Physics Informed Neural Networks in Computational Finance: High Dimensional Forward & Inverse Option Pricing

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Physics Informed Neural Networks In Computational Finance: High Dimensional Forward & Inverse Option Pricing

Master's Thesis

Ramy Tanios
rtanios@ethz.ch

Seminar for Applied Mathematics
Department of Mathematics
ETH Zürich

Supervisors:
Prof. Dr. Siddhartha Mishra
Roberto Molinaro

June 27, 2021

I would like to express my gratitude to my supervisor Prof. Dr. Siddhartha Mishra who guided me throughout this thesis, and I would like to show my appreciation for his continuous support and motivational words. I would also like to thank Roberto for his continuous and valuable feedback and for always being present for discussing results and sharing ideas and methodologies.

Abstract

Physics Informed Neural Networks have been recently widely used to successfully approximate solutions to partial differential equations. They also proved to be performant for estimating solutions and unknown model parameters from observed measurements. We therefore present the applications of such techniques for pricing multi-assets European options under the high dimensional Black-Scholes and Heston models. We then present a simple and straight-forward way for computing the Greeks that in turn play an important role in computational finance. Finally, we consider inferring the volatility function and pricing options from a set of observed prices. Numerical simulations as well as generalization errors are also presented.

Contents

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Partial differential equations (PDEs) model a variety of physical phenomena in different scientific fields such as physics, chemistry, astrophysics, fluid dynamics and finance, among others. While analytical solutions exist for a small subset of PDEs, unfortunately most of the PDEs in practice do not possess analytical solutions and hence require numerical methods for the efficient approximation of the underlying exact solutions. Among many, the most famous numerical methods are the finite difference [?], finite element [?] and finite volume methods [?]. Regardless of their wide use, these methods proved to be challenging to use in very high dimensional settings, and for inverse problems. For example, It can be shown that for parabolic linear PDEs and tensor product meshes, the finite element method discretization error at the final time is $\mathcal{O}(N^{-\frac{2}{d}})$ [?] where N is the number of mesh points and d is the space dimension. The latter estimate is a clear evidence to the curse of dimensionality that numerical methods suffer for from large d. For example for fixed N, the discretization error increases exponentially with the spatial dimension d.

Pricing financial products is a wide and important branch of computational finance. One usually resort to deterministic methods for pricing options by solving the Kolmogorov PDE that is based on the Feynman-Kac formula. In finance, the finite difference and finite element methods are widely used for option pricing. Though multi-assets option pricing is of huge interest, it entails working in high dimensional spaces, making it harder to use deterministic methods that suffer from the curse of dimensionality, as discussed previously. On the other hand, one could rely on the Feynman-Kac representation only and non-deterministric methods such as Monte-Carlo sampling [?]. However it turned out that Monte-Carlo sampling converges slowly to the exact solution, at a rate of $\frac{1}{2}$ and is computationally expensive. Hence, we need numerical methods that allow the fast computation of the solutions of high dimensional PDEs, compared to traditional methods.

Machine learning and deep learning have been widely used recently in different scientific applications such as text classification, sentimental analysis, image recognition, speech recognition, natural language processing and protein folding [?], among others. They turned out to be robust and performant techniques [?], being able to provide accurate solutions to different problems. More precisely, most of the latter applications rely on the use of Deep Neural Networks (DNNs), that represent a composition of layers consisting of affine and non-linear transformations. Since DNNs possess the *universal* *approximation property* [?, ?, ?], that is every continuous, even measurable function can be approximated by DNNs, it is intuitive and convenient to use DNNs as approximators of solutions to high dimensional PDEs. The resulting non-convex optimization problem can then be solved using different variants of the stochastic gradient descent [?]. Previous and recent work have been done in this context, for linear and non-linear hyperbolic, elliptic and parabolic PDEs in different fields such as computational fluid dynamics and finance [?, ?, ?, ?, ?, ?]. Two approaches can be followed: the supervised or the unsupervised learning, or both but this is not discussed in this project. In the first one, the exact solution or the Feynman-Kac representation of the exact solution of the PDE is used to generate data points and then a DNN is trained to learn the corresponding PDE solution [?] in a supervised fashion. Such approaches require the availability of the exact solution to produce the training dataset. In the second approach, we follow the technique in [?, ?] by collocating the PDE residuals at training points. The main advantage here is that data points of the solution are not needed and information is extracted from the physical model itself, hence the name *Physics Informed Neural Networks.* This method proves to be efficient and accurate when applied for example to scalar conversation laws, the heat equation, the Euler and Navier-Stokes equations [?, ?, ?]. In addition to that, in a recent paper of S. Mishra and R. Molinaro [?], they provide rigorous bounds on the generalization error of PINNS, which is the error on unseen data, in terms of the training error, number of training points and stability bounds of the underlying PDE. More precisely, they show that the generalization error will be sufficiently low as long as the training error is low, the number of training points is sufficiently large and the solution of underlying PDE is stable.

The goal of this paper is to illustrate the applications of PINNs for financial applications, for instance we consider the Black-Scholes [?] and Heston [?] models. More importantly, we solve in the first part the corresponding high dimensional (up to 100 dimensions) PDEs using PINNs, while traditional numerical methods were almost impossible in this case. Similar work is present in [?] where European and American options are considered, however for $d \leq 3$. In this paper we focus on European options but we consider $d \leq 100$.

In the second part of the paper, we discuss the computations of the Greeks, that are in turn very important quantities in computational finance, for instance in risk management. We show that computing them is a straightforward consequence of the forward problem and thus can be obtained easily using automatic differentiation [?], for any model and dimension in consideration. While analytical formulas for the Greeks exist only for very few models, low dimensions and some types of payoffs, other methods for computing the Greeks are present in the literature such as finite differences. The latter method is able in some cases to provide accurate results but still is affected by truncation and round-off errors. The main advantage here is that with few additional lines of code and with the use of automatic differentiation that it is based on the chain rule, we are able to compute exact derivatives efficiently and up to machine precision [?].

In the last part, we consider inverse problems where we estimate prices and unknown parameters from observed generated prices, which turned out to be efficient using PINNs

[?]. Such problems are called calibration problems and most of them are naturally ill-posed, hence they require regularization methods to ensure uniqueness. We make use of the Tikhonov reguralization that proved to be convenient in estimating the volatility function in some problems, while it was still not sufficient for others especially in high dimensional settings, where ill-posedness is more expressed. However, in all the test cases we were able to generate accurate prices that replicate the observed ones. The main advantage of PINNs over conventional methods such as the Least Squares approach [?] is that PINNs do not require the huge number of runs of the forward problem in order to estimate the unknown parameters. However the main advantage here is that inverse problems have the same cost and efficiency as the corresponding forward problems.

