Article

Not peer-reviewed version

Performance Enhancing Market Risk Calculation Through Gaussian Process Regression and Multi-Fidelity Model

Noureddine Lehdili [*] , Pascal Oswald , Hoang Dung Nguyen

Posted Date: 14 March 2025

doi: 10.20944/preprints202503.1000.v1

Keywords: value-at-risk; expected shortfall; Basel III; FRTB; option pricing, multi-asset option; Gaussian processes regression; multi-fidelity model; machine learning; neural networks

Article

Performance Enhancing Market Risk Calculation Through Gaussian Process Regression and Multi-Fidelity Modeling

N. Lehdili [1,*], P. Oswald [1] and H. Nguyen [2]

[1] Expert Leader Market & Counterparty Risks Modelling
[2] LPSM/Université Paris Cité. The research of H.D. Nguyen is funded by a CIFRE grant from Natixis
[*] Correspondence: noureddine.lehdili@natixis.com

Abstract: The market risk measurement of a trading portfolio in banks, specifically the practical implementation of the value-at-risk (VaR) and expected shortfall (ES) models, involves intensive recalls of the pricing engine. Machine learning algorithms may offer a solution to this challenge. In this study, we investigate the application of the Gaussian process (GP) regression and multi-fidelity modeling technique as approximation for the pricing engine. More precisely, multi-fidelity modeling combines models of different fidelity levels, defined as the degree of detail and precision offered by a predictive model or simulation, to achieve rapid yet precise prediction. We use the regression models to predict the prices of mono- and multi-asset equity option portfolios. In our numerical experiments, conducted with data limitation, we observe that both the standard GP model and multi-fidelity GP model outperform both the traditional approaches used in banks and the well-known neural network model in term of pricing accuracy as well as risk calculation efficiency.

Keywords: value-at-risk; expected shortfall; Basel III; FRTB; option pricing, multi-asset option; Gaussian processes regression; multi-fidelity model; machine learning; neural networks

MSC: 91G20, 91B05, 62G08, 60G15, 65D05

1. Introduction

Market risk refers to the risk of losses resulting from adverse movements in market prices, encompassing both on- and off-balance sheet positions. From a regulatory standpoint, market risk emanates from positions held in a bank trading book, as well as from commodity and foreign exchange risk positions within the entire balance sheet. Counterparty credit risk, on the other hand, relates to the risk that a counterparty in a transaction may get default before all the cash flows have been settled. Credit valuation adjustment (CVA) risk pertains to the risk of incurring losses due to fluctuations in CVA values triggered by changes in counterparty credit spreads and market risk factors. Financial risk management has always been a central activity in banking industry, but since the 2008 global financial crisis, it has become even more important: there is an increasing focus on how financial risks are being assessed, monitored and managed. In the last three decades, extensive research, spanning both academic and financial industry domains [13,15,16,48], has been dedicated to advancing banking practices and refining risk calculations to meet existing and forthcoming challenges.

The European Banking Authority (EBA) plays a crucial role in ensuring consistent implementation of the new regulations across the European Union, particularly with respect to the Fundamental Review of the Trading Book (FRTB) as part of the Basel III reforms [3], by developing technical standards, guidelines, and reports. Specifically, the Basel Committee on Banking Supervision introduced the FRTB framework to enhance accuracy and consistency in trading book capital requirements. This initiative has compelled banks with trading operations to swiftly implement significant changes. Banks had to reinforce their models, data management, information technology infrastructure, and processes.

Additionally, capital markets businesses had to optimize product structures, adjust hedging strategies, and reprice products. For some banks, unprofitable business lines due to FRTB led to downsizing or exiting their operations. The impact of FRTB on banks is multifaceted, affecting methodologies, data management, processes, and systems. These changes pose significant challenges, particularly in areas like non-modellable risk factors and the development of default risk charge (DRC) methodologies. Simultaneously, the implementation of FRTB necessitates substantial upgrades in banking systems, including computational capacities and alignment of risk factors. The FRTB also emphasizes the importance of market and reference data quality, demanding a coherent framework for sourcing and mapping positions accurately. These interconnected challenges highlight the intricate demands faced by banks in adapting to FRTB regulatory requirements. Banks must increase computational capacities, revalue positions daily, and develop new platforms to compute standardized capital charges: significantly higher calculation capacity will be required compared to that for current value-at-risk (VaR) calculations and banks estimate an increase in number of full revaluation runs by a factor of 5 to 10 [4]. Simultaneously, the banking industry response to these complex demands involves exploring the potential of machine learning, which has gained prominence for big data management, function interpolation and exploration in various sectors. In finance, machine learning applications in, such as pricing, fraud detection, credit scoring, and portfolio management, have become increasingly significant [17,19,36]. This technological advancement offers a potential solution to the intricate challenges posed by the FRTB, offering innovative ways to adapt and streamline banking processes in the face of regulatory requirements.

A recent study by the Financial Stability Board [19] highlights the potential of machine learning in enhancing risk models by detecting complex patterns in large datasets. The neural network family, famous for its universal approximation and its customization flexibility, is applied to various financial tasks ranging from derivative pricing and hedging [10,29] to calibration [26] and partial differential equations solving [28]. Zhang, Su, Song, Qiu, Xiao, and Su [50] show improvements in the volatility forecast using neural network and GELM, where the latter is a combination of the generalized autoregressive conditional heteroskedasticity (GARCH) model and extreme machine learning (ELM) model [27], this then enhances the accuracy and efficiency in value-at-risk calculations. Ruf and Wang [46] make a survey on the application of neural network models in quantitative finance. Bayesian models such as the Gaussian process regression can be explored for the yield curve forecast and derivatives pricing [40], reducing computation time. De Spiegeleer, Madan, Reyners, and Schoutens [14] apply GP models to the fast derivatives pricing and hedging under the assumption that the market is modelled by a limited set of parameters. They provide the numerical test of European call, American put and barrier option in the variance gamma and Heston models. Goudenège, Molent, and Zanette [23], Ludkovski [38] replace the linear regression model in the Longstaff and Schwartz [37] model by a Gaussian process regression to price Bermudan options. In particular, at each backward evaluation time step, they train a GP regression to estimate the time value of the option. Crépey and Dixon [12] focus on the approximation of European call in the Heston model, they furthermore study the multi-output Gaussian processes (GPs) structure to simultaneously estimate various option prices underwritten on the same asset. In Lehdili, Oswald, and Gueneau [34] the market value of the whole trading portfolio of different options on a common underlying is efficiently approximated by a Gaussian process regression. In a nutshell, the choice between neural networks and Gaussian process regression depends on the specific task requirements. The former is The latter is preferred when uncertainty analysis in predictions is crucial, or when there are limitations in the number of training points available. Despite these advancements, machine learning methods have not fully integrated into the financial risk measurement framework yet.

This work aligns with the research flow of Lehdili et al. [34], which produced encouraging results for the VaR and ES calculation of the portfolio consisting of financial instruments written on a single asset. However, one drawback of the approach is the curse of dimensionality. As the input space dimension of the pricing function increases, the number of sampling points required to train the machine learning algorithm, used as a proxy pricing function, increases substantially. This issue becomes practically challenging when evaluating multi-asset options such as best-of, worst-of and basket options. The GP techniques applied to fixed income derivatives like Bermudan swaptions also suffer from the curse of dimensionality. At this stage, the main difficulty lies in the limitation of training data rather than the scalability of Gaussian processes model for learning large datasets (cf. Section 3.2). While the direct use of machine learning model may be efficient enough, some enhancement techniques can be applied to achieve optimal results. For example, when performing the risk calculation of an interest rate derivatives portfolio, Ruiz and Zeron [47] apply dimensional reduction techniques, e.g. principal component analysis (PCA) or learning from diffusive risk factors [1], to reduce the number of market risk factors before interpolating the portfolio value by Chebyshev tensors. However, when products are highly nonlinear and depend on many diffusive risk factors, e.g. multi-asset derivatives, the dimensional reduction techniques proposed in Ruiz and Zeron [47] may be no longer appropriate. This leads us to explore other enhancement techniques such as multi-fidelity modeling. Fidelity in modeling refers the degree of detail and precision offered by a predictive model or simulation. High-fidelity models provide highly accurate outcomes but require substantial computational resources, for instance, the pricing engine in our financial case. Due to its high cost, the use of high-fidelity models is often limited in practice. By contrast, low-fidelity models, while resulting to less accurate outcomes, are cheaply accessible and often offer some useful insights of the learning problem. By combining models of different fidelity levels, multi-fidelity modeling exploits all the information along these models and their correlation structure to provide quick yet accurate predictions. Fernández-Godino, Park, Kim, and Haftka [18] do a survey of the multi-fidelity modeling. This technique can be used along with any machine learning models, such as neural networks [35,39] or Gaussian process regression [6,30,32]. In the research, we investigate multi-fidelity Gaussian process regression (mGPR). While the latter is well-known and widely used in geostatistics and physics under the name cokriging (see lecture notes in https://geostatisticslessons.com and references therein), to the best of our knowledge, we have not seen any of its application in quantitative finance. This motivates us to study this modeling for the financial application, where high-fidelity model is the derivative pricing engine and lower fidelity models can be either its simpler approximations or fast pricing engines of correlated products.

The rest of the chapter is organized as follows. Section 2 introduces the principle of market risk according to the regulation, highlighting the computational challenge in the practical implementation of VaR and ES model. Section 3 reviews the Gaussian process regression and its application in the risk calculation of the trading portfolios. Section 4 describes multi-fidelity modeling and how it works with GP model, coming along with some ideas of their application in quantitative finance. Section 5 sketches the setup and the training specification for our numerical experiments, which are reported in Section 6. Section 7 concludes our findings and provides perspectives for future research.

2. Market Risk Assessment and Computational Challenge

According to Basel Committee on Banking Supervision [4], market risk is defined as *"the risk of losses (in on- and off-balance sheet positions) arising from movements in market prices"*. Market risks include default risk, interest rate risk, credit spread risk, equity risk, foreign exchange (FX) risk commodities risk, and so on. Regulatory separates market risk with other kinds of financial risk, for example, credit risks referring to the risk of loss resulting from counterparty (the party with whom the bank makes engagement in the contract) default or operational risk referring the risk of loss resulting from the failures of internal banking system. Under the FRTB framework, banks can choose either the standardized measurement method (SMM) or the internal model-based approach (IMA) to quantify their market risks. The first approach, which can be easily implemented, is not appealing to banks due to its overestimation of the capital required to cover market risks. The second approach more accurately reflects the risk sensitivities and better postulates the economic risk carried by the banking balance sheet. However, to use their own proposed internal model, banks must gain the regulatory approval through a rigorous backtesting procedure.

Under the internal model approach the calculation of capital charge is based on tail risk measures, namely value-at-risk (VaR) and expected shortfall (ES), of the possible loss of the holding portfolio during a certain trading period. VaR and ES are measured at typically high confidence levels $\alpha = 97.5\%$ (for ES) or 99% (for VaR) in a short liquidity horizon, i.e. $h = 1$ day or 10 days. Assume that the value of a considered trading portfolio V_t at date t is a function of risk factors RF_t, denoted by $p_t(RF_t)$. At date $t + h$, the possible loss of the portfolio (or profit and loss), denoted by $L_t(h)$, is defined by

$$L_t(h) = -(V_{t+h} - V_t) = p_t(RF_t) - p_{t+h}(RF_{t+h}) \approx p_t(RF_t) - p_t(RF_t + \Delta RF_h), \tag{1}$$

where $V_{t+h} = p_{t+h}(RF_{t+h})$ is the value of portfolio at date $t + h$ and the approximation is probably the convention used by the bank, considering time horizon h is small, e.g. one day or ten days. By the approximation in (1), $L_t(h)$ can be interpreted as the potential loss of the portfolio resulted from the shock ΔRF_h of the risk factors RF_t. The value-at-risk at a confidence level α represents the minimum capital required to cover the market risk, while the expected shortfall means the average loss given the latter exceeds the VaR. These two risk measures are defined by

$$\begin{aligned} \mathrm{VaR}(L_t, h, \alpha) &= \min\{q \in \mathbb{R} : \mathrm{P}[L_t(h) \le q] = \alpha\}, \\ \mathrm{ES}(L_t, h, \alpha) &= \mathrm{E}[L_t(h) | L_t(h) \ge \mathrm{VaR}(L_t, h, \alpha)] = \frac{1}{1-\alpha} \int_\alpha^1 \mathrm{VaR}(L_t, h, \gamma) d\gamma. \end{aligned} \tag{2}$$

Statistical methods for calculating (2) include analytical, historical and Monte Carlo approaches ([45] Section 2.2 page 61). According to the the Monte Carlo method, $\mathrm{VaR}(L_t, h, \alpha)$ is the α-percentile of the set of possible losses simulated for time horizon h, while $\mathrm{ES}(L_t, h, \alpha)$ is the empirical average of losses above the $\mathrm{VaR}(L_t, h, \alpha)$. Shocks of risk factors at date $t + h$ are generated by synthetic dynamic models calibrated from the historical market data. The market risk calculation procedure by Monte Carlo is summarized in Algorithm 1, see also Hong, Hu, and Liu [25] for a review of risk calculation by the Monte Carlo methods.

Algorithm 1. Calculation of VaR and ES by full repricing approach

```
input :Calibrated diffusion model, pricing engine $p_t$, an actual vector of risk factors $RF_t$, a confidence
       level $\alpha$, time step $h$ and a large number $N$
output:estimated $\mathrm{VaR}(L_t, h, \alpha)$ and $\mathrm{ES}(L_t, h, \alpha)$
1 Compute $p_t(RF_t)$
2 Simulate $N$ scenarios of shock $\Delta RF_h$ using calibrated diffusion model
3 Compute $N$ corresponding prices $p_t(RF_t + \Delta RF_h)$
4 Compute $N$ scenarios of losses $L_t(h) = -(p_t(RF_t + \Delta RF_h) - p_t(RF_t))$
5 Compute VaR and ES by Monte Carlo
```

Remark 1. *In the FRTB, the calculation of market risk for a trading book by the internal model-based approach needs to be implemented for various liquidity horizons associated with different risk factors. To be more specific, for a considered portfolio, banks need to classify involved risk factors into five categories of liquidity horizons: 10, 20, 40, 60 and 120 days. The FRTB requires banks to assess market risk for these liquidity horizons by taking into account shocks to risk factors at the same and longer horizons. For example, the risk calculation for 10-day horizon must consider shocks to risk factors from all five categories, while the risk calculation for 20-day horizon needs to consider shocks at 20 days and longer. This framework provides a more comprehensive and accurate assessment of market risk, but requires a complex and time-consuming calculation process.*

2.1. Computational Challenge and Applications of Machine Learning

The market risk calculation appears to be doable by Algorithm 1. Nevertheless, we have not discussed yet about the computational complexity of the pricing engine. In banks, the pricing engine of a trading derivative portfolio involves various numerical algorithms depending on the nature of the portfolio. Some products or portfolios can be rapidly priced, such as using analytical formulas or accurately fast approximations. However, when financial models and/or products of the portfolio are complex as usually the case in practice, some heavily computational algorithms need to be referenced including Monte Carlo schemes, tree methods, see for example (Crépey [11] Chapter 6 and 7). The risk calculation using any of the algorithms mentioned above is referred to as the full repricing approach (i.e. full revaluation approach). The latter is the most accurate but very costly (for a portfolio of complex products) as the measurement of the tail risk, i.e. VaR and ES, of the portfolio needs to recall intensively the pricing engine.

