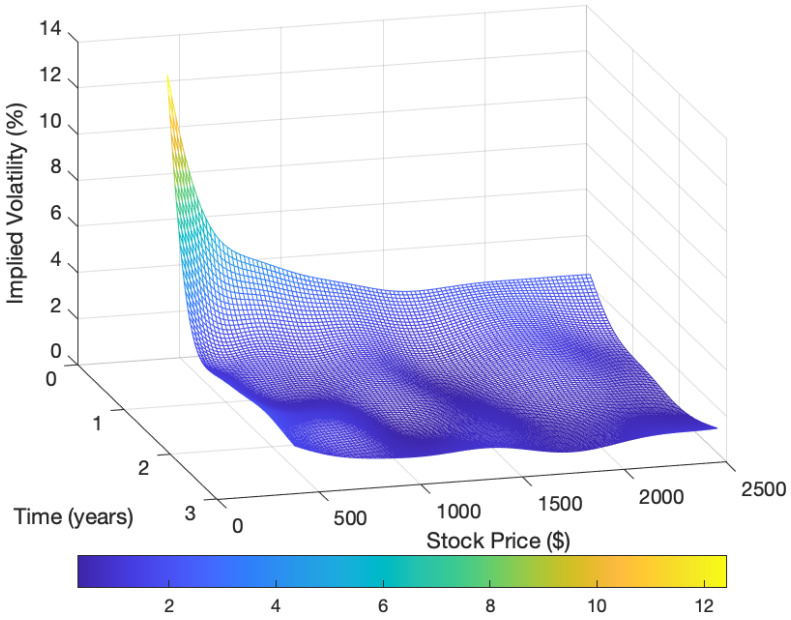


Figure 3.1: Implied volatility surface fit with GPR

4.1 European Options

The price of a European call option, under the risk-neutral valuation approach [7], is expressed as follows:

$$\begin{aligned} C_T(K) &= e^{-rt} \int_0^\infty \max(S_T - K, 0) f(S) dS \\ &= e^{-rt} \int_K^\infty (S_T - K) f(S) dS \\ &= e^{-rt} \int_K^\infty (e^{x_T} - e^k) f(x_T) dx_T, \end{aligned} \tag{4.1}$$

with f the risk neutral density function corresponding to the risk-neutral measure under the fundamental theorem of asset pricing, $x_t = \log(S_t)$, $k = \log(K)$.

4.1.1 Heston Model

The Heston Model [see 9], named after Steve Heston, is a popular option pricing model, which introduces a stochastic volatility following a CIR [see 6] process:

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{V_t} S_t dW_t \\ dV_t &= \kappa(\theta - V_t) dt + \sigma \sqrt{V_t} dZ_t \\ dW_t dZ_t &= \rho dt, \end{aligned} \tag{4.2}$$

where S_t and V_T are price and volatility processes, W_t and Z_t are Wiener processes with correlation ρ, θ the long-run mean of volatility, κ the rate of reversion, r the risk free rate and σ volatility of volatility.

It is similar in several aspects to the more common Black-Scholes model, which in contrast, assumes a constant volatility. There are empirical evidences and mathematical aspects [see 5] that such a model is a more realistic representation of market's behavior. For example, studies have shown that that asset's log-return distribution has heavy tails and high peaks, and therefore more likely leptokurtic. Such issues are addressed by different parameters of the model, such as ρ, which affects the skewness of the distribution, and σ, which affects the kurtosis of the distribution [see 11].

Under the Heston model, a vanilla European option on a non-dividend paying asset has a closed form solution:

$$C(S_t, V_t, t, T) = S_t P_1 - K e^{-r(T-t)} P_2, \tag{4.3}$$

where, with $x = \log(S_t)$:

$$\begin{aligned}
P_j(x, V_t, T, K) &= \frac{1}{2} + \frac{1}{\pi} \int_0^\infty \mathrm{Re}\left(\frac{e^{-i\phi \log(K)} f_j(x, V_t, T, \phi)}{i\phi} \right) d\phi \\
f_j &= \exp\left(C(T-t, \phi) + D(T-t, \phi) V_t + i\phi x \right) \\
C(T-t, \phi) &= r\phi i r + \frac{a}{\sigma^2} \left[(b_j - \rho\sigma\phi i + d)(T-t) - 2\log\left(\frac{1 - g e^{dr}}{1-g} \right) \right] \\
D(T-t, \phi) &= \frac{b_j - \rho\sigma\phi i + d}{\sigma^2} \left(\frac{1 - e^{dr}}{1 - g e^{dr}} \right) \\
g &= \frac{b_j - \rho\sigma\phi i + d}{b_j - \rho\sigma\phi i - d} \\
d &= \sqrt{(\rho\sigma\phi i - b_j)^2 - \sigma^2 (2 u_j \phi i - \phi^2)}
\end{aligned} \tag{4.4}$$

While a numerical method estimating this solution is extremely accurate, it is also very time consuming. Therefore it is mostly used as a baseline when comparing to other computation methods.

4.1.2 Fast Fourier Transform

A commonly used method is the Fast Fourier Transform (FFT) [see 4], which is a discrete approximation of a Fourier transform. With the Fourier transform F and the corresponding

inverse Fourier transform of an integrable function f being:

$$\begin{aligned} F\{f(x)\} &= \int_{-\infty}^{\infty} e^{i\phi x} f(x) dx = F(\phi) \\ F^{-1}\{f(x)\} &= \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\phi x} F(\phi) d\phi = f(x) \end{aligned} \tag{4.5}$$

Then we can show that the corresponding discrete approximation under FFT is:

$$\omega(k) = \sum_{j=1}^{N} e^{-\frac{i2\pi}{N}(j-1)(k-1)} f(x_j) \tag{4.6}$$

The aim of this algorithm is to reduces the number of multiplications in the required N summations. It does so by reducing them from an order of N^2 to that of $Nln_2(N)$. As mention Carr and Madan (1999) to perform such a transformation, the function needs to be square integrable and $C_t(k)$ tends to S_0 as k tends to $-\infty$, thefore one needs to introduce a parameter α, a damping factor, such as:

$$c_T(k) = e^{\alpha k} C_T(k), \tag{4.7}$$

with $\alpha > 0$. Then, it is possible to express the characteristic function of the scaled call by:

$$F_{c_T}(\phi) = \frac{e^{-T} F_{C_T}(\phi - (\alpha + 1)i)}{\alpha^2 + \alpha - \phi^2 + i(2\alpha + 1)\phi}. \tag{4.8}$$

How to then obtain F_{C_T} is detailed by Heston (1999) and is under the form:

$$F_{C_T}(\phi) = e^{C(T-t,\phi) + D(T-t,\phi)\phi + i\phi x_T}. \tag{4.9}$$

An explicit function with practical use for FFT is:

$$F_{C_T}(\phi) = \exp\left(A(\phi) + B(\phi) + C(\phi)\right), \tag{4.10}$$

where:

$$
\begin{aligned}
A(\phi) &= -\frac{\kappa\theta}{\sigma^2}\left[2\log\left(\frac{2\psi(\phi)-(\psi(\phi)-\gamma(\phi))(1-e^{\psi(\phi)T}}{2\psi(\phi)}\right)+(\psi(\phi)+\gamma(\phi))T\right] \\
B(\phi) &\frac{2\zeta(\phi)(1-e^{\psi(\phi)T}V_0}{2\psi(\phi)-(\psi(\phi)-\gamma(\phi)(1-e^{-\psi(\phi)T})} \\
C(\phi) &= i\phi(x_0+rT) \\
\zeta(\phi) &= -\frac{1}{2}(\phi^2+i\phi) \\
\psi(\phi) &= \sqrt{\gamma(\phi)^2-2\sigma^2\zeta(\phi)} \\
\gamma(\phi) &= \kappa-\rho\sigma\phi i
\end{aligned}
\tag{4.11}
$$