Chapter 2

Physics Informed Neural Networks

In this section, we present an abstract partial differential equation formulation. We also introduce *Physics Informed Neural Networks*. More precisely, we describe the architecture of PINNs and the corresponding forward and inverse problems. We also present the loss function and how the model (PDE) is used to learn the underlying exact solution. Two algorithms are presented, the forward and inverse problems' algorithms. At the end, we discuss *ensemble training*, a technique used for hyperparameters search.

2.1 The Underlying Partial Differential Equation

Let X and Y two normed Banach spaces equipped with the norms $||.||_X$ and $||.||_Y$ respectively. More precisely, let $\mathcal{D} \subset \mathbb{R}^{d'}$ a space-time domain where $d' \geq 1$ and let $Y = L^p(\mathcal{D}, \mathbb{R}^m)$ and $X = W^{s,q}(\mathcal{D}, \mathbb{R}^m)$ where $m \geq 1, 1 \leq p, q < \infty$ and $s \geq 0$. The space-time domain is written as $\mathcal{D} = (0, T) \times D$ where $D \subset \mathbb{R}^d$ is a bounded space domain, $d = d' - 1$. We focus next on scalar partial differential equations, i.e $m = 1$ and in its most general form, our equation is given by

$$\mathcal{L}u(t, \mathbf{x}) = f(t, \mathbf{x}), \quad (t, \mathbf{x}) \in \mathcal{D}, \tag{2.1}$$

where $\mathcal{L} : X \to Y$ is a differential operator, $f \in Y : \mathcal{D} \to \mathbb{R}$ the source term. Moreover, we assume that for all $f \in Y$, equation (**??**) has a unique solution $u \in X$. Note that boundary and initial conditions are already implicitly included in the formulation (**??**).

2.2 Physics Informed Neural Network

A Physics Informed Neural Network abbreviated by PINN is a multi-layer perceptron neural network consisting of neurons and input, hidden and output layers. In addition to that, the output of the network u_θ should satisfy (**??**) for it to be an admissible solution of the PDE. More precisely, given an input $\mathbf{y} = (t, \mathbf{x}) \in \mathcal{D}$, the network transforms $\mathbf{y}$ through a sequence of layers, each consisting of an affine transformation followed by a non-linear scalar transformation that we call an activation function σ, to an output denoted by $u_\theta(\mathbf{y})$ such that

$$u_\theta(\mathbf{y}) = L_K \circ \sigma \circ \cdots \circ \sigma \circ L_k \circ \cdots \circ \sigma \circ L_1(\mathbf{y}), \tag{2.2}$$

where $\theta \in \Theta$ is the set of weights of the network to be tuned. Let $K \in \mathbb{N}$ the total number of layers in the network. For $k \in \{1, \ldots, K\}$ and $v_k \in \mathbb{R}^{d_k}$, the affine function $L_k : \mathbb{R}^{d_k} \to \mathbb{R}^{d_{k+1}}$ is given by $L_k v_k = W_k v_k + b_k$ where $W_k \in \mathbb{R}^{d_{k+1} \times d_k}$ and $b_k \in \mathbb{R}^{d_k+1}$. Note that $d_1 = d'$ and $d_K = m = 1$. With that, we denote the tunable weights θ by

$$\theta = \{W_k, b_k\}_{k=1}^{K}. \tag{2.3}$$

For the activation functions σ, the most popular ones are the sigmoid activations, ReLU activations and the hyperbolic tangent (tanh) activations. In this paper, we focus on the hyperbolic tangent function as a choice for σ, as the function $x \mapsto \tanh x$ is twice differentiable which is consistent with the Black-Scholes and Heston differential operators.

2.2.1 Forward Problem & Optimization

Next, we discuss the forward problem and the corresponding optimization formulation. The main goal is to find a set of parameters θ such that the neural network u_θ approximates the solution u of the partial differential equation (**??**). To that extent, we follow the method of unsupervised learning, that is for $N \in \mathbb{N}$ we generate an unlabeled dataset $\mathcal{S} = \{\mathbf{y}_n\}_{n=1}^{N} \subset \mathcal{D}, \mathbf{y}_n \in \mathcal{D}$ for all n, and we consider minimizing the L^p-norm of the residual function of (**??**). More precisely, we define the residual function $(\mathbf{y} \mapsto \mathcal{R}_\theta(\mathbf{y})) \in L^p(\mathcal{D}, \mathbb{R})$ such that for all $\mathbf{y} \in \mathcal{S}$,

$$\mathcal{R}_\theta(\mathbf{y}) = \mathcal{L}u_\theta(\mathbf{y}) - f(\mathbf{y}). \tag{2.4}$$

Finally, our optimization problem is given by

$$\text{Find } \theta^* \in \Theta : \quad \theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \, ||\mathcal{R}_\theta||_{L^p}^p = \underset{\theta \in \Theta}{\operatorname{argmin}} \int_{\mathcal{S}} |\mathcal{R}_\theta(\mathbf{y})|^p d\mathbf{y}. \tag{2.5}$$

Figure **??** below illustrates Physics Informed Neural Networks. One can see in the dashed box the multi-layer perceptron and how the PDE model is imposed to the network u_θ in order to calculate the residual $\mathcal{R}_\theta$.
Clearly, one is not able to evaluate the integral in (**??**) analytically and so we resort to numerical quadratures to approximate it. Possible methods are Monte-Carlo methods [**?**] based on sampling uniformly, independent and identically distributed random points in the space $\mathcal{D}$. We resort to the latter method for $d \geq 20$. However, for $d < 20$ we make use of low discrepancy sequences such as the Sobol sequences as quadrature points [**?**]. Finally, we denote the approximation of the integral in (**??**) by $J(\theta)$ as the loss function and we write it as

$$J(\theta) = \sum_{n=1}^{N} \zeta_n |\mathcal{R}_\theta(\mathbf{y}_n)|^p = \sum_{n=1}^{N} \zeta_n |\mathcal{L}u_\theta(\mathbf{y}_n) - f(\mathbf{y}_n)|^p, \quad \mathbf{y}_n \in \mathcal{S}, \tag{2.6}$$

for some quadrature weights $\{\zeta_n\}_{n=1}^{N}$. It is a well known practice in standard machine learning to regularize the optimziation problem, that is one can introduce a penalty term

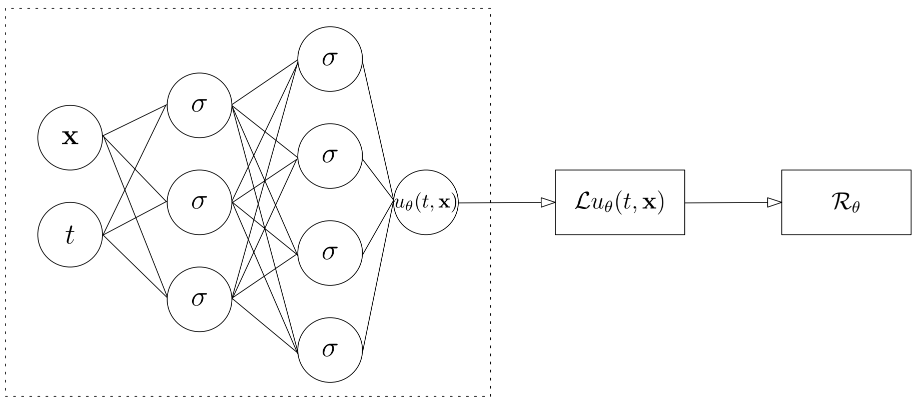


Figure 2.1: Physics Informed Neural Network.