In the context of implementing the VaR and ES calculation using Monte Carlo simulation, banking institutions tend to avoid direct use of pricers, as it is time consuming and makes them inefficient. Certain banks prefer to use proxies based on sensitivity calculation and second-order Taylor developments to compute the possible losses of derivative portfolios [8], see Section B. Although these approaches offer performance gains, their precision is often questioned. Recently, alternative approaches have emerged that aim to strike a balance between performance and precision in risk calculation, such as value-at-risk and expected shortfall. Among these methods, the works of Crépey and Dixon [12], and De Spiegeleer et al. [14] explore the application of Gaussian process regression in finance, and Ruf and Wang [46] discuss the use of neural networks in the same context. The common idea behind these studies is to use machine learning models as interpolation tools, also known as surrogate models, to estimate derivative prices. More recently, Ruiz and Zeron [47] use Chebyshev polynomials as interpolators, combined with principal component analysis techniques (PCA), for financial risk calculation, aligning with the same line of though. Figure 1 illustrates the risk calculation process that employs these innovative machine learning approaches, involving only a few complete revaluations of a derivative portfolio.

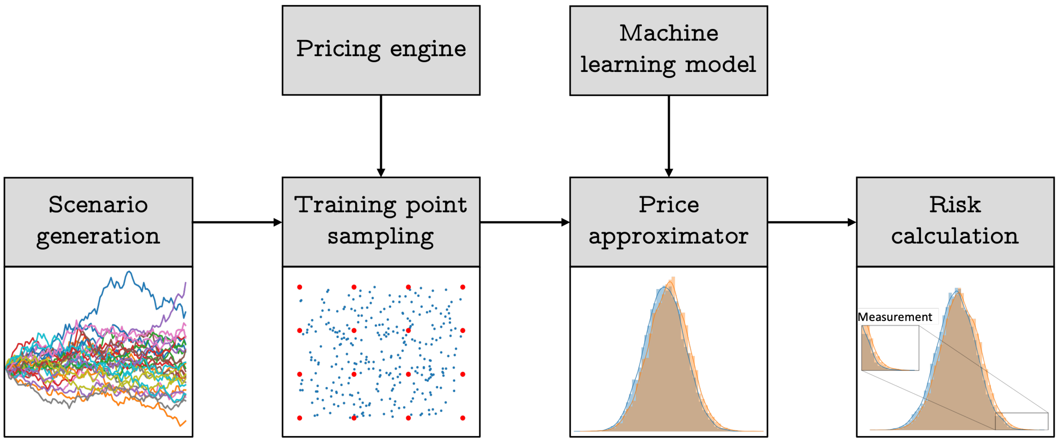

Figure 1. Risk calculation procedure.

2.2. Equity Options

In the remainder of this chapter, the objective is to evaluate and manage the market risk associated with a trading portfolio consisting of equity derivatives. These products are written on a stock whose the price at date t is denoted by S_t. Among these derivative products, vanilla options, barrier options, and American options on a single underlying asset can be mentioned. In the Black-Scholes model, the values of these options are simply determined by analytical formulas depending on the underlying asset price S_t, except the American option values for which a numerical method, e.g. binomial tree, is mandatory. The portfolio may also include multi-underlying options such as basket options, best-of options and worst-off options. For this case, we denote by $S_t = [S_t^1, \dots S_t^d]$ the prices of d underlying assets that are governed by lognormal processes:

$$\begin{aligned} dS_t^j &= rS_t^j dt + \sigma^j S_t^j dW_t^j, \quad \text{for } j = 1 \dots d, \\ d\langle W^j, W^{j'} \rangle_t &= \rho_{jj'} dt, \end{aligned} \tag{3}$$

where r is a risk-free rate, σ^j denote the asset volatilities, and W^j are Brownian motions with a correlation matrix $\mathbf{R} = (\rho_{jj'})_{j,j' \in \{1,\ldots,d\}}$. In what follows, we assume the constant risk-free rate and volatilities.

Let $\eta(S_T)$ denote the value of the European multi-asset derivative at the maturity T, also called the payoff. Subject to the above assumptions, the value of the European multi-asset option V_t at date t is the conditional expectation under the risk neutral probability of the discounted payoff, denoted by E^*,

$$p_t(S_t) = \mathrm{E}^*[e^{-r(T-t)}\eta(S_T)|S_t]. \tag{4}$$

We are particularly interested in four options whose the payoff can be written as a vanilla option payoff against a strike K given in Table 1.

Table 1. Multi-asset options covered in our numerics.

Option	Geometric average call/put	Basket call/put	Best-of call/put	Worst-of call/put
Payoff	$\left(\left(\prod_{j=1}^{d} S_T^j\right)^{\frac{1}{d}} - K\right)^{\pm}$	$\left(\sum_{j=1}^{d} \alpha_j S_T^j - K\right)^{\pm}$ [a]	$\left(\max_j S_T^j - K\right)^{\pm}$	$\left(\min_j S_T^j - K\right)^{\pm}$

[a] with $\sum_{i=1}^{d} \alpha_j = 1$. The case $\alpha_j = \frac{1}{d}$, for $j = 1, \ldots, d$, defines arithmetic average options.

Given (3), the price of geometric average options are fast referenced by the Black-Scholes formula, and the price of basket options can be accurately approximated by the Black-Scholes formula thanks to the convention that the sum of lognormal variables is approximately a lognormal variable [7]. Otherwise, the price of other options in Table 1 are evaluated by Monte Carlo simulation.

3. Gaussian Process Regression for Option Pricing

Gaussian process regression is the canonical method for Bayesian modeling of spatial functions. The method is highly recommended in cases where data is sparse because of its generalization of the Gaussian distribution from finite-dimensional vector spaces to infinite-dimensional functional spaces. Due to this principle, the model is considered as non-parametric. In this section we briefly introduce the Gaussian process regression and its application to model financial derivative products. The more background of GP model can be referenced in Rasmussen and Williams [44] and Murphy [41, Chapter 15].

In what follows, bold lowercase letters, e.g. $\mathbf{x}$, stand for vector notation, whereas bold uppercase letters, e.g. $\mathbf{X}$, are used for matrix notation.

3.1. Gaussian Processes Regression and Prediction

Let $(\Omega, \mathcal{A}, \mathrm{P})$ be a probability space, consider a random vector $\mathbf{x} : \Omega \to \mathcal{X} \subset \mathbb{R}^d$ for a positive integer d corresponding to the dimension of inputs, and a response variable $y \in \mathbb{R}$. In supervised learning, the goal is to learn a target function $f : \mathcal{X} \to \mathbb{R}$ portraying the relation between inputs and the response variable. To this end, we assume the following regression

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \tag{5}$$

where ϵ_i are i.i.d Gaussian white noise with zero mean and variance σ^2. Different from other regression models which usually assume a parameterization form of f, the Gaussian processes place directly a multivariate normal prior on the space of functions. Implicitly, for any set of input points $[\mathbf{x}_1, \ldots, \mathbf{x}_N]$ in $\mathcal{X}$, the random vector $[f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\top$ is multivariate normally distributed. More precisely, given a data set of N observations $\mathcal{D} = (\mathbf{X}, \mathbf{y}) = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, N\}$, where $\mathbf{x}_i$ is taken from set $\mathcal{X}$,

$\mathbf{X}$ is the concatenated matrix in which each row is the row-vector of one observation of inputs. Let $\mathbf{f} = [f(\mathbf{x}_1)\dots f(\mathbf{x}_N)]^\top$. In the Gaussian process regression,

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{X}, K_{\mathbf{X},\mathbf{X}}) \quad \text{implies} \quad \mathbf{y} \mid \mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{X}, K_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}_N), \tag{6}$$

where $\mathbf{I}_N$ is the identity matrix of size N, $\boldsymbol{\mu}_\mathbf{X} = [\mu(\mathbf{x}_1), \dots, \mu(\mathbf{x}_N)]^\top$ is a prior mean vector parameteried by a mean function $\mu\mathcal{X} \mapsto \mathbb{R}$, and $K_{\mathbf{X},\mathbf{X}}$ denotes the covariance matrix characterized by a kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ such that $K_{\mathbf{X},\mathbf{X}} = [k(\mathbf{x}_i, \mathbf{x}_{i'})]_{i,i'=1\dots N}$.

Unless some extra knowledge about the prior mean function, for simplicity one can choose $\mu(\cdot) = 0$. Note that the convention is related only to the prior distribution and does not imply that the posterior distribution (the prediction) has zero mean. Similarly, we can impose some knowledge of the target function on the choice of the kernel function. For example, the exponential sine squared kernel takes into account the periodic characteristic of function ([44] pp. 92). The Matern kernel and the squared exponential kernel are the most frequently used stationary kernels ([44] pp. 83-84). The Matern kernel reads

$$k(\mathbf{x}_i, \mathbf{x}_{i'}) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{l} \|\mathbf{x}_i - \mathbf{x}_{i'}\| \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{l} \|\mathbf{x}_i - \mathbf{x}_{i'}\| \right), \tag{7}$$

where Γ is the gamma function and K_ν is the modified Bessel function of the second kind. The smoothness of the approximate function is controllable through a positive parameter ν, i.e. (7) is once differentiable when $\nu = 3/2$ and twice differentiable when $\nu = 5/2$. As ν tends to infinity, the Matern kernel (7) becomes infinitely differentiable and converges to the squared exponential kernel. The length-scale parameter $l > 0$ can be a d positive vector and each component will be used to standardize a corresponding component of inputs. If the variances by column of the input $\mathbf{X}$ are equal, the vector length-scale parameter will measure the impact of input variables into the predictive conditional variance, yielding an insight about the variable importance ([44] Section 5.1 and 6.6). Other kernel functions can be found in (Rasmussen and Williams [44] Section 4). When y_i in $\mathcal{D}$ is observed directly from the groundtruth function f, i.e. $y_i = f(\mathbf{x}_i)$, the white noise ϵ_i in (5) and its variance σ in above equations will be removed and this refers to the interpolation with noise-free observations. In brief, Gaussian processes provide a lot of flexibility to integrate extra knowledge of the learning problem, apart from data set $\mathcal{D}$, into the prior model hypothesis $p(f)$, which is particularly useful for financial applications.

For a matrix of N' test points, denoted by $\mathbf{X}^* = [\mathbf{x}_1^*, \dots, \mathbf{x}_{N'}^*]^T$, the joint prior distribution of the response reads

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{matrix} \boldsymbol{\mu}_\mathbf{X} \\ \boldsymbol{\mu}_{\mathbf{X}^*} \end{matrix}, \begin{bmatrix} K_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}_n & K_{\mathbf{X},\mathbf{X}^*} \\ K_{\mathbf{X}^*,\mathbf{X}} & K_{\mathbf{X}^*,\mathbf{X}^*} \end{bmatrix} \right),$$

where $\mathbf{f}^* = [f(\mathbf{x}_1^*), \dots, f(\mathbf{x}_{N'}^*)]^\top$, $\boldsymbol{\mu}_{\mathbf{X}^*} = [\mu(\mathbf{x}_1^*), \dots, \mu(\mathbf{x}_{N'}^*)]^\top$; and $K_{\mathbf{X}^*,\mathbf{X}^*} = [k(\mathbf{x}_i^*, \mathbf{x}_{i'}^*)]_{i,i'=1\dots N'}$ and $K_{\mathbf{X},\mathbf{X}^*} = K_{\mathbf{X}^*,\mathbf{X}}^\top$ are respectively covariance matrix of $\mathbf{X}^*$, and of $\mathbf{X}$ and $\mathbf{X}^*$. Once the model is learned (see Section 3.2), the predictive (posterior) distribution of $\mathbf{f}^*$ is

$$\mathbf{f}^* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}^* \sim \mathcal{N}(\mathrm{E}[\mathbf{f}^* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}^*], \mathrm{Var}[\mathbf{f}^* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}^*]), \tag{8}$$

where the posterior mean and variance are given as follows

$$\begin{aligned} \bar{\mathbf{f}}^* &:= \mathrm{E}[\mathbf{f}^* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}^*] = \boldsymbol{\mu}_{\mathbf{X}^*} + K_{\mathbf{X}^*,\mathbf{X}} \left[K_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}_n \right]^{-1} \mathbf{y}, \\ \mathrm{Var}[\mathbf{f}^* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}^*] &= K_{\mathbf{X}^*,\mathbf{X}^*} - K_{\mathbf{X}^*,\mathbf{X}} \left[K_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}_n \right]^{-1} K_{\mathbf{X},\mathbf{X}^*}. \end{aligned} \tag{9}$$

Remark 2. *(i) As we will discuss in Section 3.2, the repeated computation of the inverse matrix* $\left[K_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}_n \right]^{-1}$ *in the learning procedure is a bottleneck of the GP model. However, in the prediction phase (cf. (9)) the matrix* *inversion is required only one time and it is even already computed during the learning phase. In particular, the posterior mean is numerically computed using the following linear expression, for $i' = 1 \dots N'$,*

$$\bar{\mathbf{f}}^*_{i'} = \mu(\mathbf{x}^*_{i'}) + \sum_{i=1}^{N} \omega_i k(\mathbf{x}_i, \mathbf{x}^*_{i'}), \tag{10}$$

where $\boldsymbol{\omega} = [\omega_1, \dots, \omega_N]^\top = \left[K_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}_n\right]^{-1} \mathbf{y}$.
***(ii)** The posterior mean can be also written by the following linear combination, without the loss of generality, suppose the zero mean prior,*

$$\bar{\mathbf{f}}^*_{i'} = \sum_{i=1}^{N} \gamma_i y_i, \tag{11}$$

where $\boldsymbol{\gamma} = [\gamma_1, \dots, \gamma_N]^\top = K_{\mathbf{X}^*,\mathbf{X}}\left[K_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}_n\right]^{-1}$. (11) interprets the GP prediction of function f at a new point, say $\mathbf{x}^*_{i'}$, as a linear combination from a set of N observable points $(y_i)_{i=1\dots N}$. However, the GPs model is not a linear interpolation since the non-linear term is incorporated in γ_i. Moreover, in the general case where exact observations are not accessible, the model will interpolate from the noisy observations taking into account the variance of noise. Hence GP model is considered as a powerful non-linear interpolation tool.