This characteristic function can then be evaluated to approximate by the Fourier transform:

$$
C_k(k_u) = \frac{e^{\alpha k_u}}{\pi}\sum_{j=1}^{N} e^{-\frac{i2\pi}{N}(j-1)(u-1)} e^{ibv_j} F_{c_T}(v_j)\frac{\eta}{3}\left[3+(-1)^j-\delta_{j-1}\right], \tag{4.12}
$$

where the following parameters have been chosen to reach a compromise between accuracy and computational time :

$$
\begin{aligned}
v_j &= \eta(j-1) \\
\eta &= \frac{c}{N} \\
c &= 600 \\
N &= 4096 \\
b &= \frac{\pi}{\eta} \\
k_u &= -b+\frac{2b}{N}(u-1), \quad u = 1,2,...,N+1
\end{aligned}
\tag{4.13}
$$

One strength of this numerical method is that it evaluates call prices for a vector of strike prices with many terms in common in the calculations.

4.1.3 GPR Implementation

Due to the curse of dimensionality with high dimensional input parameter space, random combinations are sampled uniformly over the ranges given in Table 4.1. For each combination of parameters, the price of the call option is calculated by the FFT method. The training set is then given as input to the GPR model as $X = [K, T, r, \kappa, \rho, \theta, \eta, \nu_0]$. The test set is constructed by sampling uniformly new inputs over a smaller range, which is specified in Table 4.1.

Variable	Training Set	Test Set
K	$40\% \to 160\%$	$50\% \to 150\%$
T	$11M\% \to 1Y\%$	$11M\% \to 1Y\%$
r	$1.5\% \to 2.5\%$	$1.5\% \to 2.5\%$
κ	$1.4 \to 2.6$	$1.5 \to 2.5$
ρ	$-0.85 \to -0.55$	$-0.8 \to -0.6$
θ	$0.45 \to 0.75$	$0.5 \to 0.7$
η	$0.01 \to 0.1$	$0.02 \to 0.1$
ν_0	$0.01 \to 0.1$	$0.02 \to 0.1$

Table 4.1: Training and test set for the Heston model

Number of points	MAE	AAE
1000	1.859e-2	5.413e-3
5000	1.396e-2	4.574e-3
10000	1.147e-2	4.791e-3
20000	8.372e-2	2.253e-3

Table 4.2: Fit Performance for vanilla European call for different number of points

The empirical results are summarized in Table 4.2. We see that the accuracy increases as the training set becomes larger. But by doing so, this increases the computational load. Nevertheless, there are other factors that can be used to improve the accuracy without increasing the amount of data points. One can indeed decrease the number of parameters by fixing a few ones in order the reduce the dimensionality, or reduce the width of the training set since a higher density gives a better accuracy. An other interesting result from the regression is the speedup performance of the predictions compared to the FFT calculations. For an input of 10000 points we observe a speedup of 16 times.

4.2 American Options

American options, in contrast to European options, can be exercised before they expire. Therefore their pricing becomes an optimal stopping problem, and therefore there is generally no closed form solution for this kind of option.

4.2.1 Binomial Tree

Tree based methods are often used to price these options by mapping price movements of the underlying security. These price movements are represented by a grid of equally spaced time steps, with a series of nodes at each step indicating the price of the security and of the option.

At each node the security movements are translated by a chosen probability of going up or

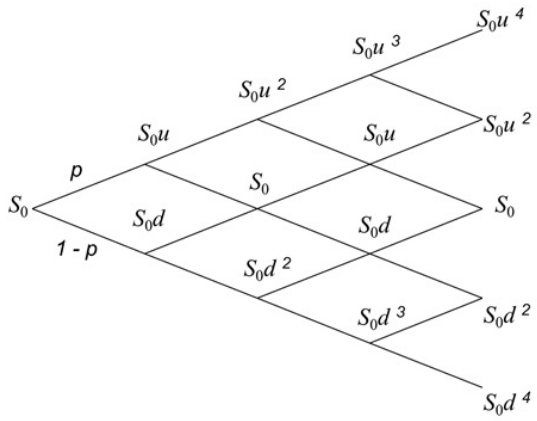


Figure 4.1: Binomial tree

down, then the option price is evaluated, and finally discounted back to obtain the price at the first node. The advantage of tree methods is that not only they can be used to value just about any type of option, but also that they can be easily implemented. We see that through this method the price of a European option converges to the Black-Scholes price. Here, the valuation of American options is done by assessing the profitability of early exercise at every node.