to the loss function, depending on the norm of the weights with respect to we optimize. More precisely the optimization problem (**??**) can be modified as

$$\text{Find } \theta^* \in \Theta : \quad \theta^* = \underset{\theta\in\Theta}{\operatorname{argmin}}\, J(\theta) + \lambda_{L^q}||\theta||_q^q, \tag{2.7}$$

where $0 \ll \lambda_{L^q} \ll 1$ is the intensity of the penalization. Increasing the value of λ_{L^q} makes the weights decay faster in contrast to small values of λ_{L^q}. The integer $q \in \{1, 2\}$ indicates which norm we use to evaluate the penalization of the weights. For $q = 1$, we induce sparsity to the set of paramaters θ (LASSO regularization), that is some of the weights are set exactly to 0 while on the other hand for $q = 2$ a weight decay towards 0 occurs but does not induce sparsity (Ridge regularization).

We expand the optimization problem (**??**) to show the loss in the initial and boundary conditions. Let $N_B, N_I \in \mathbb{N}$ be the number of interior points in Ω, the number of spatial boundary points on $\partial\Omega \times (0, T)$ and the number of temporal boundary points in $\Omega \times \{0\}$, respectively. In addition to the previously defined set $\mathcal{S}$, we define the sets

$$\mathcal{S}_B = \{\mathbf{y}_i \in (0, T) \times \partial D, 1 \leq i \leq N_B\} \text{ and } \mathcal{S}_I = \{\mathbf{y}_i \in \{0\} \times D, 1 \leq i \leq N_I\}. \tag{2.8}$$

We define also two residuals $\mathcal{R}^B_\theta$ and $\mathcal{R}^I_\theta$ such that

$$\begin{aligned} \forall \mathbf{y} = (t, \mathbf{x}) \in \mathcal{S}_B, \quad \mathcal{R}^B_\theta(\mathbf{y}) &= u_\theta(\mathbf{y}) - u(t, \mathbf{x}), \\ \forall \mathbf{y} = (0, \mathbf{x}) \in \mathcal{S}_I, \quad \mathcal{R}^I_\theta(\mathbf{y}) &= u_\theta(\mathbf{y}) - \psi(\mathbf{x}). \end{aligned}$$

where $\psi(\mathbf{x})$ is a given initial condition. In the sequel, we choose $p = 2$ and the optimization problem reads

$$\begin{aligned}\text{Find } \theta^* \in \Theta : \quad \theta^* &= \underset{\theta\in\Theta}{\operatorname{argmin}} \, \lambda_{res}||\mathcal{R}_\theta||_{L^2}^2 + ||\mathcal{R}_\theta^B||_{L^2}^2 + ||\mathcal{R}_\theta^I||_{L^2}^2 + \lambda_{L^q}||\theta||_q^q, \\ &= \underset{\theta\in\Theta}{\operatorname{argmin}} \, \lambda_{res} \int_{\mathcal{S}} |\mathcal{R}_\theta(\mathbf{y})|^2 d\mathbf{y} + \int_{\mathcal{S}_B} |\mathcal{R}_\theta^B(\mathbf{y})|^2 d\mathbf{y} + \int_{\mathcal{S}_I} |\mathcal{R}_\theta^I(\mathbf{y})|^2 d\mathbf{y} \\ &+ \lambda_{L^q}||\theta||_q^q, \end{aligned} \tag{2.9}$$

where λ_{res} is a hyperparameter balancing the residuals in the PDE against the initial and boundary residuals, and λ_{L^q} a regularization parameter. Finally, the integrals are approximated again numerically using either Sobol sequences or Monte-Carlo sampling depending on the dimension d and the final optimization problems formulates as

$$\text{Find } \theta^* \in \Theta : \quad \theta^* = \underset{\theta\in\Theta}{\operatorname{argmin}} \, J(\theta) + \lambda_{L^q}||\theta||_q^q, \tag{2.10}$$

where the loss $J(\theta)$ is given by

$$J(\theta) = \lambda_{res} \sum_{n=1}^{N} \zeta_n |\mathcal{R}_\theta(\mathbf{y}_n)|^2 + \sum_{n=1}^{N_B} \zeta_n^B |\mathcal{R}_\theta^B(\mathbf{y}_n)|^2 + \sum_{n=1}^{N_I} \zeta_n^I |\mathcal{R}_\theta^I(\mathbf{y}_n)|^2, \tag{2.11}$$

for some weights $\{\zeta_n\}_{n=1}^N, \{\zeta_n^B\}_{n=1}^{N_B}, \{\zeta_n^I\}_{n=1}^{N_I}$ known as the quadrature weights. The following algorithm (**??**) summarizes the major steps that need to be done to solve the forward problem, that is approximating the solution u to the model (**??**).

Algorithm 1: The Forward Problem

```
Input: Domain $\mathcal{D}$, training dataset $(t_i, \mathbf{x}_i)_i \in \mathcal{S} \cup \mathcal{S}_I \cup \mathcal{S}_B \subset \mathcal{D}$ defining the quadrature
  points in (??), number of epochs $N_{epochs}$, number of batches $N_{batches}$, learning rate $\eta$
  and a differential operator $\mathcal{L}$.
Output: $u_\theta$ approximating the solution $u$ to (??).
  Initialize the weights $\theta$ of the network.
  for $i = 1$ to $N_{epochs}$ do
    for $j = 1$ to $N_{batches}$ do
      Forward pass the batch $\mathcal{B}_j$ through the network to get $u_\theta$.
      Compute the residual $\mathcal{L}(u_\theta) - f$ using automatic differentation.
      Compute the loss $L(\theta) := J(\theta) + \lambda_{L^q}||\theta||_q^q$ according to (??) and (??).
      Perform a backward propagation to compute $\nabla_\theta L(\theta)$.
      Update the weights according to $\theta \leftarrow \theta - \eta\nabla_\theta L(\theta)$.
    end for
  end for
```

2.2.2 Inverse Problem & Optimization

In this section, we discuss the applications of PINNs for inverse problems and the corresponding optimization formulation. We consider the model (**??**) with the unknown solution u, and we additionally assume that there is an unknown parameter function π to be estimated in our model. For example, π could be the thermal conductivity in the heat equation, or the volatility parameters in the Black-Scholes model. In our discussion we consider the general case that π depends on the space and time dimensions and we write $\pi(t, \mathbf{x})$ for $(t, \mathbf{x}) \in \mathcal{D}$. Our new model is then given by

$$\mathcal{L}(u(t, \mathbf{x}), \pi(t, \mathbf{x})) = f, \quad (t, \mathbf{x}) \in \mathcal{D}. \tag{2.12}$$

Finally, we also assume that we are given some noiseless measurements of the solution u in the domain $\mathcal{D}^* \subset \mathcal{D}$ and we denote them by $u^*(t, \mathbf{x})$ for $\mathbf{y} = (t, \mathbf{x}) \in \mathcal{D}^*$.