3.2. Estimation of Model Parameters

The learning procedure of Gaussian process model involves finding the best suited parameters of the mean function $\boldsymbol{\mu}_{\mathbf{X}}$, the covariance kernel function k, jointly denoted by $\boldsymbol{\theta}$ and the white noise variance σ. The set of GP parameters can be estimated either by maximum likelihood or maximum a posteriori, or cross-validation method (see (Rasmussen and Williams [44] Section 5.4) and ([41] Section 15.2.4)). By mean of computational facility, maximum likelihood is the the most used among these and this method is chosen to be a default method for the standard Gaussian processes. The criterion Ł that we aim to maximize is the marginal likelihood, also called by the (model) evidence reads

$$\begin{aligned} Ł(\mu, \sigma, \boldsymbol{\theta}) = \log \mathrm{P}(\mathbf{y}|\mathbf{X}) = -\Big[&(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{X}})^\top \Big(K_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}_N\Big)^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{X}}) \\ &+ \log\det\Big(K_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}_N\Big)\Big] + \mathit{constant}. \end{aligned}$$

Notice that the first term of (12) correspond to the data-fit can be seen as the the negative squared loss with a kernel-based weight in linear regression whereas the second term relates to the penalty of the model complexity (see Rasmussen and Williams [44] Section 2.4 and 5.4.1).

Maximizing the evidence (or minimizing the negative evidence) is usually solved by a global optimization method, namely BFGS ([42] Chapter 6), as in Scikit-learn Python package or by any gradient-based method such as Adam [31] in Gpytorch Python package. In any case, the numerical implementation must be done by full batch and involves reversing the covariance matrix of size $N \times N$ whose the complexity is $\mathcal{O}(N^3)$ (using Cholesky decomposition). Contrarily, the prediction step (cf. (9)) costs only $\mathcal{O}(N^2)$. Hence, even Gaussian process model is a great interpolation tool, the classical model is not appropriate for high-dimensional problem that requires an important amount of training points N to explore the target surface. Many innovative approaches are proposed to tackle this issue and in general they are developed through the use of low-rank approximation (or possibly decomposition) of the covariance matrix [22,49] and/or the use of other learning methods such as (stochastic) variational inference [5,24,49]. We refer the reader to the documentation of Gpytorch package on Python [21] for the detail as well as numerical implementation of these approaches.

3.3. Application to Derivative Portfolio Valuation

The application of Gaussian process regression is not restricted to evaluating mono-asset derivatives as illustrated below, but also applies to multi-asset derivatives in the same manner. However, when the number of underlying assets is high, one needs to use Monte Carlo sampling instead of the grid point sampling technique used in low-dimensional problems. The regression model can directly approximate the value of the entire portfolio without any additional computational cost for learning and prediction, regardless of the portfolio size and complexity. This is hence favorable for risk calculation applications ([9] Remark 3).

When the considered portfolio V_t consists of a possibly large number of mono-asset derivatives, one can still use efficiently GP regression by the decomposition technique provided in Lehdili et al. [34]. More precisely, the portfolio may depend on a vector of d assets (risk factors) $S = [S^1, \dots, S^d]$ but each of its derivatives is underwritten on single stock. The portfolio price V_t is the conditional expectation of the total derivative payoffs given the stock asset vector S_t, hence, a function of S_t. By the nature of the considered portfolio, we are able to categorize V_t by risk factor

$$V_t = \sum_{j=1}^{d} V_t^j(S_t^j), \tag{12}$$

where V_t^j is the sub-portfolio corresponding to the j-th stock. We build the approximate of each V_t^j by one-dimensional Gaussian process model. To this end, for $j = 1, \dots, d$, we sample N equidistant training points of S_t^j, say $\mathbf{s}^j = [s_1^j, \dots, s_N^j]$, in a determined range $[\underline{s}^j, \bar{s}^j]$, and the corresponding price $\mathbf{v}^j = [V_t^j(s_1^j), \dots, V_t^j(s_N^j)]$. Since this is the one-dimensional interpolation case and the price functions in finance are usually well-behaved, we only need to recall the pricing engine few times for sampling training data. For example, in the mono-asset portfolio experiments [34], the GPs can reach the good results using only $N = 10$. Once the learning is done, the GP price of a sub-portfolio at a new point s^* is the predictive mean of the posterior distribution

$$\mathrm{E}[V_t^j(s^*) \mid \mathbf{s}^j, \mathbf{v}^j, s^*] = \sum_{i=1}^{N} \gamma_j V_t^j(s_i^j), \tag{13}$$

where γ_j is determined in Remark 2 **(ii)** (with the adaptation of notations). The GP price of the whole portfolio is then deduced by (12). The market risk calculation is straightforward by using the GP pricing model instead of the pricing engine in the Monte Carlo method (cf. Algorithm 1).

Remark 3. ***(i)*** *For a derivative which depends on several market risk factors such as asset volatility, dividend and interest rate and so on, the bank sometimes uses the Black model for the pricing engine. In this case, we can train bring it back to the two-dimensional interpolation problem, i.e. the discounted future price and the volatility, and efficiently apply GP.*
(ii) *In the case of basket options, one can use the current price of $\sum_{j=1}^{d} \alpha_j S_t^j$ (cf. Table 1) as the unique learning feature, leading to one-dimension Gaussian process regression.*
(ii) *When the derivative depends on many risk factors, i.e. multi-asset products. The algorithm may require a large number of training data for a good approximation. In our financial application, the time to get pricing training points (sampled by costly pricing engine) is more relevant than the model learning difficulty of GPR mentioned in Section 3.2.*

4. Multi-fidelity Gaussian Processes Regression

In the previous section, we describe how to apply GPR to interpolate the surface of price function using a few number of price points. However, the number of required points increases dramatically with the number of risk factors. Suppose that we have access to other kinds of information, e.g. a weak pricing engine based on a lower number of Monte Carlo simulations or using few discretization time steps in the tree model, the question arises how to take into benefit of this knowledge in modeling. For this purpose, we investigate multi-fidelity Gaussian process regression (mGPR) that allows us to learn the ground truth target from several information sources, each with a different level of fidelity. Any detail of this model can be found in Brevault, Balesdent, and Hebbal [6], Kennedy and O'Hagan [30], Le Gratiet [32].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 March 2025 doi:10.20944/preprints202503.1000.v1

Assuming that we have s data sets, denoted by $\mathcal{D}_1 = (\mathbf{X}_1, \mathbf{y}_1), \ldots, \mathcal{D}_s = (\mathbf{X}_s, \mathbf{y}_s)$, sorted by increasing level of fidelity (i.e. $\mathcal{D}_s$ is the most accurate and expensive data whereas $\mathcal{D}_1$ is the most simplified and abundant data). In Kennedy and O'Hagan [30], the autoregressive model is used to model the cross-correlation structure between different levels:

$$\begin{aligned} f_j(\mathbf{x}) &= \rho_{j-1} f_{j-1}(\mathbf{x}) + \xi_j(\mathbf{x}), \quad j = 2 \ldots s, \\ [\mathbf{y}_1]_i &= f_1(\mathbf{x}_i) + \epsilon, \quad \mathbf{x}_i \in \mathbf{X}_1, \end{aligned} \tag{14}$$

where ρ_{j-1} are scalar correlation parameters between $\mathbf{y}_{j-1}$ and $\mathbf{y}_j$, $\xi_j(\mathbf{x})$ measures the mismatch between $\mathbf{f}_j$ and $\rho_{j-1} f_{j-1}(\mathbf{x})$ and ϵ is the while noise of the lowest fidelity model. In mGPR, ξ_j is supposed independent of $f_{j-1}, \ldots, f_1$. The goal of the model is to lean f_s which is the relationship between the input and the highest fidelity response.

4.1. Two-fidelity Gaussian Process Regression Model

In this section, we focus on the mGPR with two fidelity levels. Let assume a data set of N_l low-fidelity points, denoted by $\mathcal{D}_l = (\mathbf{X}_l, \mathbf{y}_l) = (\mathbf{x}_i, y_{l_i})_{i=1\ldots N_l}$ and a second set of $N_h (\ll N_l)$ high-fidelity points, denoted by $\mathcal{D}_h = (\mathbf{X}_h, \mathbf{y}_h) = (\mathbf{x}_i, y_{h_i})_{i=N_l-N_h+1\ldots N_l}$, such that N_h last points of low-fidelity data coincides with N_h high-fidelity data, i.e.

$$\mathbf{X}_l = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_{N_l-N_h+1}^\top \\ \vdots \\ \mathbf{x}_{N_l}^\top \end{pmatrix} \quad \text{and} \quad \mathbf{X}_h = \begin{pmatrix} \mathbf{x}_{N_l-N_h+1}^\top \\ \vdots \\ \mathbf{x}_{N_l}^\top \end{pmatrix}. \tag{15}$$

The multi-fidelity modeling involves the following regression functions

$$\begin{cases} f_h(\mathbf{x}) = \rho f_l(\mathbf{x}) + \xi(\mathbf{x}), \quad \mathbf{x} \in \mathcal{X}, \\ f_l(\mathbf{x}) \perp \xi(\mathbf{x}) \mid \mathbf{x}, \end{cases} \tag{16}$$

where the parameter ρ describes the correlation of the high- and low-fidelity data, f_l is a Gaussian process model of $\mathbf{y}_l$ against $\mathbf{X}_l$ and δ is the second GP model conditionally independent with the former. In what follows, we consider GP models with constant prior mean [20,30] parameterized by coefficient β, Marten kernel covariance function $k_{\boldsymbol{\theta}}$ with $\nu = 2.5$ and parameterized by $\boldsymbol{\theta}$ as per (7), and a multiplicative form of the marginal variance σ^2. In particular, the prior model is

$$\mathbf{f}_l \sim \mathcal{N}(\beta_l, \sigma_l^2 K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)), \tag{17}$$

and

$$\boldsymbol{\xi} \sim \mathcal{N}(\beta_h, \sigma_h^2 K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)), \tag{18}$$

where $\mathbf{f}_l$ and $\boldsymbol{\xi}$ are mean vectors $[f_l(\mathbf{x}_1) \ldots f_l(\mathbf{x}_{N_l})]^\top$ and $[\xi(\mathbf{x}_{N_l-N_h+1}) \ldots \xi(\mathbf{x}_{N_l})]^\top$, and the covariance matrices $K_{\boldsymbol{\theta}_{l(h)}}(\mathbf{X}_{l(h)}, \mathbf{X}_{l(h)}) = [k_{\boldsymbol{\theta}_{l(h)}}(\mathbf{x}_i, \mathbf{x}_{i'})]_{\mathbf{x}_i, \mathbf{x}_{i'} \in \mathbf{X}_{l(h)}}$. Hence the multi-fidelity model is characterized by the parameters of the first and second Gaussian processes: $(\beta_l, \sigma_l, \boldsymbol{\theta}_l)$ and $(\beta_h, \sigma_h, \boldsymbol{\theta}_h)$, and the correlation parameter ρ.

4.2. Illustrative Application of Multi-Fidelity Model in Option Pricing

For the pricing and risk calculation problem, our proposed approach is to learn the pricing engine, cf. the high fidelity model, with the less precise price values, cf. lower fidelity data. For instance, if the pricing engine involves Monte Carlo simulation of 100,000 paths, we can sample weak price values calculated by only, e.g. 100 paths as low-fidelity data (see Section 6.2 for numerical experiments). In the American option pricing case, where the accurate price value is calculated using a binomial tree with 100 time steps, the weaker price can be generated by a sparser tree with only, e.g. 10 time steps. Conversely, if the target is the American option, which is expensively computed by tree model, its European counterpart, which is quickly evaluated using the Black-Scholes formula, can serve as lower fidelity data.

Remark 4. *Barone-Adesi and Whaley [2] propose an analytic approximation of an American option which is equal to the value of its European counterpart plus an add-on or their intrinsic values with a switch condition detailed in Section A.*

Figure 2 and Table 2 demonstrate the numerical test of an American put of strike 62. The configuration of this experiment is detailed in Section 5 and 6.1. In particular, the GPR is trained with only 5 American put prices, while the mGPR integrates additionally 10 European prices as low-fidelity data. All training points are drawn from a regular grid of the underlying asset within an interval$[1, 140]$, and test points are uniformly sampled within the same interval. We observe a significant mismatch of the GPR estimate (cf. blue curve), notably in the in-the-money region, whereas the mGPR estimate (cf. yellow curve) satisfactorily fits the American put price computed using a binomial tree with 100 time steps. Additionally, the mGPR provides smaller prediction uncertainty represented by the light yellow interval, which closely aligns with the yellow curve and is visually indistinguishable in Figure 2. For further comparison, we also implement a second standard GPR, learning the discrepancy between American and European prices, and the approximation introduced in Barone-Adesi and Whaley [2] (see Section A). Their results are then reported in Table 2.

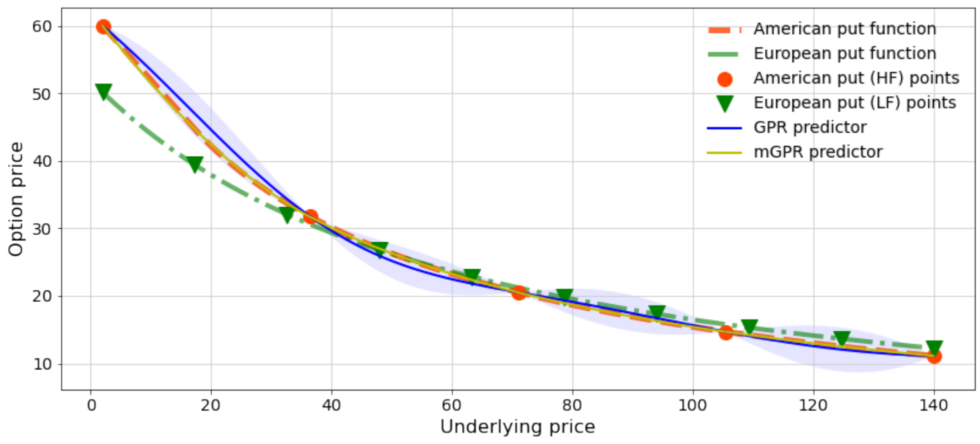

Figure 2. American put prices predicted by GPRs learned from 5 points. Red circles represent training points in GPR and high-fidelity points in mGPR, and green inverted triangles are low-fidelity points in mGPR. The light blue and light yellow (very close to yellow curve) regions depict the uncertainty of the GPR and mGPR predictions respectively.