One popular tree method is the binomial model, often called CRR tree [see 8]. In this tree, at each step, the price of the underlying instrument will move up or down by a specific value, respectively u and d. The value of these factors depends on the underlying volatility σ and the time duration of a step $t = T/n$ such as we have:

$$\begin{aligned} u &= \exp\left(\sigma\sqrt{t}\right) \\ d &= \exp\left(\sigma\sqrt{t}\right) = \frac{1}{u} \end{aligned} \tag{4.14}$$

with the corresponding probabilities of ups and downs being: $p = \frac{e^{(r-q)t}-d}{u-d}$ and $1-p$.

We can see here that the CRR tree is recombinant, that is to say that any path having the same amount of ups and downs will end up in the same node. We can see such a tree in Figure 4.1.

In the case of American option, the exercise value is evaluated at each node, which for a put option is:

$$\max\left[K - S, 0\right] \tag{4.15}$$

Then, starting an iterative operation from the final node, the binomial value is calculated such as:

$$C_{t-\Delta t,i} = e^{r\,\Delta t}\left(pC_{t,i} + (1-p)C_{t,i}\right), \tag{4.16}$$

where $C_{t,i}$ is the option value for the ith node at time t. But since early exercise is possible

Variable	Training Set	Test Set
K	$40\% \rightarrow 160\%$	$50\% \rightarrow 150\%$
T	$11M\% \rightarrow 1Y\%$	$11M\% \rightarrow 1Y\%$
r	$1.5\% \rightarrow 2.5\%$	$1.5\% \rightarrow 2.5\%$
σ	$0.005 \rightarrow 0.55$	$0.1 \rightarrow 0.5$

Table 4.3: Training and test set for the Heston model

Number of points	MAE	AAE
1000	1.249e-3	5.838e-4
5000	1.054e-3	4.406e-4
10000	8.206-4	3.467e-4
20000	7.516-4	3.394e-4

Table 4.4: Fit Performance for American put for different number of points

in the case of American options, the value at each nodes becomes:

$$\begin{aligned} C_{t-\Delta t,i} &= \max\left[ExerciseValue, BinomialValue\right] \\ &= \max\left[K - S_{t,j}, e^{r\Delta t}\left(pC_{t,i} + (1-p)C_{t,i}\right]\right) \end{aligned} \tag{4.17}$$

4.2.2 GPR Implementation

The Gaussian process model is fitted on a training set from values obtained from a binomial tree for an American put option with time step of one business day. The input data is fed to the model as $X = [K, T, r, \sigma]$ and is uniformly sampled over the ranges in Table 4.3. The predictive performance is summarized in 4.4. In terms of time performance, the GPR has the advantage of not being impacted by the number of time steps. Therefore, the speedup becomes even greater as the maturity increases. For 10000 data points and one year maturity, the GPR performs more than 60 times faster.

4.3 Barrier Options

Barrier option is a common type of path dependant exotic option. Therefore, unlike a vanilla option, the payoff is depending on the underlying not only at expiry, but also on a set of observable times. Here specifically, a barrier option payoff is determined whether or not the price of the underlying crosses a certain level, or potentially several ones. Most of the time these options are cheaper than vanilla ones since the contract is more probable to become worthless.

Two general types of barrier options exist, which are knock-out option where the payoff is null if the barrier is crossed, and knock-in option where in contrary the contract becomes

valid once the barrier is crossed. Here we will focus on a down-and-in barrier call option, which means the contract, the right to buy, is granted once a lower barrier than the initial underlying price is crossed.

For this type of option, increasing the absolute difference between the initial price and the barrier level is negatively correlated to the option price, since with other parameters staying the same, the probability to hit the barrier decreases. As the difference decreases, the price converges with the vanilla equivalent. The volatility is also an important factor in the pricing, as a higher volatility leads to a higher probability to hit the barrier, and therefore for the knock-out is positively correlated to the option price.