What we seek then are two functions $u : \mathcal{D} \to \mathbb{R}$ and $\pi : \mathcal{D} \to \mathbb{R}$ such that u is consistent with the observed values u^* in the domain $\mathcal{D}^*$ and such that u and π satisfy the model (**??**). The optimization problem for the inverse model is similar to the one for forward problem except for the addition of the loss term corresponding to the given noiseless measurements. In addition to that, inverse problems are usually ill-posed and so adding a Tikhonov regularization proves to be necessary to ensure well-posedness [**?**, **?**, **?**].

In more detail, let N_M be the number of given measurements of u in $\mathcal{D}^*$, let $\mathcal{S}_M = \{\mathbf{y}_i \in \mathcal{D}^*, 1 \leq i \leq N_M\}$ and let $\lambda_\Phi > 0$ the Tikhonov regularization parameter. The optimization problem reads

$$\text{Find } \theta^* \in \Theta : \quad \theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}}\, J(\theta) + \lambda_{L^q} ||\theta||_q^q + \lambda_\Phi \Phi(\pi_\theta), \tag{2.13}$$

where π_θ is the PINN approximation of π and Φ a Tikhonov functional. The modified $J(\theta)$ is given by

$$J(\theta) = \lambda_{res} \sum_{n=1}^{N} \zeta_n |\mathcal{R}_\theta(\mathbf{y}_n)|^2 + \sum_{n=1}^{N_I} \zeta_n^I |\mathcal{R}_\theta^I(\mathbf{y}_n)|^2 + \frac{1}{N_M} \sum_{n=1}^{N_M} |u_\theta(\mathbf{y}_n) - u^*(\mathbf{y}_n)|^2. \tag{2.14}$$

The loss in equation (**??**) differs from the loss in equation (**??**) for the forward problem in the sense that boundary conditions here are not imposed, however the mean squared loss for the given measurements is added. In addition to that, the Tikhonov functional Φ can be defined as the H_1-seminorm of π_θ [**?**], i.e

$$\Phi(\pi_\theta) := |\pi_\theta|_{H^1}^2 = ||\nabla \pi_\theta||_{L^2}^2 = \int_{\mathcal{D}} ||\nabla \pi_\theta(t, \mathbf{x})||_2^2 d\mathbf{x} dt. \tag{2.15}$$

However, in general it depends on the problem in consideration and will be discussed in more detail later in the report. Algorithm (**??**) describes the solution approach of the inverse problem.

Algorithm 2: The Inverse Problem

```
Input: Domain $\mathcal{D}$, training dataset $(t_i, \mathbf{x}_i)_i \in \mathcal{S} \cup \mathcal{S}_I \cup \mathcal{S}_B \subset \mathcal{D}$ defining the quadrature
  points in (??), $((t_i, \mathbf{x}_i), u_i^*)_i \in \mathcal{S}_M \times \mathbb{R}$ noiseless measurements of the solution $u$ in
  some domain $\mathcal{D}^* \subset \mathcal{D}$, number of epochs $N_{epochs}$, number of batches $N_{batches}$,
  learning rate $\eta$ and a differential operator $\mathcal{L}$.
Output: $u_\theta$ approximating the solution $u$ to (??) and $\pi_\theta$ approximating the unknown
  parameter $\pi$.
  Initialize the weights $\theta$ of the network.
  for $i = 1$ to $N_{epochs}$ do
    for $j = 1$ to $N_{batches}$ do
      Forward pass the batch $\mathcal{B}_j$ through the network to get $u_\theta$.
      Compute the residual $\mathcal{L}(u_\theta) - f$ using automatic differentation.
      Compute the loss $L(\theta) := J(\theta) + \lambda_{L^q} ||\theta||_q^q + \lambda_\Phi \Phi(\pi_\theta)$ according to (??) and (??).
      Perform a backward propagation to compute $\nabla_\theta L(\theta)$.
      Update the weights according to $\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$.
    end for
  end for
```

2.3 Ensemble Training

We denote by ε_{T} the training error, given by the loss in equation (**??**) or (**??**), and ϵ_{T} the normalized training error, that is

$$\epsilon_{\mathrm{T}} := \frac{\varepsilon_{\mathrm{T}}}{||u_\theta(T,.)||_{L^2(D)}} = \frac{\varepsilon_{\mathrm{T}}}{(\int_D |u_\theta(T,\mathbf{x})|^2 d\mathbf{x})^{\frac{1}{2}}}, \tag{2.16}$$

where u_θ is the PINN approximation of the the exact solution u of the PDE (**??**). We also denote by ϵ_{G} the L_2 relative generalization error at the final time T, that is the error on unseen data, defined by

$$\begin{aligned}\epsilon_{\mathrm{G}} &:= \frac{||u_\theta(T,.) - u(T,.)||_{L^2(D)}}{||u(T,.)||_{L^2(D)}} \\ &= \frac{(\int_D |u_\theta(T,\mathbf{x}) - u(T,\mathbf{x})|^2 d\mathbf{x})^{\frac{1}{2}}}{(\int_D |u(T,\mathbf{x})|^2 d\mathbf{x})^{\frac{1}{2}}}. \end{aligned} \tag{2.17}$$

Integrals in equations (**??**) and (**??**) are approximated again by means of quadrature rules. More precisely, we sample N uniformly random points in the domain D and the integrals are then replaced by weighted sums and the errors are approximated as

$$\epsilon_{\mathrm{T}} \approx \frac{\varepsilon_{\mathrm{T}}}{(\frac{1}{N}\sum_{i=1}^{N} |u_\theta(T,\mathbf{x}_i)|^2)^{\frac{1}{2}}}, \tag{2.18}$$

$$\epsilon_{\mathrm{G}} \approx \frac{(\sum_{i=1}^{N} |u_\theta(T,\mathbf{x}_i) - u(T,\mathbf{x}_i)|^2)^{\frac{1}{2}}}{(\sum_{i=1}^{N} |u(T,\mathbf{x}_i)|^2)^{\frac{1}{2}}}. \tag{2.19}$$

Choosing a set of hyperparameters for a given problem that leads to a low generalization error ϵ_{G} is obviously not a trivial decision. As a result, a suitable method for this choice is mandatory to adapt. We consider the ensemble training technique presented in [**?**]. More precisely, table (**??**) presentes the hyperparameters that needs to be tuned for the training process. Each combination of parameters is regarded as one realization from the total parameter space $\mathcal{H} = \times_i \mathcal{H}_i$, with size $|\mathcal{H}| = \prod_i |\mathcal{H}_i|$, where $\mathcal{H}_i$ is a space of the i-th parameter in table (**??**). For example, $\mathcal{H}_2 = \{20, 24, 28\}$ is the space of number of neurons, namely 20, 24 or 28 that we choose for training PINNs. For each choice of hyperparameters, algorithm (**??**) (forward problem) or (**??**) (inverse problem) is then initiated.