Table 2. Out-of-sample mean absolute error (MAE) of the predictors against the American put price computed by binomial tree of 100 time steps. The GPR with control variate learns the discrepancy between American price and its European counterpart. The BW approximation refers to the Barone-Adesi and Whaley [2] approach.

Model	GPR	GPR with control variate	BW approximation	mGPR
MAE	0.6848	0.3199	0.2859	0.1367

4.3. Conditional Distribution of the Estimate

We describe now the posterior mean and variance of the prediction once the model is completely learned. Following the demonstration in Forrester et al. [20, Appendix A], for a new point $\mathbf{x}^* \in \mathcal{X}$, the prediction of the high-fidelity model follows normal distribution:

$$f_h(\mathbf{x}^*) \mid \mathbf{f}_l = \mathbf{y}_l, \mathbf{f}_h = \mathbf{y}_h, (\beta_l, \sigma_l, \boldsymbol{\theta}_l), (\rho, \beta_h, \sigma_h, \boldsymbol{\theta}_h) \sim \mathcal{N}(m_h(\mathbf{x}^*), s_h^2(\mathbf{x}^*)), \tag{19}$$

with mean and variance functions:

$$\begin{aligned} m_h(\mathbf{x}^*) &= \beta + K(\mathbf{x}^*)^\top \mathbf{K}^{-1}(\mathbf{y} - \beta), \\ s_h^2(\mathbf{x}^*) &= \rho^2\sigma_l^2 + \sigma_h^2 - K(\mathbf{x}^*)^\top \mathbf{K}^{-1} K(\mathbf{x}^*), \end{aligned} \tag{20}$$

where

$$\beta = \frac{\text{ß}^\top \mathbf{K}^{-1}\mathbf{y}}{\text{ß}^\top \mathbf{K}^{-1}\text{ß}}, \qquad \text{with} \qquad \mathbf{y} = \begin{bmatrix} \mathbf{y}_l \\ \vdots \\ \mathbf{y}_h \end{bmatrix}. \tag{21}$$

In addition, $K(\mathbf{x}^*)^\top$ is the covariance function between $f_h(\mathbf{x}^*)$ and $[f_h(\mathbf{x}_1) \dots f_h(\mathbf{x}_{N_l}), f_h(\mathbf{X}_h)]$

$$K(\mathbf{x}^*)^\top = \Big[\rho\sigma_l^2 k_{\boldsymbol{\theta}_l}(\mathbf{x}^*, \mathbf{X}_l) \quad \rho^2\sigma_l^2 k_{\boldsymbol{\theta}_l}(\mathbf{x}^*, \mathbf{X}_h) + \sigma_h^2 k_{\boldsymbol{\theta}_h}(\mathbf{x}^*, \mathbf{X}_h)\Big], \tag{22}$$

and the covariance matrix $\mathbf{K}$ of vector $(\mathbf{f}_l, \mathbf{f}_h)$ reads

$$\mathbf{K} = \begin{pmatrix} \sigma_l^2 K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l) & \rho\sigma_l^2 K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_h) \\ \rho\sigma_l^2 K_{\boldsymbol{\theta}_l}(\mathbf{X}_h, \mathbf{X}_l) & \rho^2\sigma_l^2 K_{\boldsymbol{\theta}_l}(X_h, X_h) + \sigma_h^2 K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h) \end{pmatrix}. \tag{23}$$

We can notice that the computation of (20) involves reversing the massive covariance matrix $\mathbf{K}$ of size $(N_l + N_h) \times (N_l + N_h)$. To reduce the computational complexity, one can apply the Schur complement of block matrix $\mathbf{K}$ which is thus presented by Proposition 3.1 in Le Gratiet [32].

prop 4.1. *By sorting the input data such that $\mathbf{X}_l = (\mathbf{X}_l \backslash \mathbf{X}_h, \mathbf{X}_h)$ (cf. (15)) and with all above notations, the inverse matrix of $\mathbf{K}$ in (23) has the form:*

$$\mathbf{K}^{-1} = \begin{pmatrix} \sigma_l^{-2} K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)^{-1} + \begin{pmatrix} 0 & 0 \\ 0 & \rho^2\sigma_h^{-2} K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1} \end{pmatrix} & -\begin{pmatrix} 0 \\ \rho\sigma_h^{-2} K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1} \end{pmatrix} \\ -\begin{pmatrix} 0 & \rho\sigma_h^{-2} K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1} \end{pmatrix} & \sigma_h^{-2} K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1} \end{pmatrix}. \tag{24}$$

Instead of performing the inversion matrix in (23) which has a complexity of $\mathcal{O}((N_l + N_h)^3)$, the computation of $\mathbf{K}^{-1}$ in (24), involving the inverse of $K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)^{-1}$ and $K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1}$, has a much lower complexity of $\mathcal{O}(N_l^3 + N_h^3)$. By substituting (24) into (20) and doing some calculus, one can simplify the above formulas as follows

$$\beta = \frac{\sigma_l^{-2}\text{ß}^\top K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)^{-1}\mathbf{y}_l + (1-\rho)\sigma_h^{-2} K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1}(\mathbf{y}_h - \rho[y_{l_{N_l - N_h + 1}} \dots y_{l_{N_l}}]^T)}{\sigma_l^{-2}\text{ß}^\top K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)^{-1}\text{ß} + (1-\rho)^2\sigma_h^{-2}\text{ß}^\top K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1}\text{ß}}, \tag{25}$$

yielding the posterior mean and variance

$$\begin{aligned} m_h(\mathbf{x}^*) &= \beta + \rho K_{\boldsymbol{\theta}_l}(\mathbf{x}^*, \mathbf{X}_l) K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)^{-1}(\mathbf{y}_l - \beta) + K_{\boldsymbol{\theta}_h}(\mathbf{x}^*, \mathbf{X}_h) K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1}(\mathbf{y}_h - \beta), \\ s_h(\mathbf{x}^*) &= \sigma_h^2 + \rho\sigma_l^2 - \sigma_h^2 K_{\boldsymbol{\theta}_h}(\mathbf{x}^*, \mathbf{X}_h) K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1} K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{x}^*) \\ &\quad - \rho^2\sigma_l^2 K_{\boldsymbol{\theta}_l}(\mathbf{x}^*, \mathbf{X}_l) K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)^{-1} K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{x}^*). \end{aligned} \tag{26}$$

4.4. Bayesian Estimation of Model Parameters

Due to the conditional independence assumption in the second line of (16), the first set $(\beta_l, \sigma_l, \boldsymbol{\theta}_l)$ can be separately learnt by, for example, the maximum likelihood method using data set $\mathcal{D}_l$. While the parameters of the second Gaussian processes $(\beta_h, \sigma_h, \boldsymbol{\theta}_h)$ and the correlation parameter ρ need to be jointly estimated to carry out the idea of formulation (16) ([32] Section 3.3.2). Similarly to (12), the parameters of the low-fidelity model are estimated by maximizing the following log-likelihood

$$-\frac{1}{2}\left[N_l \log(\sigma_l^2) + \log(\det(K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)) + \frac{(\mathbf{y}_l - \beta_l)^\top K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)^{-1}(\mathbf{y}_l - \beta_l)}{\sigma_l^2}\right] \tag{27}$$

Differentiating the above equation w.r.t. β_l and σ_l^2 and solving it at 0 yield

$$\hat{\beta}_l = \frac{ß^\top K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)^{-1}\mathbf{y}_l}{ß^\top K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)^{-1}ß}, \tag{28}$$

and

$$\hat{\sigma}_l^2 = \frac{1}{N_l}(\mathbf{y}_l - \hat{\beta}_l)^\top K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)^{-1}(\mathbf{y}_l - \hat{\beta}_l). \tag{29}$$

By substituting (28) and (29) into (27), we have the following optimization problem

$$\hat{\boldsymbol{\theta}}_l = \arg\min_{\boldsymbol{\theta}_l}\left[N_l \log(\hat{\sigma}_l^2) + \log(\det(K_{\boldsymbol{\theta}_l}(\mathbf{X}_l, \mathbf{X}_l)))\right]. \tag{30}$$

To learn the second Gaussian process model, we define

$$\mathbf{d} = \mathbf{y}_h - \rho \begin{pmatrix} y_{l_{N_l - N_h + 1}} \\ \vdots \\ y_{l_{N_l}} \end{pmatrix}. \tag{31}$$

The log-likelihood of the model reads

$$-\frac{1}{2}\left[N_h \log(\sigma_h^2) + \log(\det(K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)) + \frac{(\mathbf{d} - \beta_h)^\top K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1}(\mathbf{d} - \beta_h)}{\sigma_h^2}\right], \tag{32}$$

yieling the MLEs of

$$\hat{\beta}_h = \frac{ß^\top K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1}\mathbf{d}}{ß^\top K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1}ß}, \quad \text{and} \tag{33}$$

$$\hat{\sigma}_h^2 = \frac{1}{N_l}(\text{-}\hat{\beta}_h)^\top K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)^{-1}(\text{-}\hat{\beta}_h). \tag{34}$$

To compute the above parameters, one needs to estimate $\hat{\boldsymbol{\theta}}_h, \rho$ by solving the following optimization

$$\hat{\boldsymbol{\theta}}_h, \rho = \arg\min_{\boldsymbol{\theta}_h, \rho}\left[N_h \log(\hat{\sigma}_h^2) + \log(\det(K_{\boldsymbol{\theta}_h}(\mathbf{X}_h, \mathbf{X}_h)))\right]. \tag{35}$$

Note that (35) can be directly solved by numerical methods mentioned in Section 3.2 since the inversion of the small matrix of size $N_h \times N_h$ is not problematic. While the same procedure may not be applied for system (30) when the number of low-fidelity N_l is large. We then present in the next subsection the sparse Gaussian process model for low-fidelity model.

5. Experiment Design and Model Specification

In this section, we present numerical tests dedicated to evaluating the accuracy and performance of the algorithms studied earlier in this chapter. More specifically, these experiments are conducted on two trading portfolios (cf. Section 6.1 and 6.3), interspersed by an examination of individual multi-asset calls (cf. Section 6.2). The first one concerns a portfolio of single underlying derivatives, including plain vanilla (call and put), barrier and American options. The second one consists of multi-asset options, such as geometric, basket, best-of- and worst-of- options. The comparison is based on benchmark methods and scores explained below. We conclude this section by specifying the model of the implemented algorithms.

First, we analyze the case of the single-asset options portfolio described in Lehdili et al. [34]. This portfolio, with a time maturity of ten years, consists of 100 distinct calls and puts issued on each of the four considered underlying assets. In particular, each asset is used to write 10 vanilla, 10 barrier and 5 American options with their strikes randomly generated from 60 to 140. More details about market and product parameters in the portfolio are presented in Lehdili et al. [34, Tables 1 and A.1]. We then evaluate the price of each multi-asset option, as indicated in Table 1. Each option expires after two years and is based on the same five assets. The actual values of geometric call options can be calculated using the Black-Scholes formula and serve as a reference. In the case of high dimensionality, the considered portfolio comprises 500 derivatives, depending on the performance of 20 stocks, generating a total of 20 market risk factors. Each derivative is characterized by various parameters, including a random selection of underlying assets from 2 to 20, a strike price ranging from 80 to 160, and a number of options from 100 to 500. It is noteworthy that each derivative can be either a call or a put option among the four types introduced in Section 2.2. The portfolio maturity is set at 5 years, meaning that all its derivatives have a maturity ranging from 6 months to 5 years. In the Black-Scholes model framework, we choose a risk-free rate of 1%. Initial asset prices are arbitrarily chosen within a range of 100 to 120, and the volatility of these assets is set to vary between 0.2 and 0.5 with a random correlation matrix, refer to Table 3 and Figure 3. Finally, we examine the pricing and risk calculation problem for different time horizons, namely one day, ten days, and one month. At longer horizons, the magnitude of shocks and their volatilities increase, leading to a significant challenge in adapting the number of interpolation points. Financially, the price function becomes less linear, and the tail of the potential loss distribution becomes heavier.

Table 3. Initial stock prices.

S_0^1	103.79	S_0^{11}	115.33
S_0^2	100.41	S_0^{12}	102.28
S_0^3	109.73	S_0^{13}	118.77
S_0^4	115.22	S_0^{14}	118
S_0^5	108.82	S_0^{15}	103.47
S_0^6	110.19	S_0^{16}	113.28
S_0^7	100.27	S_0^{17}	100.43
S_0^8	110.23	S_0^{18}	102.45
S_0^9	119.78	S_0^{19}	106.64
S_0^{10}	117.87	S_0^{20}	103.32

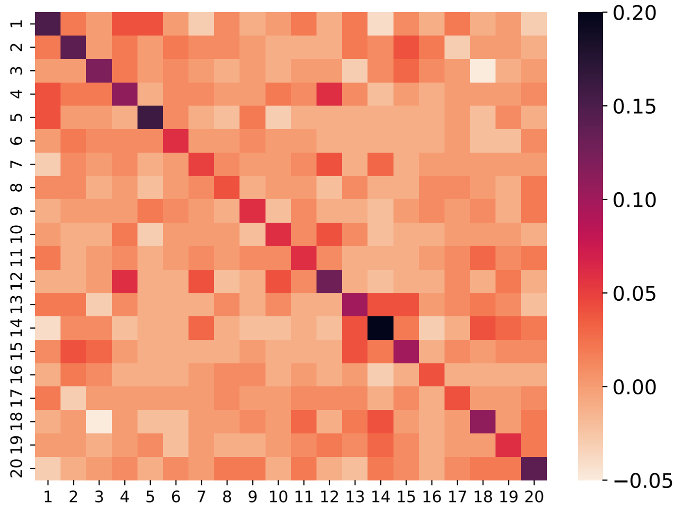

Figure 3. Heatmap of the stock covariance matrix.

5.1. Benchmarking

For the numerical implementation of market risk metrics calculation, the full pricing approach uses $N = 100,000$ paths in any case. This level is chosen to ensure a satisfactory level of convergence of high quantiles estimation. However, a limited number (N_{train}) of these prices is chosen to train the machine learning models.

Regarding the exact pricing engine, the value of vanilla and barrier options in the mono-asset derivatives portfolio are calculated by analytical formulas, while the binomial tree with 100 time steps is used to evaluate American options. For multi-asset options, the exact pricing engine is defined by a Monte Carlo simulation of $100,000$ paths, except for geometric options whose the price can be computed by the Black-Scholes formula.

We implement the sensitivity-based pricing approximation (SxS) in Section B. To get sensitivities data, we shock relatively 0.1% of each risk factor and use the central finite difference, yielding totally 40 portfolio evaluations for the first order sensitivity-based approach. The second order sensitivities are numerical unstable and lead to a worse approximation of the price. Hence they are not considered in our numerics.