4.3.1 Discretization

The price of the barrier is influenced by the frequency of the monitoring. There is a closed formula for continuous monitoring for several barrier option types. For a down-and-out put option with strike price K, barrier S_b and maturity T, under the Black-Scholes model, we have [see 13]:

$$C_{\text{down-and-in}} = S_0 e^{-qt} \left(\frac{S_b}{S_0}\right)^{2\lambda} N(y) – K e^{-rt} \left(\frac{S_b}{S_0}\right)^{2\lambda-2} N(y - \sigma\sqrt{(t)}) \tag{4.18}$$

with:

$$\begin{aligned} \lambda &= \frac{r - q + \sigma^2/2}{\sigma^2} \\ y &= \frac{\log\left(\frac{S_b^2}{S_0 K}\right)}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T} \end{aligned} \tag{4.19}$$

In reality, however, prices are monitored discretely and there exist a discrepancy between the previous analytical solution and the actual price. This shift can be determined by a set of factor which are the monitoring frequency, the asset volatility and a constant $\beta \approx 0.5826$ [see 3].

This shift can be explained that by decreasing the amount of observe events, the probability of hitting the barrier decreases, and we can expect the price of a down-and-out to also decreases as the number of observation decreases.

4.3.2 Monte Carlo paths simulation

As it is often the case for financial derivatives with path dependent payoffs, a set of simulations solutions are used in order to price them. Monte Carlo is a popular solution among these, as it is relatively easy to implement and gives a lot of flexibility regarding

the simulated data.

The main concept behind Monte Carlo simulations is to estimate some parameter:

$$\theta = E[g(X)], \tag{4.20}$$

where X is a random variable and g an arbitrary function. It consists in generating a sample containing a certain amount of independent draws from a chosen density function. Then, the estimator is given as:

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} g(X_i) \tag{4.21}$$

Then, by the law of large numbers, we have that for when n tends to infinity then the estimator converges in probability to the actual value of the estimated parameter. If we note the sample variance as:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(g(X_i) - \hat{\theta}\right)^2 \tag{4.22}$$

Then the central limit theorem says that as n tends to infinity:

$$\frac{\hat{\theta} - \theta}{\sqrt{s/n}} \sim N(0,1) \tag{4.23}$$

And therefore, by using the corresponding quantile Tables we can appreciate the accuracy:

$$P\left(\hat{\theta} - z_{1-\alpha/2}\frac{s}{\sqrt{n}} < \theta < \hat{\theta} + z_{1-\alpha/2}\frac{s}{\sqrt{n}}\right) \approx 1 - \alpha \tag{4.24}$$

Therefore the precision of the estimator is directly linked to the standard error $\frac{s}{\sqrt{n}}$. In order to increase the speed of convergence of the estimator for a set of draws, variance reduction techniques may be used, such as antithetic variates or control variates. Very often in the case of Monte Carlo simulations, the price of the underlying asset is supposed to behave as a geometric Brownian motion:

$$dS_t = \mu S_t dt + \sigma S_t W_t \tag{4.25}$$

Then, by applying Ito's lemma:

$$d\log S_t = \left(\mu - \frac{\sigma^2}{2}\right)dt + \sigma dW_t \tag{4.26}$$

By integrating we then obtain:

$$S_t = S_0 \exp\left((\mu - \frac{\sigma^2}{2})t + \sigma \int_0^t dW(\tau)\right) \tag{4.27}$$

We can then discretize over the time interval with small time steps, and by using the properties of standard wiener processes we have:

$$S_{t+dt} = S_t \exp\left((\mu - \frac{\sigma^2}{2})dt + \sigma\sqrt{dt}\epsilon\right) \tag{4.28}$$

where $\epsilon \sim N(0,1)$ is a standard normal random variable. This last equation can be used for generating paths to price the barrier option.