A key hyperparameter in the training process is θ_0, the initial weights of the neural network. Since the loss function is non-convex, we can ensure that the optimizer will converge to a local minima. As a result, the choice of the initial weights plays a crucial role in the training process, as different weights initializations would converge to different local minimas, and hence will lead to different generalization errors and performances. A widely used practice in machine learning is to retrain models. More precisely, we retrain each model (for instance, 10 or 15 retrainings) with different starting initial

1	$K-1$	Number of hidden layers
2	d_k	Number of neurons in the k-th layer
3	q	Exponent in equation (??)
4	λ_{res}	Residual parameter in the loss (??)
5	λ_{L^q}	Regulization parameter in the loss (??)
6	λ_{Φ}	Tikhonov parameter in the loss (??)
7	θ_0	Initial weights of the neural network

Table 2.1: Hyperparameters

weights in order to select one of the trained networks. Ideally, one would choose the best configuration based on the lowest generalization error ϵ_{G}. Since this is not possible in practice as the exact solution of the problem is not available at hand, we follow in this project the approach of choosing the best setup based on the lowest training error ϵ_{T} which can be computed during training and therefore is available at hand. This is motived by the error bounds derived in [?].

In order to keep the ensemble training process tractable, that is $|\mathcal{H}|$ as small as possible, we choose parameter spaces according to table (??).

Hyperparameter	Forward Problem	Inverse Problem
$K-1$	$\{4, 8\}$	
d_k	$\{16, 20, 24, 28, 32\}$	$\{16, 20, 24, 28, 32, 36, 40\}$
q	$\{1, 2\}$	
λ_{res}	$\{0.1, 1, 10\}$	
λ_{L^q}	$\{0, 10^{-6}, 10^{-5}\}$	
λ_{Φ}	**Not applicable**	$\{0, 10^{-8}, 10^{-6}, 10^{-4}\}$

Table 2.2: Hyperparameters' spaces.

To sum up the *ensemble training* technique, for a given realization of the hyperparameters space $\mathcal{H}$ we choose between the different retrainings the setup that leads to the lowest training error ϵ_{T}. Among all the hyperparameters combination, again the lowest training error is the criteria for the choice of the best neural network architecture.

Chapter 3

Financial Models

3.1 Feynman-Kac Formula

Let $T \in (0, \infty)$ and $d \in \mathbb{N}$, let $(\Omega, \mathcal{F}, \mathbb{P})$ a probability space with a normal filtration $\mathbb{F} = (\mathbb{F}_t)_{t\in[0,T]}$ and let $\mathbf{W} = (W^1, W^2, \ldots, W^d) : [0,T] \times \Omega \to \mathbb{R}^d$ a standard $(\Omega, \mathcal{F}, \mathbb{P}, (\mathbb{F}_t)_{t\in[0,T]})$-Brownian motion. Let $\mathbf{X} = (X^1, \ldots, X^d) : [0,T] \times \Omega \to \mathbb{R}^d$ an $\mathbb{F}$-adapted stochastic process with continuous sample paths satisfying for every $t \in [0,T]$

$$\mathbf{X}_t = \mathbf{X}_0 + \int_0^t \mu(\mathbf{X}_s)ds + \int_0^t \sigma(\mathbf{X}_s)d\mathbf{W}_s, \quad \mathbb{P} - a.s, \tag{3.1}$$

where $\mu : \mathbb{R}^d \to \mathbb{R}^d$ and $\sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ are Lipschitz continuous functions. Let $\psi : \mathbb{R}^d \to \mathbb{R}$ a function and let $v = (v(t,\mathbf{x}))_{t\in[0,T]\times\mathbb{R}^d} \in C^{1,2}([0,T]\times\mathbb{R}^d, \mathbb{R})$ such that $v(T,\mathbf{x}) = \psi(\mathbf{x})$ and it holds for all $t \in [0,T)$ and $\mathbf{x} \in \mathbb{R}^d$

$$v(t,\mathbf{x}) = \mathbb{E}[v(T,\mathbf{X}_T)|\mathbf{X}_t = \mathbf{x}] = \mathbb{E}[\psi(\mathbf{X}_T)|\mathbf{X}_t = \mathbf{x}]. \tag{3.2}$$

Equation (**??**) is motivated by the Martingale Option Pricing theory [**?**]. Introducing the new variable $t \leftarrow T - t$, the function $u(t,\mathbf{x}) = v(T-t,\mathbf{x})$ satisfies $u(0,\mathbf{x}) = \psi(\mathbf{x})$ and for all $t \in [0,T]$ and $\mathbf{x} \in \mathbb{R}^d$

$$(\partial_t u)(t,\mathbf{x}) = \frac{1}{2}\mathrm{Tr}(\sigma(x)\sigma(x)^\top(\mathrm{Hess}_{\mathbf{x}}u)(t,\mathbf{x})) + \langle\mu(\mathbf{x}), (\nabla_{\mathbf{x}}u)(t,\mathbf{x})\rangle. \tag{3.3}$$

One can also make use of the modified and widely used in finance version of the Feynman-Kac formula. If equation (**??**) is modified to accomodate the risk-free interest rate function $r : \mathbb{R}^d \to \mathbb{R}$, i.e for all $t \in [0,T]$ and $\mathbf{x} \in \mathbb{R}^d$

$$u(t,\mathbf{x}) = \mathbb{E}[e^{\int_0^t r(\mathbf{X}_s)ds}u(0,\mathbf{X}_t)|\mathbf{X}_0 = \mathbf{x}] = \mathbb{E}[e^{\int_0^t r(\mathbf{X}_s)ds}\psi(\mathbf{X}_t)|\mathbf{X}_0 = \mathbf{x}]. \tag{3.4}$$

The corresponding partial differential equation is given by

$$(\partial_t u)(t,\mathbf{x}) = \frac{1}{2}\mathrm{Tr}(\sigma(x)\sigma(x)^\top(\mathrm{Hess}_{\mathbf{x}}u)(t,\mathbf{x})) + \langle\mu(\mathbf{x}), (\nabla_{\mathbf{x}}u)(t,\mathbf{x})\rangle - ru(t,\mathbf{x}), \tag{3.5}$$

with initial condition $u(0,\mathbf{x}) = \psi(\mathbf{x})$.

3.2 Price Models

In this section we discuss the financial models used in this project. However we start by some useful definitions:

- **Leverage Effect:** The negative correlation between the stock return and the volatility is termed as the Leverage effect. It is observed in practice and it means that large drops in the stock price lead to an increase of the volatility, while when the stock price increases the volatility decreases.
- **Clustering Effect:** Large market moves are followed by large ones, and small market moves are followed by smalls ones. That is illustrated in figure (**??**) where the x-axis represents the date and the y-axis represents the log-returns.
- **Implied Volatility:** The volatility that should be inserted in the Black-Scholes model in order to replicate market options' prices.