Regarding machine learning methods, we implement a two-hidden-layer neural network (see Section C) (NN) and two Gaussian process models introduced in Section 3 (GPR) and 4 (mGPR). In the latter, the high-fidelity data consists of the $100,000$ Monte Carlo paths prices, while the low-fidelity comprises weak prices based on 100 MC paths in Section 6.2 and 500 MC paths in Section 6.3. For measuring the pricing mismatch, we use the mean absolute percentage error (MAPE) over $N = 100,000$ scenarios in the VaR and ES computation:

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{p}_0(S_h) - p_0(S_h)|}{p_0(S_0)}, \tag{36}$$

where $p_0(S_h)$ and $\hat{p}_0(S_h)$ are respectively actual price (i.e. analytical or Monte Carlo price) and predicted price (e.g. GPR approximative price), and $p_0(S_0)$ is the actual price without shock. The MAPE evaluating the average relative error of the estimate and the true observation is a common choice for benchmarking in finance. For risk assessment purpose, we measure the error of value-at-risk and expected shortfall calculated based on surrogate models (i.e. $\widehat{\text{VaR}}(\cdot, h, \alpha)$ and $\widehat{\text{ES}}(\cdot, h, \alpha)$) against the full revaluation ones calculated based on actual prices (i.e. $\mathrm{VaR}(\cdot, h, \alpha)$ and $\mathrm{ES}(\cdot, h, \alpha)$). For the relative measure, the error is then standardised by today price $p_0(S_0)$

$$\text{err. VaR}\,\alpha = \frac{\left|\widehat{\mathrm{VaR}}(\cdot, h, \alpha) - \mathrm{VaR}(\cdot, h, \alpha)\right|}{p_0(S_0)}, \quad \text{err. ES}\,\alpha = \frac{\left|\widehat{\mathrm{ES}}(\cdot, h, \alpha) - \mathrm{ES}(\cdot, h, \alpha)\right|}{p_0(S_0)}. \tag{37}$$

Finally, the computation time is also reported to compare the resource efficiency.

5.2. Model Specification

For neural network model, we train a two-hidden-layer Relu network of 50 hidden units (cf. Section C). We observe numerically that more complex architectures do not significantly improve the approximation performance. Network parameters are calibrated by the stochastic gradient descent algorithm, i.e. Adam optimizer [31], using 2000 epochs. The learning rate is set at 0.01 and each batch takes one fifth of the training data size.

The standard Gaussian process regression is trained using the scikit-learn Python package. We set a zero mean prior distribution and use Matern kernels with $\nu = 2.5$ (see (7)) to model the covariance matrix. A single length-scale is selected; however, the input is normalized by subtracting its empirical mean and then dividing by its empirical standard error. The optimization is restarted 5 times for the numerical stability. To implement multi-fidelity Gaussian process regression, we use emukit package [43] which is built upon the Gpy framework developed by machine learning researchers from the University of Sheffield. The model configuration is either set as the standard Gaussian processes or left at its default. The optimization is rerun 5 times.

6. Numerical Results

The main aim of this section is to provide a numerical analysis of algorithms based on Gaussian process regression (GPR) and the multi-fidelity model (mGPR) when applied to the repeated valuation and market risk calculation of an equity trading portfolio. It is worth emphasizing that a key advantage of the proposed new methods is that valuation and risk assessment are performed at the portfolio level, rather than on a deal-by-deal basis, which is the case with traditional approaches.

6.1. Mono-Asset Options Portfolio Case

Figures 4 and 5 and Table 4 outline the performance of Gaussian Process Regression (GPR) in the full revaluation and risk calculation of a mono-asset portfolio with the sub-portfolio techniques outlined in Section 3.3. The accuracy of the GPR is evident in both pricing and risk calculation. For this mono-asset experiment, we train the regression models by a regular grid and backtest with more (uniformly) random data [14,34]. In particular, the training and test stock prices are shocked around their initial values using their diffusion models, e.g.

$$\left[S_0^i \exp\left((r - \frac{(\sigma^i)^2}{2})h - 3\sqrt{h}\sigma^i\right), S_0^i \exp\left((r - \frac{(\sigma^i)^2}{2})h + 3\sqrt{h}\sigma^i\right)\right],$$

where S_0^i, σ^i, r and h denote respectively the initial stock price and volatility of the i-th asset, and the common interest rate and time step.

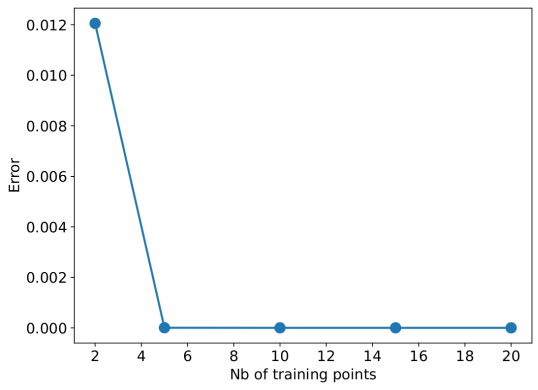

Figure 4. Convergence of MAPE of GPR estimates when increasing the number of training points in mono-asset portfolio case.

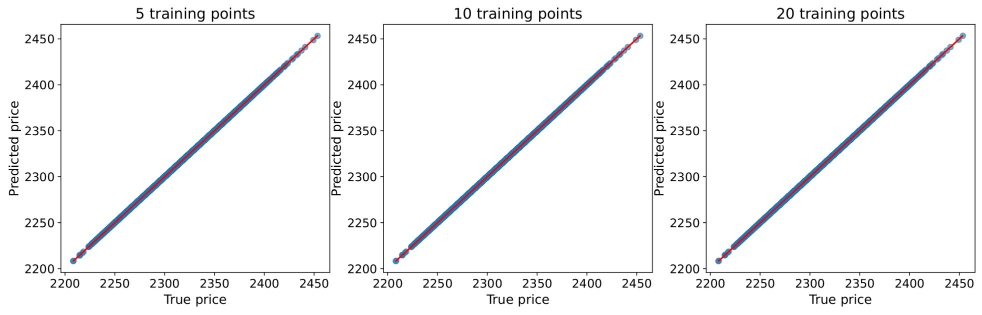

Figure 5. q-q plots of GPR predicted price using 5 (*Left*), 10 (*Center*) and 20 (*Right*) training points against the true price, i.e. analytical and binomial tree prices, in the mono-asset portfolio case.

In Figures 4 and 5, a precise fit of the price function is achieved when employing 5 points or more for training each sub-portfolio. The risk measures estimated by the GPR approach consistently converge to the benchmark measures obtained by the full revaluation approach, i.e. the intensive Monte Carlo method with 100,000 paths. The latter takes approximately 2 hours and 25 minutes on our machine. When learning from 10 points (resp. 5 points), the GPR achives a speed up of 1,000 times (resp. 2,000 points) compared to the full revaluation approach. The high runtime of the full revaluation approach in this experiment is primarily due to the American pricing calculations using a binomial tree with 100 time steps. This section concludes the performance of the Gaussian process regression in the mono-asset portfolio pricing and risk calculation.

Table 4. VaR and ES estimates using GPR against the true measures in the mono-asset portfolio case.

	Confidence level	True measure	$N_{train} = 5$	$N_{train} = 10$	$N_{train} = 20$
VaR	90%	39.83	39.84	39.83	39.83
	95%	50.90	50.89	50.92	50.91
	97.5%	60.28	60.29	60.29	60.29
	99%	70.53	70.55	70.56	70.54
ES	90%	53.98	53.98	53.99	53.98
	95%	63.02	63.02	63.03	63.02
	97.5%	70.95	70.95	70.97	70.96
	99%	80.15	80.15	80.17	80.15
	Speed-up	2h25 - benchmark	x2000	x1000	x500

6.2. Multi-Asset Options Case

Figure 6 and Table 5 display the results of the pricing and risk calculation for a five-asset geometric average call. Both GPR and mGPR models are trained using Monte Carlo prices with 100,000 paths, while the analytical prices are used as a reference for backtesting. Additionally, mGPR uses 200 prices obtained through the Monte Carlo method with 100 paths as low-fidelity data, which is still less expensive than one price point used 100,000 paths. The top left plot in Figure 6 shows that two GPR models trained with 50 points can outperform Monte Carlo prices. The error decreases, reaching 10 basis points at 200 training points. Notably, mGPR outperforms GPR at lower numbers of data points before their error curves converge.

The quantile-quantile (q-q) plots in Figure 6 trace the left tail of the simulated price distribution below the 10th percentile, corresponding to the right tail of the loss in the risk calculation in Table 5. Despite the Monte Carlo price having a higher MAPE error compared to GPR prices, its tail perfectly matches the true price. The tails of GPR and mGPR prices also converge as the number of training points increases. We stop the convergence curve of mGPR at 200 training points (corresponding to high-fidelity data) because we only use 200 low-fidelity points. In Table 5, the one-day VaR 99% and ES 97.5% estimated by GPRs using 200 data points have an error of around 7 basis points. Notably, GPR models significantly save time in risk calculation, including sampling and training time with 200 points (13 seconds for GPR compared to 7488 seconds for the brute force Monte Carlo approach). The mGPR model remains an effective approximation for risk measurement, especially in cases of high pricing engine costs, such as with complex portfolios, and when using minimal price points is preferred.

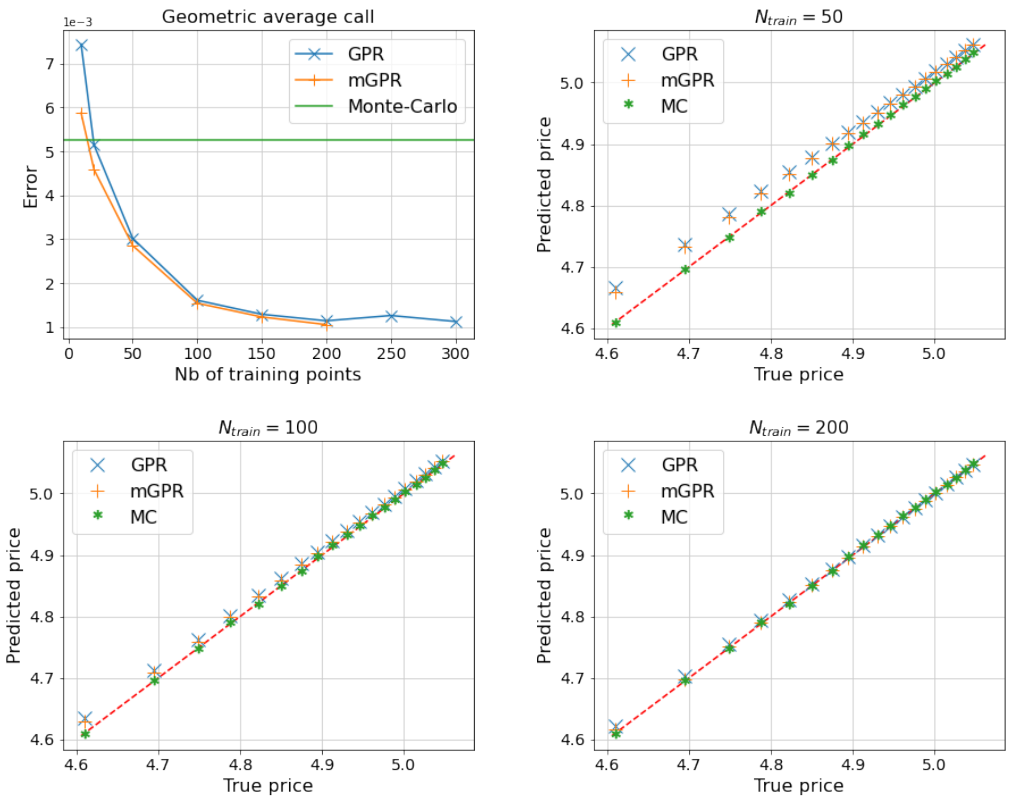

Figure 6. MAPE of estimates of 5-asset geometric average call as a function of the number of training points (*Top Left*) and q-q plots comparing predictors learned from 50 (*Top Right*), 100 (*Bottom Left*) and 200 (*Bottom Right*) points against the true price, i.e. Black-Scholes price, below their 10% quantile levels. The mGPR uses 200 low-fidelity points.

Table 5. One day VaR 99% and ES 97.5% of 5-asset geometric average call estimated by proxy pricing models.

	Model	Number of training points (N_{train})					
		10	20	50	100	150	200
VaR 99%	GPR	0.6900	0.7639	0.7768	0.8007	0.7989	0.8108
	mGPR	0.7749	0.8028	0.7815	0.8054	0.8018	0.8155
	MC	0.8220					
	True	0.8190					
ES 97.5%	GPR	0.6931	0.7650	0.7757	0.8011	0.8005	0.8108
	mGPR	0.7781	0.8060	0.7811	0.8057	0.8033	0.8159
	MC	0.8235					
	True	0.8202					
Computational time (in second)	GPR	0	1	3	6	10	13
	mGPR	10	10	12	23	30	32
	MC	7488					
True initial price		5.5135					

Figures 7–9 and Tables 6–8 show the results for other options, with similar computational times to those in Table 5. Most of the remarks made in the geometric average call case are still valid for these other cases. In the arithmetic average basket call case, the risk measures computed by two GPR models have an error of about 23 basis points, which is less than 10 basis points in the two other cases. These above errors can be computed by applying results in Tables 6–8 into (37).