The problem becomes harder when we allow the volatility to be stochastic. Different discretization schemes can be used in this regard. One of them is the quadratic exponential method, which was introduced by Anderson in 2006 [see 1] and is very accurate when approximating a stochastic volatility, typically in the case of the Heston model. To represent a non-central Chi squared distribution, it uses a combination of a quadratic function of a standard Gaussian variable for the larger values, and an exponent function for the lower segment of the distribution. The segment of large value can be approximated by the following formula:

$$V(t) = a(b + Z_v)^2, \tag{4.29}$$

where Z_v is a standard normal distributed random variable, a and b being parameters to determine by moment matching, such as:

$$\begin{aligned} a &= \frac{m}{1+b^2} \\ b^2 &= 2\psi^{-1} - 1 + 2\sqrt{2\psi^{-1}}\sqrt{2\psi^{-1}-1} \\ \psi &= \frac{s^2}{m^2} \\ m &= \theta + (V_0 - \theta)^{\kappa dt} \\ s^2 &= \frac{V_0\eta^2 e^{\kappa dt}}{-\kappa}(1 - e^{\kappa dt}) + \frac{\theta\eta^2}{-2\kappa}(1 - e^{\kappa dt})^2 \end{aligned} \tag{4.30}$$

The segment of lower values can be approximated by using the inverse transform sampling method such as :

$$V(t) = L^{-1}(U_v), \tag{4.31}$$

where U_v is a uniform random variable. In our case, the inverse transform is defined by:

$$L^{-1}(U_v) = \begin{cases} \beta^{-1}\log(\frac{1-p}{1-U_v}) & p < U_v \leq 1 \\ 0 & 0 \leq U_v \leq p, \end{cases} \tag{4.32}$$

where:

$$\begin{aligned} p &= \frac{\psi - 1}{\psi + 1} \\ \beta &= \frac{1-p}{m} \end{aligned} \tag{4.33}$$

Applying Ito's formula to the Heston model dynamics, the underlying stock price, for $s < t$, can be expressed as:

$$S_t = S_s \exp\left(\int_s^t (r - \frac{1}{2}V(u)du + \int_s^t \sqrt{V(u)dW})\right) \tag{4.34}$$

This can be further decomposed into:

$$\log(S_t) = log(S_s) + \int_s^t r du - \frac{1}{2}\int_s^t V(u)du + \rho\int_s^t \sqrt{V(u)dW_1} + \sqrt{1-\rho^2}\int_s^t \sqrt{V(u)dW_2} \tag{4.35}$$

By integrating the variance process we obtain:

$$V_t = V_s + \int_s^t \kappa(\theta - V(u))du + \eta\int_s^t \sqrt{V(u)dW_1}, \tag{4.36}$$

which is equivalent to :

$$\int_s^t \sqrt{V(u)dW_1} = \frac{V_t - V_s + \kappa\theta\Delta t - \kappa\int_s^t V(u)du}{\eta}, \tag{4.37}$$

with $\Delta t = t - s$. Additionally, by doing a mid-point discretization on the integral of the variance process on a small increment we have:

$$\int_s^t V(u)du \approx (V_s + V_t)\frac{\Delta t}{2}. \tag{4.38}$$

Variable	Training Set	Test Set
K	$40\% \rightarrow 160\%$	$50\% \rightarrow 150\%$
H	$0.55 \rightarrow 0.99$	$0.6 \rightarrow 0.95$
T	$11M\% \rightarrow 1Y\%$	$11M\% \rightarrow 1Y\%$
r	$1.5\% \rightarrow 2.5\%$	$1.5\% \rightarrow 2.5\%$
κ	$1.4 \rightarrow 2.6$	$1.5 \rightarrow 2.5$
ρ	$-0.85 \rightarrow -0.55$	$-0.8 \rightarrow -0.6$
θ	$0.35 \rightarrow 0.75$	$0.4 \rightarrow 0.7$
η	$0.01 \rightarrow 0.16$	$0.02 \rightarrow 0.16$
ν_0	$0.01 \rightarrow 0.16$	$0.02 \rightarrow 0.16$