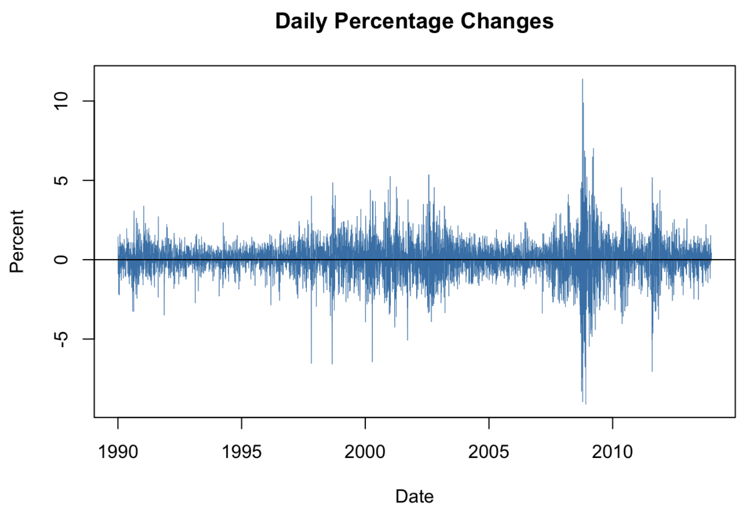


Figure 3.1: The Clustering effect of the volatility.

In the following sections, we start by considering the Black-Scholes model. It is the simplest and widely used model in computational finance, due to the fact that it assumes a constant return on the stock and constant volatility. In the multi-dimensional case, we differentiate between two cases: uncorrelated and correlated noise. In the uncorrelated noise, we assume that stock returns are uncorrelated, so that the evolution of each stoch price process is independent of the others. A more realistic assumption is to consider a correlation between the stocks since in reality different stocks can be positively or negatively correlated, and that is indeed the correlated noise case. For example, prices of Facebook shares are observed to be positively correlated with the S%P 500 market.

Even so the Black-Scholes model is widely used in practice due to its simplicity and to the existence of analytical solutions, it also suffers from some limitations. The most important limitations of the Black-Scholes model are listed and discuss below:

- **Constant volatility:** The volatility of each stock is deterministic and constant while in real life this is not really the case. In practice, the implied volatility exhibits a volatility smile as a function of the strike price (see figure (**??**)), while clearly the Black-Scholes model assumes a constant implied volatility. In addition to that, a constant volatility will not be able to capture the Leverage and Clustering effects that are observed in pratice.

- **Log-normal returns:** The Black-Scholes model assumes that the returns follow a log-normal distribution. Thus, pricing options that are out-of-money through the Black-Scholes model yields large errors because such extreme price changes are observed in practice but rarely happen under the Black-Scholes model. As a result, the Black-Scholes model is not able to capture extreme events(market crashes) that are observed in practice. Returns' distribution is usually asymmetric, exhibits fat tails compared to the log-normal distribution.

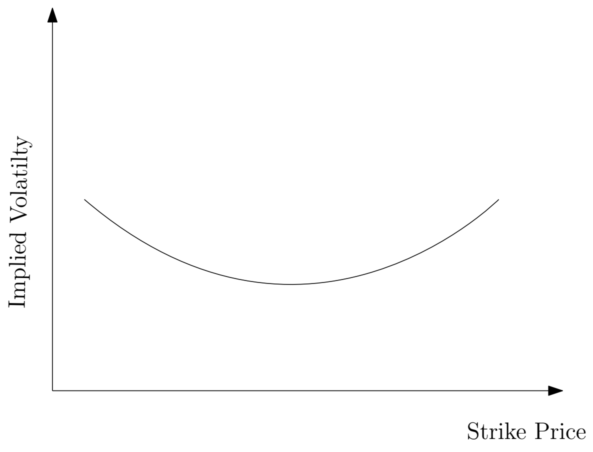


Figure 3.2: Volatility smile.

In order to improve the Black-Scholes model, we consider next the Heston model. In this case, the problem of constant volatility seen in the Black-Scholes model is alleviated by considering a stochastic volatility. More precisely, the volatility of the stock return in this case is neither constant nor a deterministic function but modeled as a stochastic process itself, that is correlated with the stock price process. Choosing the latter correlation to be negative clearly models the Leverage effect. In addition to that, the implied volatility obtained from prices calculated through the Heston model indeed exhibits a volatility smile. Finally, another important feature of the Heston model is that the volatility process follows a Cox-Ingersoll-Ross (CIR) process, that is it is a mean-reverting process.

This means that the volatility is always attracted to its long-term mean. If that would not be the case, markets would contain assets with exploding or close to zero volatilies, and these happen rarely in practice [?]. In contrast to the Black-Scholes model, the implementation of the Heston model is more complicated and calculating option prices then are more computationally expensive.

3.3 Black-Scholes Model With Uncorrelated Noise

We consider in this section the multi-dimensional Black-Scholes model with uncorrelated Brownian motion parts. Hence, the matrix σ in (**??**) is a diagonal matrix reflecting the independencies between the Brownian motion parts. In this case, the stochastic process $\mathbf{X}$ satisfies that for every $t \in [0,T]$

$$d\mathbf{X}_t = \mu \mathbf{X}_t dt + \operatorname{diag}(\sigma_1 X_t^1, \ldots, \sigma_d X_t^d) d\mathbf{W}_t, \quad \mathbf{X}_0 = \mathbf{x}_0, \quad \mathbb{P} - a.s. \tag{3.6}$$

where $\mu \in \mathbb{R}$ is the stock return and $\sigma_1, \ldots, \sigma_d$ are positive stocks volatilities. The solution of SDE (**??**) is given by

$$\mathbf{X}_t = [x_0^i \exp\big((\mu - \frac{1}{2}\sigma_i^2)t + \sigma_i W_t^i\big)]_{i=1}^d. \tag{3.7}$$