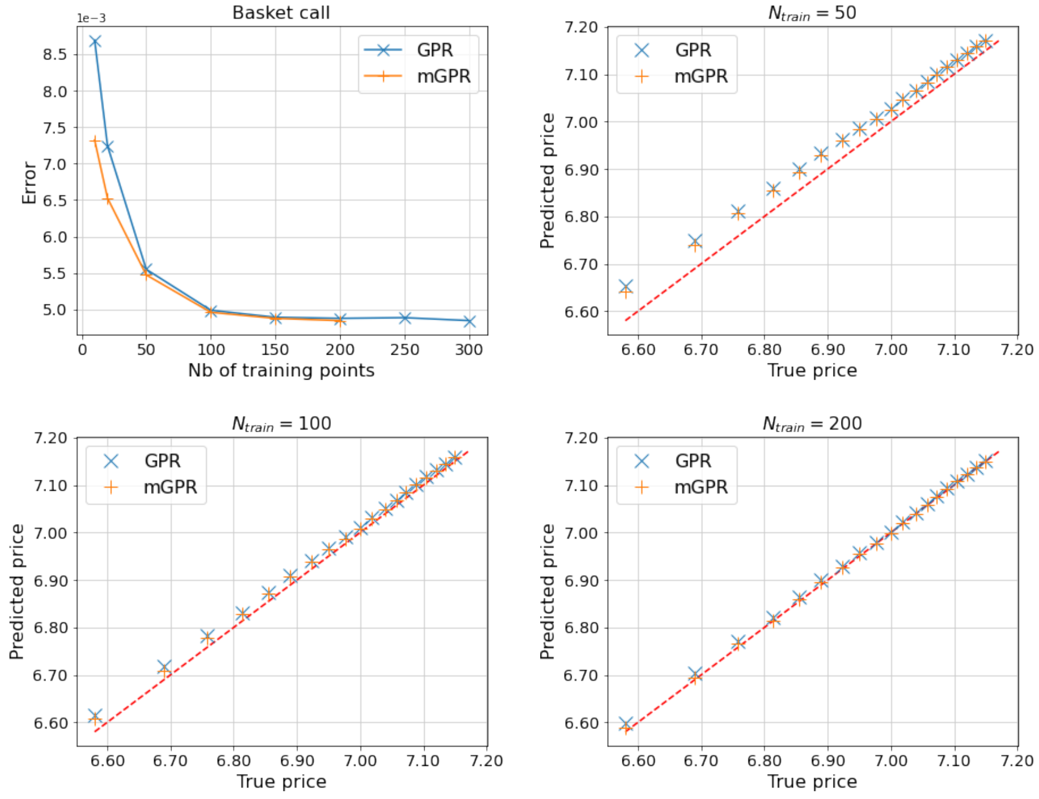

Figure 7. MAPE of estimates of 5-asset basket call as a function of the number of training points (*Top Left*) and q-q plots comparing predictors learned from 50 (*Top Right*), 100 (*Bottom Left*) and 200 (*Bottom Right*) points against the true price, i.e. Monte-Carlo price with 100,000 paths, below their 10% quantile levels.

Table 6. One day VaR 99% and ES 97.5% of 5-asset basket call estimated by proxy pricing models.

	Model	Number of training points (N_{train})					
		10	20	50	100	150	200
VaR 99%	GPR	0.9163	1.0067	1.0289	1.0597	1.0588	1.0732
	mGPR	1.0123	1.0550	1.0371	1.0673	1.0630	1.0824
	True	1.1013					
ES 97.5%	GPR	0.9199	1.0077	1.0280	1.0615	1.0607	1.0748
	mGPR	1.0143	1.0583	1.0371	1.0674	1.0643	1.0832
	True	1.1016					
True initial price		7.7770					

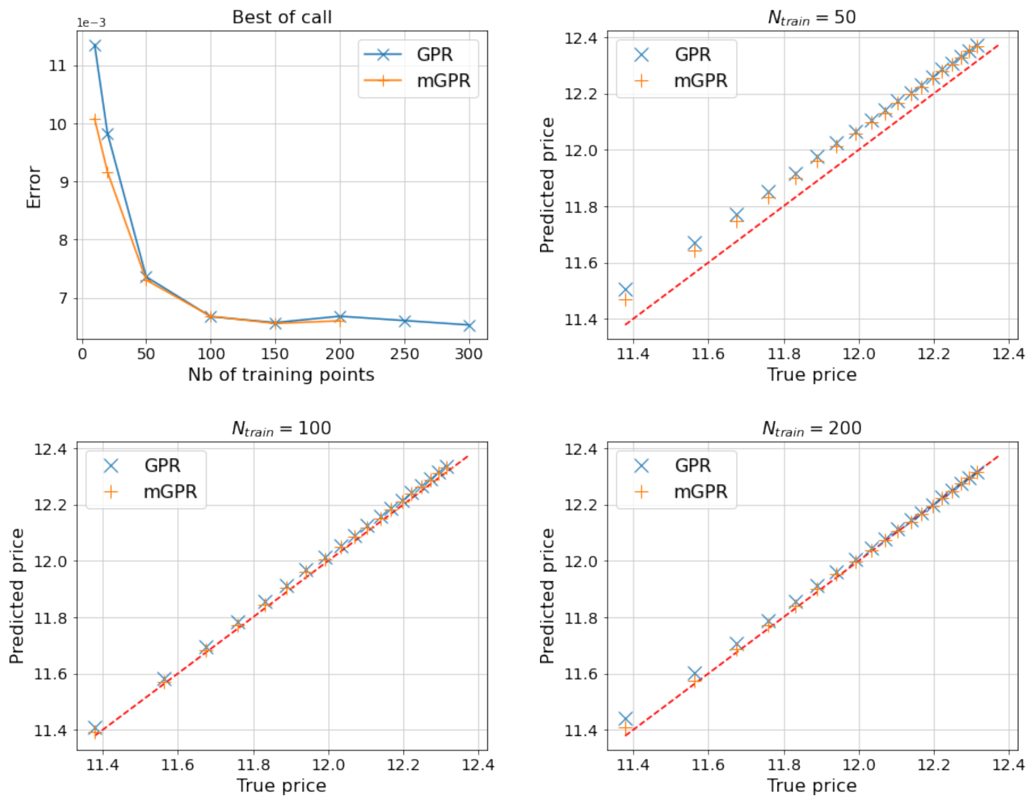

Figure 8. MAPE of estimates of 5-asset best-of-call as a function of the number of training points (*Top Left*) and q-q plots comparing predictors learned from 50 (*Top Right*), 100 (*Bottom Left*) and 200 (*Bottom Right*) points against the true price, i.e. Monte-Carlo price with 100,000 paths, below their 10% quantile levels.

Table 7. One day VaR 99% and ES 97.5% of 5-asset best-of-call estimated by proxy pricing models.

	Model	Number of training points (N_{train})					
		10	20	50	100	150	200
VaR 99%	GPR	1.4704	1.5850	1.6380	1.7258	1.6907	1.7064
	mGPR	1.5869	1.6604	1.6682	1.7403	1.6982	1.7346
	True	1.7483					
ES 97.5%	GPR	1.4714	1.5878	1.6435	1.7279	1.6928	1.7084
	mGPR	1.5906	1.6675	1.6721	1.7438	1.7010	1.7373
	True	1.7517					
True initial price		13.3101					

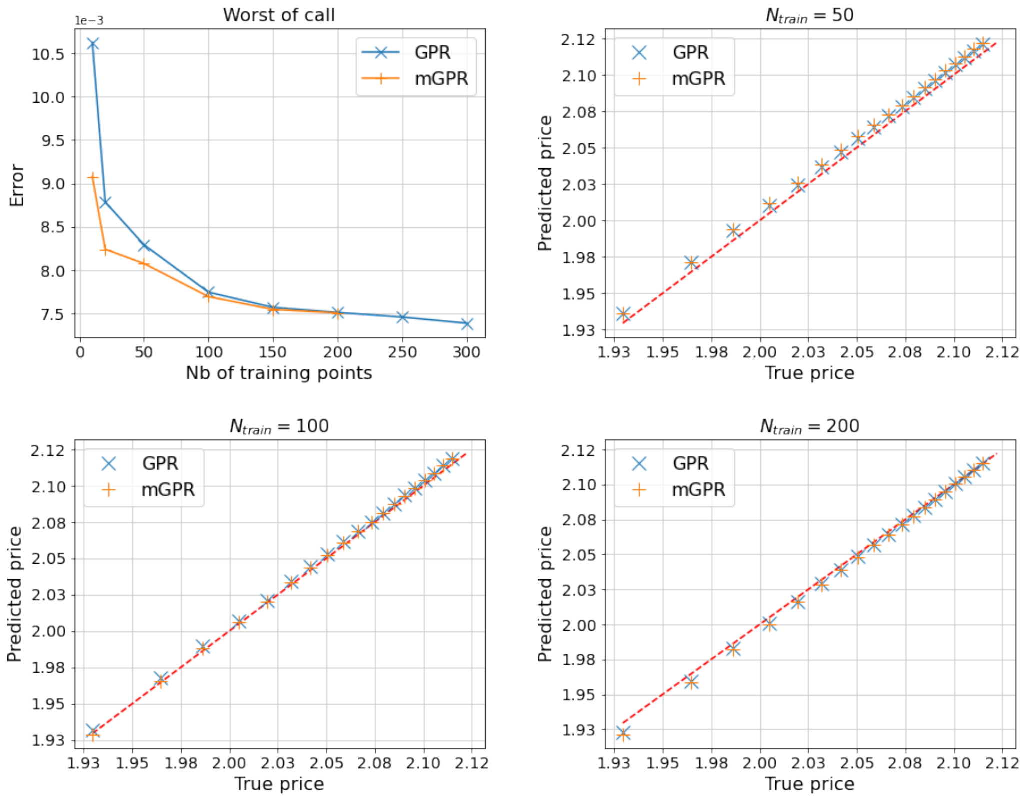

Figure 9. MAPE of estimates of 5-asset worst-of call as a function of the number of training points (*Top Left*) and q-q plots comparing predictors learned from 50 (*Top Right*), 100 (*Bottom Left*) and 200 (*Bottom Right*) points against the true price, i.e. Monte-Carlo price with 100, 000 paths, below their 10% quantile levels.

Table 8. One day VaR 99% and ES 97.5% of 5-asset worst-of call estimated by proxy pricing models.

	Model	Number of training points (N_{train})					
		10	20	50	100	150	200
VaR 99%	GPR	0.3153	0.3447	0.3584	0.3619	0.3661	0.3704
	mGPR	0.3625	0.3673	0.3584	0.3640	0.3677	0.3711
	True	0.3651					
ES 97.5%	GPR	0.3625	0.3673	0.3584	0.3640	0.3677	0.3711
	mGPR	0.3624	0.3670	0.3587	0.3645	0.3681	0.3715
	True	0.3657					
True initial price		2.3296					

6.3. Multi-Asset Options Portfolio Case

Figure 10 and Table 9 illustrate the results of a multi-asset options portfolio . The time horizon of the value-at-risk and expected shortfall is set at one day. The second-order sensitivity-based approach (SxS) and the neural network learning from 40 points are not reported here due to their low precision. The learning of mGPR incorporates 500 prices computed by 500 Monte Carlo paths as low-fidelity data. Despite the portfolio complexity, both GPR models (GPR and mGPR) achieve impressive results with only 40 points. The GPR model has a MAPE of 0.61%, slightly outperformed by the mGPR model with a MAPE of 0.57%. While SxS performs about 10 times worse with an error of 6.13%. Figure 10 highlights this difference, showing that the predicted left tail of SxS prices diverges significantly from the true price represented by the red diagonal line. While the predicted left tails of GPR and mGPR prices are not perfect, but they still concentrate around the tail of the true price, i.e. the red line.

In Table 9, mGPR predictions offer more accurate estimates of the VaR 99% and ES 97.5% compared to GPR predictions, with errors of 34bps and 32bps, respectively, versus 54bps for both errors in GPR. As more data becomes available, the performance of all models improves. Generally, two GPR models outperform NN, with a MAPE of 0.42% (0.38%) compared to the one of NN's 0.5% (0.43%) when learning with $N_{train} = 100$ ($N_{train} = 500$). The difference is more pronounced in the prediction of the left tail, as seen in the two bottom plots in Figure 10. The left tails predicted by GPR and mGPR closely align with the true one, while NN underestimates the left tail. When trained with $N_{train} = 100$, mGPR's errors in risk measures, around 30bps, are only half of GPR's errors. However, this difference disappears when training with $N_{train} = 500$.

Table 10 displays the runtime for various risk calculation methods on our server with an Intel(R) Xeon(R) Gold 5217 central processing unit (CPU) and a Nvidia Tesla V100 graphics processing unit (GPU). The full repricing approach involves simulating 100, 000 scenarios of shock, with each scenario being a Monte Carlo simulation of 100, 000 paths computed over 500 derivatives. Without advanced programming on the GPU, the sampling procedure in banks may take up to one day. Alternative approaches show significant computational performance gains. For instance, standard Gaussian processes regression achieves a speedup of 230 times compared to the full repricing approach when learning with $N_{train} = 100$ points, i.e. a total of 2.6 comparing to 600 seconds. Despite mGPR's longer learning times (i.e. 56 when learning with $N_{train} = 100$), its computational efficiency compared to full repricing remains remarkable. It is important to note that, under realistic constraints of the more time-consuming pricing engine in banks, the mGPR approach may take a bit longer than the GPR approach for the learning procedure, but it provides better precision in risk calculation, especially when limiting the use of the pricing engine. Tables 11 and 12 and Figures 11 and 12 correspond to the risk assessment results for a ten-day and one-month horizon, and observations from the one-day horizon case hold true for other time horizons.

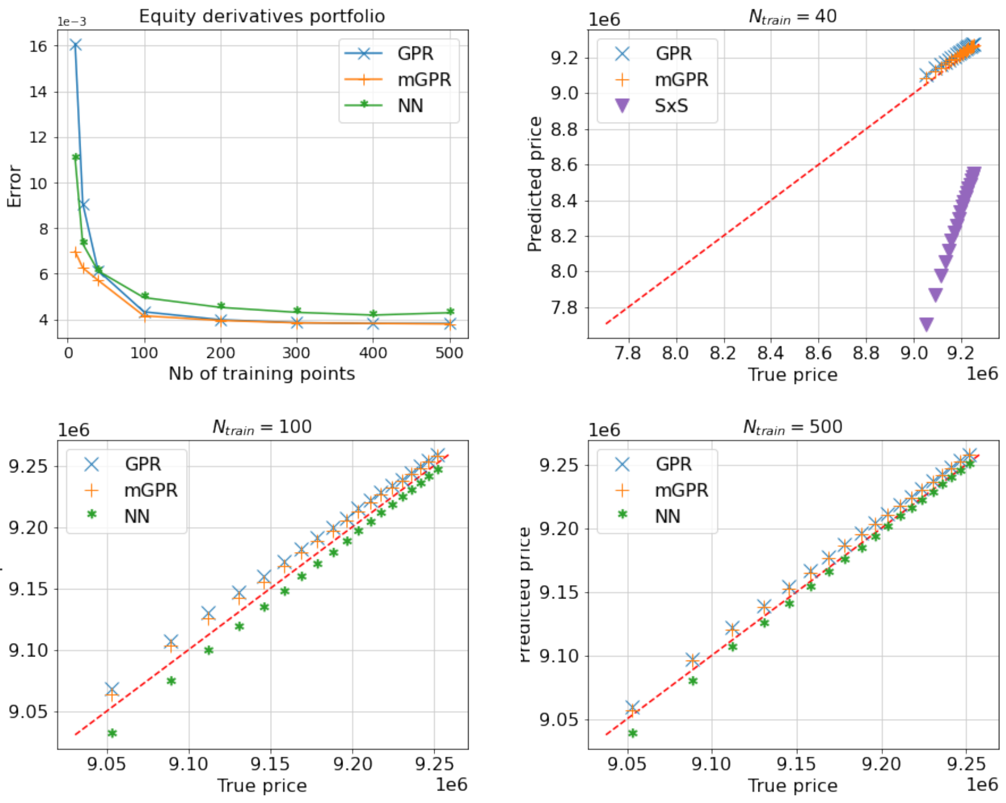

Figure 10. MAPE of estimates of equity option portfolio with one day horizon as a function of the number of training points (*Top Left*) and q-q plots comparing predictors learned from 40 (*Top Right*), 100 (*Bottom Left*) and 500 (*Bottom Right*) points against the true price, i.e. Monte-Carlo price with 100, 000 paths, below their 10% quantile levels. The MAPE of SxS prediction is not included in figures due to its high value but can be found in Table 9.