Table 4.5: Training and test set for the Monte Carlo simulations

By inserting these results in the log price we get:

$$\begin{aligned}\log(S_t) = &\log(S_s) + r\Delta t - \frac{\kappa\rho}{2\eta}(V_s + V_t)\Delta t - \frac{1}{4}(V_s + V_t)\Delta t \\ &+ \frac{\rho}{\eta}(V_t - V_s + \kappa\theta\Delta t) + \sqrt{1-\rho^2}\sqrt{V_s + V_t}\sqrt{\Delta t/2}Z,\end{aligned} \tag{4.39}$$

where Z is a standard random variable. Then, in order to generate sample path, the process is the following:

1. Given V_t, evaluate m, s^2 and ψ,
2. Make a random uniform draw U_v,
3. If ψ is inferior or equal to an arbitrary value (set to 1/2 by Andersen), get a large segment value for V_t, else a small segment,
4. Compute $\log(S_{t+\Delta t})$ from the last equation

4.3.3 GPR Implementation

The Gaussian process model is fitted on a training set from values obtained from path simulations for an down-and-out where the Heston model is used to describe the market. The Monte Carlo values are obtained from 100 000 simulated paths and are using a time step of one business day. The input data is fed to the model as $X = [K, H, T, r, \kappa, \rho, \theta, \eta, \nu_0]$ and is again uniformly sampled over the ranges in Table 4.5. Since such simulation is very time consuming, the speedup from the GPR is considerable, and over 1000 times faster.

Number of points	MAE	AAE
1000	1.793e-1	7.644e-2
2000	9.473e-2	4.991e-2
3000	6.441e-2	4.357e-2
4000	5.172e-2	3.866e-2
5000	3.919e-2	3.723e-2

Table 4.6: Fit Performance for the Barrier option for different number of points

5 Conclusion

We showed how Gaussian process regression is a logical extension of a Bayesian linear regression by bringing more alternatives and flexibility. We presented several applications under which this type of regression is performing well, such as curve fitting, summarizing surfaces and pricing estimation from different models. An important factor in the procedure is the choice of kernel function and the optimization of its hyperparameters. Since the prediction process depends on the inversion of large covariance matrices, performing algorithm are required to carry out these calculations. As we have seen, due to an exponentially increasing amount of points required in higher dimensions, a clever choice in the input data is necessary. However, since the learning process or fitting is only need to be completed once, the prediction process is much less affected by these limitations, and showed that it even performs faster than several pricing techniques. These methods and techniques can be implemented to a much broader range of applications, which makes Gaussian process regression a promising tool.

References

[1] Andersen, L. (2007). Efficient simulation of the Heston stochastic volatility model. *Journal of Computational Finance*, 11.

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

[3] Broadie, M., Glasserman, P., and Kou, S. (1997). A continuity correction for discrete barrier options. *Mathematical Finance*, 7(4):325–349.

[4] Carr, P., Stanley, M., and Madan, D. (2001). Option valuation using the Fast Fourier Transform. *Journal of Computational Finance*, 2.

[5] Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236.

[6] Cox, J. C., Ingersoll, J. E., and Ross, S. A. (1985). A theory of the term structure of interest rates. *Econometrica*, 53(2):385–407.

[7] Cox, J. C. and Ross, S. (1976). The valuation of options for alternative stochastic processes. *Journal of Financial Economics*, 3(1-2):145–166.

[8] Cox, J. C., Ross, S. A., and Rubinstein, M. (1979). Option pricing: A simplified approach. *Journal of Financial Economics*, pages 229–264.

[9] Heston, S. L. (2015). A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *The Review of Financial Studies*, 6(2):327–343.

[10] Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, 209:415–446.

[11] Mondal, M., Alim, M., Rahman, M., and Biswas, M. H. A. (2017). Mathematical analysis of financial model on market price with stochastic volatility. *Journal of Mathematical Finance*, 07:351–365.

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

[13] Rubinstein, M. and Reiner, E. S. (1991). Breaking down the barriers. *Risk Magazine*, 4(8):28–35.

[14] Spiegeleer, J. D., 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.

[15] Woodbury, M. A. (1950). Inverting modified matrices. Statistical Research Group, Memo. Rep. 42, Princeton University, Princeton, N. J.