Indeed for every $i \in \{1, \ldots, d\}$, let $\Psi : [0,T] \times \mathbb{R} \to \mathbb{R}$ a smooth function defined by $\Psi(t,w) = x_0^i \exp\big((\mu - \frac{1}{2}\sigma_i^2)t + \sigma_i w\big)$. Using Itö's formula

$$\begin{aligned} dX_t^i = d\Psi(t, W_t^i) &= (\partial_t \Psi)(t, W_t^i)dt + (\partial_w \Psi)(t, W_t^i)dW_t^i + \frac{1}{2}(\partial_{ww}^2 \Psi)(t, W_t^i)d[W^i]_t \\ &= (\mu - \frac{1}{2}\sigma_i^2)\Psi(t, W_t^i)dt + \sigma_i \Psi(t, W_t^i)dW_t^i + \frac{1}{2}\sigma_i^2 \Psi(t, W_t^i)dt \\ &= \mu X_t^i dt + \sigma_i X_t^i dW_t^i, \end{aligned}$$

and that finishes the proof realizing that the latter equation is the i-th component of the (**??**). Let $\psi : \mathbb{R}^d \to \mathbb{R}$ a payoff function that can be thought of as an initial condition. The price of an option with payoff ψ and underlying d stocks is given by

$$u(t, \mathbf{x}) = \mathbb{E}[\psi(\mathbf{X}_t) | \mathbf{X}_0 = \mathbf{x}] = \mathbb{E}[\psi([x^i \exp\big((\mu - \frac{1}{2}\sigma_i^2)t + \sigma_i W_t^i\big)]_{i=1}^d)]. \tag{3.8}$$

The Feynman-Kac theorem states that the function u satisfies for every $t \in [0,T], \mathbf{x} \in \mathbb{R}^d, u(0, \mathbf{x}) = \psi(\mathbf{x})$ and

$$(\partial_t u)(t, \mathbf{x}) = \frac{1}{2} \sum_{i=1}^d |\sigma_i x_i|^2 (\partial_{x_i x_i}^2 u)(t, \mathbf{x}) + \sum_{i=1}^d \mu x_i (\partial_{x_i} u)(t, \mathbf{x}). \tag{3.9}$$

The RHS in (**??**) is degenerate. To over come this problem, we introduce the process $Y^i = \log\big(X^i\big)$ for every $i \in \{1, \dots, d\}$. Using Itö's formula, we can write

$$dY^i_t = (\mu - \frac{1}{2}\sigma_i^2)dt + \sigma_i dW^i_t, \quad Y^i_0 = \log\big(x^i_0\big), \quad \mathbb{P} - a.s. \tag{3.10}$$

In vector form, the process $\mathbf{Y}$ follows

$$d\mathbf{Y}_t = [\mu - \frac{1}{2}\sigma_i^2]_{i=1}^d dt + \operatorname{diag}(\sigma_1, \dots, \sigma_d) d\mathbf{W}_t, \quad \mathbf{Y}_0 = \log(\mathbf{x}_0), \quad \mathbb{P} - a.s. \tag{3.11}$$

and the solution is

$$\mathbf{Y}_t = [Y^i_0 + (\mu - \frac{1}{2}\sigma_i^2)t + \sigma_i dW^i_t]_{i=1}^d. \tag{3.12}$$

Let $v(t, \log \mathbf{x}) := u(t, \mathbf{x})$. We then find that $v(t, \mathbf{y}) = u(t, \exp(\mathbf{y})) = \mathbb{E}[\psi(\exp(\mathbf{Y}_t))| \exp(\mathbf{Y}_0) = \exp(\mathbf{y})] = \mathbb{E}[\tilde{\psi}(\mathbf{Y}_t)|\mathbf{Y}_0 = \mathbf{y}] = \mathbb{E}[\tilde{\psi}([y^i + (\mu - \frac{1}{2}\sigma_i^t)t + \sigma_i dW^i_t]_{i=1}^d)]$ satisfies for every $t \in [0, T], \mathbf{y} \in \mathbb{R}^d, v(0, \mathbf{y}) = u(0, \exp(\mathbf{y})) = \psi(\exp(\mathbf{y})) = \tilde{\psi}(\mathbf{y})$ and

$$(\partial_t v)(t, \mathbf{y}) = \frac{1}{2}\sum_{i=1}^d \sigma_i^2 (\partial^2_{y_i y_i} v)(t, \mathbf{y}) + \sum_{i=1}^d (\mu - \frac{1}{2}\sigma_i^2)(\partial_{y_i} v)(t, \mathbf{y}). \tag{3.13}$$

For training PINNs, one can observe that the transformed model (**??**) has a simpler PDE in the sense that the space dimension $\mathbf{x}$(or the corresponding transformed variable $\mathbf{y}$) is not present explicitly in its formulation, whereas it is present in the non-transformed model (**??**). On the other hand, the solution(and initial conditions) of the non-transformed model are easier to learn, compared to the transformed model. In this paper whe choose to train the non-transformed model, and hence the complete model setup is given by

$$\begin{cases} (\partial_t u)(t, \mathbf{x}) & = \frac{1}{2}\sum_{i=1}^d |\sigma_i x_i|^2 (\partial^2_{x_i x_i} u)(t, \mathbf{x}) + \sum_{i=1}^d \mu x_i (\partial_{x_i} u)(t, \mathbf{x}), \quad t \in (0, T), \mathbf{x} \in D, \\ u(t, \mathbf{x}) & = \frac{1}{N_s}\sum_{n=1}^{N_s} \psi([x^i \exp\Big((\mu - \frac{1}{2}\sigma_i^t)t + \sigma_i dW^{i,n}_t\Big)]_{i=1}^d), \quad t \in (0, T), \mathbf{x} \in \partial D, \\ u(0, \mathbf{x}) & = \psi(\mathbf{x}), \quad \mathbf{x} \in D. \end{cases}$$

Following [**?**], we make use of the Feynman-Kac solution to impose Dirichlet boundary conditions, where the expectation is replaced by a sample mean of size $N_s \in \mathbb{N}$ and $W^{i,n}$ is the n-th sample of the Brownian motion corresponding to the i-th dimension. In addition to that, the differential operator $\mathcal{L}$ in (**??**) is given for all $u \in C^{1,2}([0, T] \times \mathbb{R}^d, \mathbb{R})$ by

$$\mathcal{L}u(t, \mathbf{x}) = (\partial_t u)(t, \mathbf{x}) - \frac{1}{2}\sum_{i=1}^d |\sigma_i x_i|^2 (\partial^2_{x_i x_i} u)(t, \mathbf{x}) - \sum_{i=1}^d \mu x_i (\partial_{x_i} u)(t, \mathbf{x}). \tag{3.14}$$

3.4 Black-Scholes Model With Correlated Noise

In this section, we modify the previously defined Black-Scholes model and we account for correlated Brownian motion parts. More precisely, let $\boldsymbol{\Sigma} = (\Sigma_{i,j})_{i,j=1}^d \in \mathbb{R}^{d\times d}$ and let $\boldsymbol{\zeta}_k = (\Sigma_{k,1}, \dots, \Sigma_{k,d}) \in \mathbb{R}^d$, for $k \in \{1, \dots, d\}$. Consider the stochastic model

$$d\mathbf{X}_t = \mu(\mathbf{X}_t)dt + \sigma(\mathbf{X}_t)d\mathbf{W}_t, \quad \mathbf{X}_0 = \mathbf{x}_0, \quad \mathbb{P} - a.s., \tag{3.15}$$

where $\mu(\mathbf{X}_t) = [\mu X_t^i]_{i=1}^d \in \mathbb{R}^d$, $\sigma(\mathbf{X}_t) = [X_t^i \beta_i \boldsymbol{\zeta}_i]_{i=1}^d \in \mathbb{R}^{d\times d}$ and $\beta_1, \dots, \beta_d$ are positive stock volatilities. In this case, the matrix σ is no longer diagonal as it accounts in its off-diagonals for the correlations between the different Brownian motions. For every $i \in \{1, \dots, d\}$, the model (**??**) is defined as

$$dX_t^i = \mu X_t^i dt + \beta_i X_t^i \langle \boldsymbol{\zeta_i}, d\mathbf{W}_t \rangle_{\mathbb{R}^d}, \quad X_i^i = x_0^i, \quad \mathbb{P} - a.s. \tag{3.16}$$

and its solution is given as

$$X_t^i = x_0^i \exp\big((\mu - \frac{1}{2}||\beta_i \boldsymbol{\zeta}_i||_{\mathbb{R}^d}^2)t + \langle \beta_i \boldsymbol{\zeta}_i, \mathbf{W}_t \rangle_{\mathbb{R}^d}\big). \tag{3.17}$$