Table 9. Pricing approximation and risk calculation of the portfolio with one day horizon.

N_{train}	Full	40			100			500		
model	pricing	SxS	GPR	mGPR	NN	GPR	mGPR	NN	GPR	mGPR
MAPE	9,447,616	6.13%	0.61%	**0.57%**	0.48%	0.43%	**0.42%**	0.43%	**0.38%**	**0.38%**
VaR 99%	358,862	1,575,039	307,983	325,145	374,986	340,394	343,926	368,887	350,397	351,493
ES 97.5%	359,972	1,584,907	309,301	328,193	377,288	342,130	347,312	370,475	351,211	353,116
Err. VaR 99%	-	12.87%	0.54%	**0.36%**	0.17%	0.20%	**0.16%**	0.11%	0.09%	**0.08%**
Err. ES 97.5%	-	12.97%	0.54%	**0.34%**	0.18%	0.19%	**0.13%**	0.11%	0.09%	**0.07%**

Table 10. Computation time comparison of VaR and ES calculation by alternative methods.

N_{train}	Full	40			100			500		
model	valorisation	SxS	GPR	mGPR	NN	GPR	mGPR	NN	GPR	mGPR
Learning time	0	0	1	41	25	2	56	30	13	135
Sampling time	600	0.2			0.6			3		

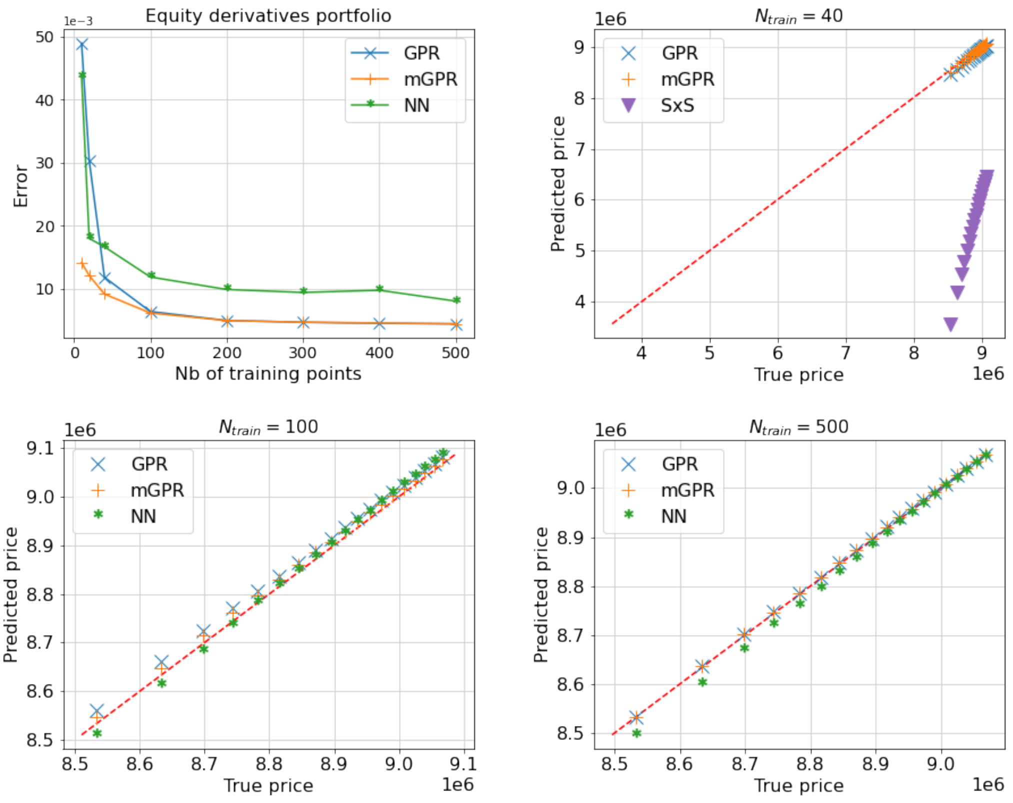

Figure 11. MAPE of estimates of equity option portfolio with ten days horizon as a function of the number of training points (*Top Left*) and q-q plots comparing predictors learned from 40 (*Top Right*), 100 (*Bottom Left*) and 500 (*Bottom Right*) points against the true price, below their 10% quantile levels. The MAPE of SxS prediction is not included in figures due to its high value but can be found in Table 11.

Table 11. Pricing approximation and risk calculation of the portfolio with ten days horizon.

N_{train}	Full	40			100			500		
model	pricing	SxS	GPR	mGPR	NN	GPR	mGPR	NN	GPR	mGPR
MAPE	9,447,616	20.27%	1.18%	**0.92%**	1.19%	0.64%	**0.61%**	0.8%	**0.44%**	**0.44%**
VaR 99%	814,166	5,252,412	896,191	843,186	835,971	785,850	799,501	847,257	810,857	811,540
ES 97.5%	814,604	5,320,871	895,574	843,831	836,584	787,204	800,163	848,514	812,423	812,947
Err. VaR 99%	-	46.98%	0.87%	**0.31%**	0.23%	0.30%	**0.16%**	0.35%	**0.04%**	**0.03%**
Err. ES 97.5%	-	47.70%	0.86%	**0.31%**	0.23%	0.29%	**0.15%**	0.36%	**0.02%**	**0.02%**

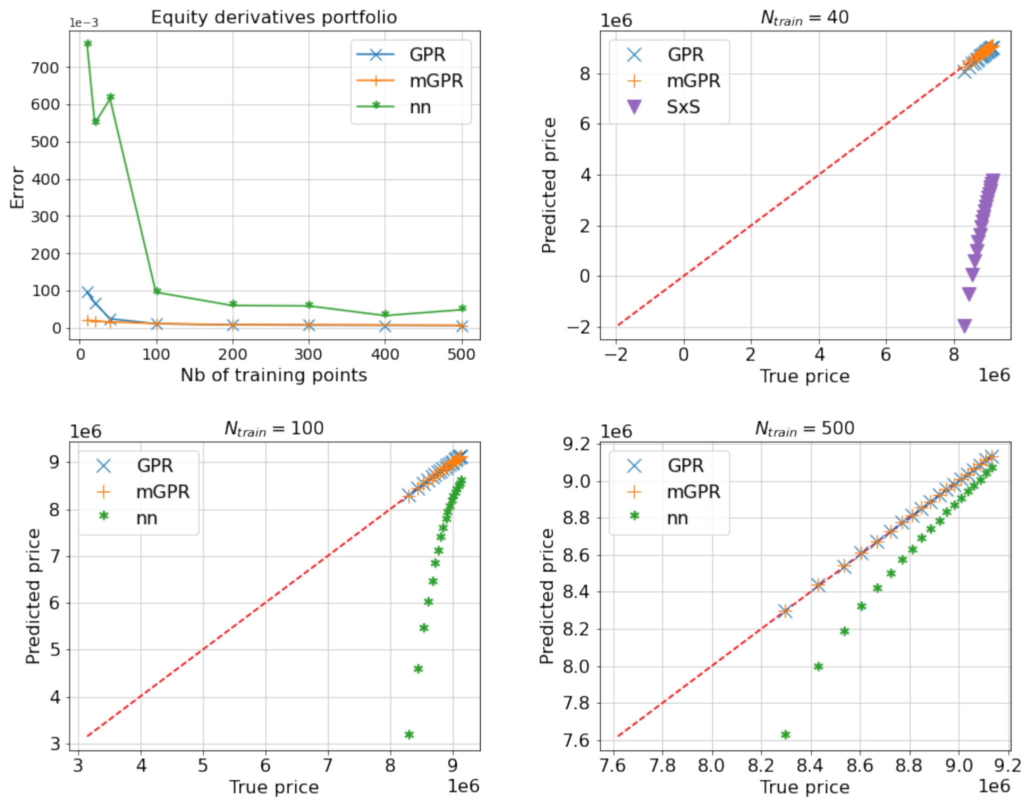

Figure 12. MAPE of estimates of equity option portfolio with one month horizon as a function of the number of training points (*Top Left*) and q-q plots comparing predictors learned from 40 (*Top Right*), 100 (*Bottom Left*) and 500 (*Bottom Right*) points against the true price, i.e. Monte-Carlo price with 100,000 paths, below their 10% quantile levels. The MAPE of SxS prediction is not included in figures due to its high value but can be found in Table 12.

Table 12. Pricing approximation and risk calculation of the portfolio with one month horizon.

N_{train}	Full	40			100			500		
model	pricing	SxS	GPR	mGPR	NN	GPR	mGPR	NN	GPR	mGPR
MAPE	9,447,616	38.44%	2.35%	**1.50%**	9.48%	**1.02%**	1.09%	1.72%	**0.60%**	**0.60%**
VaR 99%	1,016,862	10,137,189	1,175,837	1,046,029	1,092,622	995,794	1,006,735	1,103,964	1,009,737	1,010,166
ES 97.5%	1,011,890	10,267,028	1,177,443	1,042,407	1,104,343	991,431	1,004,123	1,119,376	1,005,141	1,005,065
Err. VaR 99%		96.54%	1.68%	**0.31%**	0.8%	0.22%	**0.11%**	0.92%	**0.08%**	**0.07%**
Err. ES 97.5%		97.96%	1.75%	**0.32%**	0.98%	0.22%	**0.08%**	1.14%	**0.07%**	**0.07%**

7. Conclusions

We have introduced statistical and machine learning tools to handle the repeated valuation of extensive portfolios comprising linear and non-linear derivatives. To be more precise, we applied a Bayesian non-parametric technique, known as Gaussian process regression (GPR). To evaluate the accuracy and performance of these algorithms, we have considered a multi-asset options portfolio and a portfolio of non-linear derivatives, including vanilla and barrier/American options on an equity asset. The numerical tests demonstrated that the GPR algorithm outperforms pricing models in efficiently conducting repeated valuations of large portfolios. It is noteworthy that the GPR algorithm eliminates the need to separately revalue each derivative security within the portfolio, leading to a significant speed-up in calculating value-at-risk (VaR) and expected shortfall (ES). This independence from the size and composition of the trading portfolio is remarkable. Consequently, we argue that it is more advantageous for banks to construct their risk models using the power of GPR techniques in terms of calculation accuracy and speed-up.

Moreover, we have investigated multi-fidelity modeling technique and have explored its applications in pricing and risk calculation, a relatively novel concept in quantitative finance. The multi-fidelity modeling approach leverages limited access to the price engine while incorporating information from other related, more affordable resources to enhance the approximation. Trained regression models offer price estimates at the portfolio level, making them more favorable for risk calculation compared to classical pricing approaches that evaluate products individually. Our numerical findings indicate that machine learning models such as neural networks and Gaussian process regression provide more accurate results and significant time savings compared to traditional sensitivity-based approaches, maintaining precision in risk calculation akin to the full repricing approach. With the limitation of training price data, GPR models yield more impressive results than neural network. Despite the longer learning time required for mGPR, especially when low-fidelity data is available, it yields better or at least equal precision compared to the standard GPR model. Notably, as multi-fidelity modeling is not yet well-known in quantitative finance, it presents an intriguing direction for research, allowing scientists to explore traditional relationships in finance to define low-fidelity models and enhance the learning process.

Future research delves into other topics and applications of algorithms such that Gaussian process regression and multi-fidelity modeling, with a particular focus on the fixed income portfolio. We also envisage the use of the replicating portfolio value, assumed to be accessible, as low-fidelity data to leverage the learning of the target portfolio value. Additionally, the price of European basket options can serve as low-fidelity data to approximate the price of their American/Bermudian counterparts. The Black-Scholes price of derivatives can be used to improve the approximation of their Heston price. We leave these interesting applications there for future study.

Appendix A Barone-Adesi and Whaley Approximation of American Option Values

In contrast to European options, American ones grant the holder the right to exercise the contract at any time up to the expiration date. This leads to a more complicated mathematical problem when pricing American options, for which we need to use some numerical methods, such as tree models with high time steps as mentioned in Section 2.2 and 6.1. For this reason, pioneers in quantitative finance have looked for a way to approximate the American option values. Among them, Barone-Adesi and Whaley [2] propose an analytical approximation of American option value based on the price its European counterparts, as presented below.

For a European call (put) of strike K, maturity T and underlying asset S_t following log normal dynamic (cf. (3)), its value c_{BS} (p_{BS}) is computed by the well-known Black-Scholes formula:

$$\begin{aligned} c_{BS}(S_t,t) &= \Phi(d_1(S_t))S_t - \Phi(d_2(S_t))Ke^{-r(T-t)}, \text{ and} \\ p_{BS}(S_t,t) &= -\Phi(-d_1(S_t))S_t + \Phi(-d_2(S_t))Ke^{-r(T-t)}, \end{aligned} \tag{A1}$$

where

$$\begin{aligned} d_1(\cdot) &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{\cdot}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right], \\ d_2(\cdot) &= d_1(\cdot) - \sigma\sqrt{T-t}, \end{aligned}$$

with r is risk-free rate, σ is the asset volatility and Φ is cumulative density function of the standard normal variable.

Barone-Adesi and Whaley [2] approximation of American option values reads

$$C_{BW}(S_t,t) = \begin{cases} c_{BS}(S_t,t) + A_2\left(\frac{S_t}{S^*}\right)^{q_2}, & \text{when } S_t < S^* \text{ and} \\ S_t - K, & \text{when } S_t \geq S^*, \end{cases} \tag{A2}$$

$$P_{BW}(S_t,t) = \begin{cases} p_{BS}(S_t,t) + A_1\left(\frac{S_t}{S^{**}}\right)^{q_1}, & \text{when } S_t > S^{**} \text{ and} \\ K - S_t, & \text{when } S_t \leq S^{**}, \end{cases} \tag{A3}$$

where

$$\begin{aligned}
q_1 &= \frac{1}{2}\left[-\left(\frac{2r}{\sigma^2}-1\right)-\sqrt{\left(\frac{2r}{\sigma^2}-1\right)^2+\frac{8r}{\sigma^2(1-e^{-rT})}}\right],\\
q_2 &= \frac{1}{2}\left[-\left(\frac{2r}{\sigma^2}-1\right)+\sqrt{\left(\frac{2r}{\sigma^2}-1\right)^2+\frac{8r}{\sigma^2(1-e^{-rT})}}\right],\\
A_1 &= -\frac{S^{**}}{q_1}[1-\Phi(-d_1(S^{**}))],\ A_2 = \frac{S^*}{q_2}[1-\Phi(d_1(S^*))],
\end{aligned} \tag{A4}$$

with S^* and S^{**} satisfy

$$\begin{aligned}
S^* - K &= c_{BS}(S^*,t) + \frac{S^*}{q_2}[1-\Phi(d_1(S^*))], \text{ and}\\
K - S^{**} &= p_{BS}(S^{**},t) + \frac{S^{**}}{q_1}[1-\Phi(-d_1(S^{**}))].
\end{aligned} \tag{A5}$$

Appendix B Sensitivity-Based Pricing Approximation

In order to measure the market risk by Monte Carlo without recalling expensive pricing engine, many banks use Taylor approximation for portfolio evaluation. This technique is also known as delta-gamma approximation method in literature [8,9]. With the same notation used in Section 2, for each scenario of underlying assets RF_{t+h}, the corresponding value of the portfolio $V_{t+h} = p_{t+h}(RF_{t+h})$ is approximated by

$$p_{t+h}(RF_{t+h}) \approx p_t(RF_t) + (\delta RF_h)^\top \Delta + \frac{1}{2}(\delta RF_h)^\top \Gamma\, \delta RF_h, \tag{A6}$$

where Δ and Γ are, respectively, the gradient (i.e. delta) and Hessian matrix (i.e. gamma) of the portfolio value V_t with respect to risk factor RF, and $\delta RF_h = RF_{t+h} - RF_t$. Because of the nature of (A6), pioneers in banks usually call this pricing approximation by sensitivities times shocks, abbreviated by SxS. In practice, Δ and Γ are not analytically available in closed form and they are usually estimated by finite difference (or bump and revalue) in practice. If there are d risk factors affecting the portfolio, then one needs to recall the price engine $2d$ times for estimating Δ by central finite difference [11, Chapter 8] and additionally $d(d-1)$ times for estimating Γ. This approximation pricing approach provides a significant computational improvement in risk calculation for small and medium numbers of risk factors d, i.e. less than 100. However, its precision can reach an acceptable level once several conditions are met: the portfolio value is well-approximated by a linear or quadratic functions of risk factors; the shocked risk factors RF_{t+h} remain locally in the neighborhood of their initial values RF_t; and the Δ and Γ must be accurately estimated.

Appendix C Neural Network Regression

Alongside Gaussian processes, the neural network is also a powerful non-linear model for interpolation. Different to GP regression, the neural network can be learned quickly and is scalable with the number of training points, however the model requires a sufficient training data set to learn. In this section, we provide the main notation and concept of neural networks referring the reader to LeCun, Bengio, and Hinton [33] for a detailed presentation.

We denote by $\mathcal{NN}_{d,o,h,u,\sigma}$ neural network family of h hidden layers, u hidden units and activation function σ, furthermore, this family maps input vectors in $\mathcal{X} \in \mathbb{R}^d$ to output vectors in $\mathcal{Y} \in \mathbb{R}^o$. Functions in this family are represented by the following sequential mapping (see also Figure A1),

$$\begin{aligned}
\mathbf{a}^0 &= \mathbf{z}^0 = \mathbf{x}\\
\mathbf{a}^l &= \sigma(\mathbf{z}^l) = \sigma\left(\mathbf{W}^l\mathbf{a}^{l-1} + \mathbf{b}^l\right) \quad ,l = 1,\dots,h\\
z^{h+1} &= (\mathbf{w}^{h+1})^\top \mathbf{a}^h + b^{h+1}.
\end{aligned} \tag{A7}$$

This family of functions is parameterized by weights $\mathbf{W}^1 \in \mathbb{R}^{u\times d}, \ldots, \mathbf{W}^l \in \mathbb{R}^{u\times u}, \ldots, \mathbf{w}^{h+1} \in \mathbb{R}^u$ and biases $\mathbf{b}^1 \in \mathbb{R}^u, \ldots \mathbf{b}^l \in \mathbb{R}^u, \ldots, b^{h+1} \in \mathbb{R}^o$. We denote the set of trainable parameters as $\theta = \left(\mathbf{W}^1, \ldots, \mathbf{W}^{h+1}, \mathbf{b}^1, \ldots, \mathbf{b}^h, b^{h+1}\right)$. Additionally, nonlinear activation function σ, element-wisely applied, plays a crucial role in establishing the complex mapping between inputs and outputs of the network. The rectified linear unit function (Relu), i.e. $\mathrm{Relu}(x) = \max(x, 0)$ is usually chosen, that is also our case, because of its generalisation property LeCun et al. [33, Section 6.3.1].

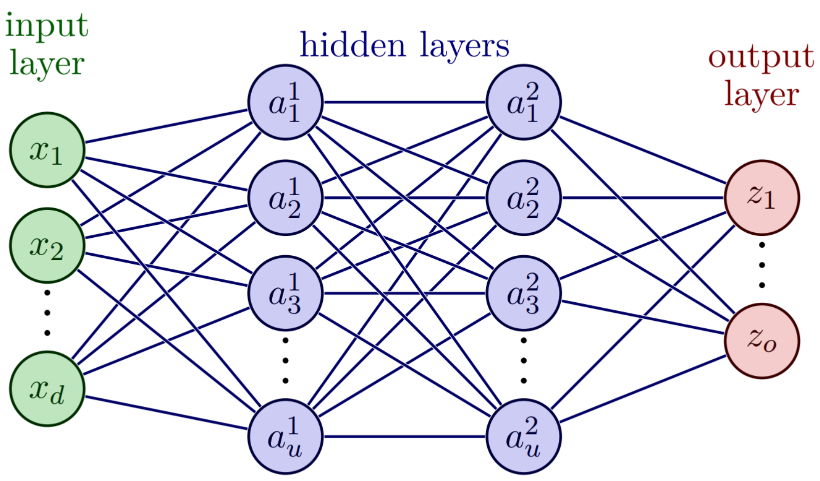

Figure A1. Two-hidden-layer neural network architecture.

Given a set of training data $\{(\mathbf{x}_i, y_i)_{i=1\ldots N_{train}}\}$, the neural network regression looks for minimizing the following mean square error

$$\hat{f} \in \underset{f\in\mathcal{NN}_{d,o,h,u,\sigma}}{\arg\min} \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \|f(\mathbf{x}_i) - y_i\|_2^2, \tag{A8}$$

where $\|\cdot\|_2$ is the Euclidean norm notation. Eqn. (A8) is solved by the combination of backpropagation technique and (stochastic) gradient descent. The detail of these algorithms can be found in (LeCun et al. [33] Chapter 8).

References

1. Abbas-Turki, L., S. Crépey, and B. Saadeddine (2023). Pathwise CVA regressions with oversimulated defaults. *Mathematical Finance 33*(2), 274–307.
2. Barone-Adesi, G. and R. E. Whaley (1987). Efficient analytic approximation of american option values. *the Journal of Finance 42*(2), 301–320.
3. Basel Committee on Banking Supervision (2013). Fundamental review of the trading book: A revised market risk framework. *Available at https://www.bis.org/publ/bcbs265.pdf*.
4. Basel Committee on Banking Supervision (2019). Minimum capital requirements for market risk. *Available at https://www.bis.org/bcbs/publ/d457.pdf*.
5. Blei, D. M., A. Kucukelbir, and J. D. McAuliffe (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association 112*(518), 859–877.
6. Brevault, L., M. Balesdent, and A. Hebbal (2020). Overview of gaussian process based multi-fidelity techniques with variable relationship between fidelities, application to aerospace systems. *Aerospace Science and Technology 107*, 106339.
7. Brigo, D., F. Mercurio, F. Rapisarda, and R. Scotti (2003). Approximated moment-matching dynamics for basket-options pricing. *Quantitative Finance 4*(1), 1–16.
8. Britten-Jones, M. and S. M. Schaefer (1999). Non-linear value-at-risk. *Review of Finance 2*(2), 161–187.
9. Broadie, M., Y. Du, and C. C. Moallemi (2015). Risk estimation via regression. *Operations Research 63*(5), 1077–1097.

10. Buehler, H., L. Gonon, J. Teichmann, and B. Wood (2019). Deep hedging. *Quantitative Finance 19*(8), 1271–1291.
11. Crépey, S. (2013). *Financial Modeling, A Backward Stochastic Differential Equations Perspective*. Springer Finance Textbook Series.
12. Crépey, S. and M. Dixon (2019). Gaussian process regression for derivative portfolio modeling and application to CVA computations. *Journal of Computational Finance 24*(1), 1–35.
13. Culkin, R. and S. R. Das (2017). Machine learning in finance: the case of deep learning for option pricing. *Journal of Investment Management 15*(4), 92–100.
14. De Spiegeleer, J., D. B. Madan, S. Reyners, and W. Schoutens (2018). Machine learning for quantitative finance: Fast derivative pricing, hedging and fitting. *Quantitative Finance 18*(10), 1635–1643.
15. Dowd, K. (2003). *An introduction to market risk measurement*. John Wiley & Sons.
16. Dowd, K. (2007). *Measuring market risk*. John Wiley & Sons.
17. Ferguson, R. and A. Green (2018). Deeply learning derivatives. *CompSciRN: Artificial Intelligence (Topic)*.
18. Fernández-Godino, M. G., C. Park, N.-H. Kim, and R. T. Haftka (2016). Review of multi-fidelity models. *Available at arXiv:1609.07196*.
19. Financial Stability Board (2017). Artificial intelligence and machine learning in financial services: Market developments and financial stability implications. *Available at https://www.fsb.org/wp-content/uploads/P011117.pdf*.
20. Forrester, A. I. J., A. Sóbester, and A. J. Keane (2007). Multi-fidelity optimization via surrogate modelling. *the Royal Society A: Mathematical, Physical and Engineering Sciences 463*, 3251 – 3269.
21. Gardner, J., G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson (2018). Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in neural information processing systems 31*, 7576–7586.
22. Gardner, J., G. Pleiss, R. Wu, K. Weinberger, and A. Wilson (2018). Product kernel interpolation for scalable Gaussian processes. *International Conference on Artificial Intelligence and Statistics 84*, 1407–1416.
23. Goudenège, L., A. Molent, and A. Zanette (2021). Variance reduction applied to machine learning for pricing Bermudan/American options in high dimension. Oleg Kudryavtsev, Antonino Zanette. *Applications of Lévy Processes*.
24. Hoffman, M. D., D. M. Blei, C. Wang, and J. Paisley (2013). Stochastic variational inference. *Journal of Machine Learning Research 14*, 1303–1347.
25. Hong, L. J., Z. Hu, and G. Liu (2014). Monte carlo methods for value-at-risk and conditional value-at-risk: a review. *ACM Transactions on Modeling and Computer Simulation (TOMACS) 24*(4), 1–37.
26. Horvath, B., A. Muguruza, and M. Tomas (2021). Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. *Quantitative Finance 21*(1), 11–27.
27. Huang, G.-B., Q.-Y. Zhu, and C.-K. Siew (2006). Extreme learning machine: theory and applications. *Neurocomputing 70*(1-3), 489–501.
28. Huré, C., H. Pham, and X. Warin (2020). Deep backward schemes for high-dimensional nonlinear PDEs. *Mathematics of Computation 89*(324), 1547–1579.
29. Hutchinson, J. M., A. W. Lo, and T. Poggio (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The journal of Finance 49*(3), 851–889.
30. Kennedy, M. C. and A. O'Hagan (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika 87*(1), 1–13.
31. Kingma, D. P. and J. Ba (2015). Adam: A method for stochastic optimization. *The 3rd International Conference on Learning Representations*.
32. Le Gratiet, L. (2013). *Multi-fidelity Gaussian process regression for computer experiments*. Ph. D. thesis, Université Paris-Diderot-Paris VII.
33. LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *Nature 521*(7553), 436–444.
34. Lehdili, N., P. Oswald, and H. Gueneau (2019). Market risk assessment of a trading book using statistical and machine learning. *Available at DOI: 10.13140/RG.2.2.23796.71047*.
35. Li, S., W. Xing, R. Kirby, and S. Zhe (2020). Multi-fidelity bayesian optimization via deep neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Eds.), *Advances in Neural Information Processing Systems*, Volume 33, pp. 8521–8531. Curran Associates, Inc.
36. Liu, S., C. W. Oosterlee, and S. M. Bohte (2019). Pricing options and computing implied volatilities using neural networks. *Risks 7*(1), 16.

37. Longstaff, F. A. and E. S. Schwartz (2001). Valuing American options by simulation: a simple least-squares approach. *The review of financial studies 14*(1), 113–147.
38. Ludkovski, M. (2018). Kriging metamodels and experimental design for Bermudan option pricing. *Journal of Computational Finance 22*(1), 37–77.
39. Meng, X. and G. E. Karniadakis (2020). A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *Journal of Computational Physics 401*, 109020.
40. Mu, G., T. Godina, A. Maffia, and Y. C. Sun (2018). Supervised machine learning with control variates for american option pricing. *Foundations of Computing and Decision Sciences 43*(3), 207–217.
41. Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. The MIT Press.
42. Nocedal, J. and S. J. Wright (1999). *Numerical optimization*. Springer.
43. Paleyes, A., M. Mahsereci, and N. D. Lawrence (2023). Emukit: A python toolkit for decision making under uncertainty. *Proceedings of Python in Science Conference 22*, 68–75.
44. Rasmussen, C. E. and C. K. Williams (2006). *Gaussian Processes for Machine learning*. The MIT Press.
45. Roncalli, T. (2020). *Handbook of Financial Risk Management*. Chapman and Hall/CRC.
46. Ruf, J. and W. Wang (2019). Neural networks for option pricing and hedging: A literature review. *Journal of Computational Finance 24*(1), 1–46.
47. Ruiz, I. and M. Zeron (2021). *Machine Learning for Risk Calculations: A Practitioner's View*. John Wiley & Sons.
48. Saunders, A., M. M. Cornett, and O. Erhemjamts (2021). *Financial institutions management: A risk management approach*. McGraw-Hill.
49. Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. *Proceedings of Machine Learning Research 5*, 567–574.
50. Zhang, H.-G., C.-W. Su, Y. Song, S. Qiu, R. Xiao, and F. Su (2017). Calculating value-at-risk for high-dimensional time series using a nonlinear random mapping model. *Economic Modelling 67*, 355–367.