Indeed, let $\Psi : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ a smooth function such that for every $t \in [0,T]$ and $\boldsymbol{w} \in \mathbb{R}^d$, $\Psi(t, \boldsymbol{w}) = x_0^i \exp\big((\mu - \frac{1}{2}||\beta_i \boldsymbol{\zeta}_i||_{\mathbb{R}^d}^2)t + \langle \beta_i \boldsymbol{\zeta}_i, \boldsymbol{w} \rangle_{\mathbb{R}^d}\big)$. Using Itö's formula

$$\begin{aligned} dX_t^i = d\Psi(t, \mathbf{W}_t) &= (\partial_t \Psi)(t, \mathbf{W}_t)dt + \sum_{l=1}^d (\partial_{w_l} \Psi)(t, \mathbf{W}_t) dW_t^i + \frac{1}{2} \sum_{l,k=1}^d (\partial^2_{w_l w_k} \Psi)(t, \mathbf{W}_t) d[W^l, W^k]_t \\ &= X_t^i(\mu - \frac{1}{2}||\beta_i \boldsymbol{\zeta}_i||_{\mathbb{R}^d}^2)dt + \beta_i X_t^i \sum_{l=1}^d \zeta_i^{(l)} dW_t^i + \frac{1}{2}\beta_i^2 X_t^i \sum_{l=1}^d (\zeta_i^{(l)})^2 dt \\ &= \mu X_t^i dt + \beta_i X_t^i \langle \boldsymbol{\zeta}_i, d\mathbf{W}_t \rangle_{\mathbb{R}^d}. \end{aligned}$$

Let $\psi : \mathbb{R}^d \to \mathbb{R}$ a payoff function. The price of an option with payoff ψ and underlying d stocks is given by

$$u(t, \mathbf{x}) = \mathbb{E}[\psi(\mathbf{X}_t)|\mathbf{X}_0 = \mathbf{x}] = \mathbb{E}[\psi(x_0^i \exp\big((\mu - \frac{1}{2}||\beta_i \boldsymbol{\zeta}_i||_{\mathbb{R}^d}^2)t + \langle \beta_i \boldsymbol{\zeta}_i, \mathbf{W}_t \rangle_{\mathbb{R}^d}\big))]. \tag{3.18}$$

The Feynman-Kac theorem states that the function u satisfies for every $t \in [0,T]$ and $\mathbf{x} \in \mathbb{R}^d, u(0, \mathbf{x}) = \psi(\mathbf{x})$ and

$$(\partial_t u)(t, \mathbf{x}) = \frac{1}{2} \sum_{i,j=1}^d x_i x_j \beta_i \beta_j \langle \boldsymbol{\zeta}_i, \boldsymbol{\zeta}_j \rangle_{\mathbb{R}^d} (\partial^2_{x_i x_j} u)(t, \mathbf{x}) + \sum_{i=1}^d \mu x_i (\partial_{x_i} u)(t, \mathbf{x}). \tag{3.19}$$

Clearly, degeneracy exists in (**??**). Proceeding as in the uncorrelated noise case, we introduce the process $Y^i = \log\big(X^i\big)$ for every $i \in \{1, \dots, d\}$. Using Itö's formula, we can write

$$dY_t^i = (\mu - \frac{1}{2}\beta_i^2 ||\boldsymbol{\zeta}_i||^2_{\mathbb{R}^d})dt + \langle \beta_i \boldsymbol{\zeta}_i, d\mathbf{W}_t \rangle_{\mathbb{R}^d}, \quad Y_0^i = \log\big(x_0^i\big), \quad \mathbb{P} - a.s, \tag{3.20}$$

where we have used the fact that

$$\begin{aligned} d[X^i]_t &= d[X_0^i + \int_0^t \mu X_s^i ds + \int_0^t X_s^i \langle \beta_i \boldsymbol{\zeta}_i, d\mathbf{W}_s \rangle_{\mathbb{R}^d}]_t \\ &= d[\int_0^t \mu X_s^i ds + \int_0^t \sum_{l=1}^d \beta_i X_s^i \zeta_i^{(l)} dW_s^{(l)}]_t \\ &= d[\sum_{l=1}^d \int_0^t \beta_i X_s^i \zeta_i^{(l)} dW_s^{(l)}]_t \\ &= \sum_{l=1}^d d[\int_0^t \beta_i X_s^i \zeta_i^{(l)} dW_s^{(l)}]_t \quad \text{(By independence of the BM's.)} \\ &= \sum_{l=1}^d d\int_0^t \beta_i^2 (X_s^i)^2 (\zeta_i^{(l)})^2 ds = \sum_{l=1}^d \beta_i^2 (X_t^i)^2 (\zeta_i^{(l)})^2 dt \\ &= \beta_i^2 (X_t^i)^2 ||\boldsymbol{\zeta}_i||^2_{\mathbb{R}^d}. \end{aligned}$$

In vector form, the process $\mathbf{Y}$ follows

$$d\mathbf{Y}_t = [\mu - \frac{1}{2}\beta_i^2 ||\boldsymbol{\zeta}_i||^2_{\mathbb{R}^d}]_{i=1}^d dt + [\beta_i \boldsymbol{\zeta}_i]_{i=1}^d d\mathbf{W}_t, \quad \mathbf{Y}_0 = \log(\mathbf{x}_0), \quad \mathbb{P} - a.s. \tag{3.21}$$

Let $v(t, \log \mathbf{x}) := u(t, \mathbf{x})$. We then find that $v(t, \mathbf{y}) = u(t, \exp(\mathbf{y})) = \mathbb{E}[\psi(\exp(\mathbf{Y}_t))| \exp(\mathbf{Y}_0) = \exp(\mathbf{y})] = \mathbb{E}[\tilde{\psi}(\mathbf{Y}_t)|\mathbf{Y}_0 = \mathbf{y}] = \mathbb{E}[\tilde{\psi}([y^i + (\mu - \frac{1}{2}\beta_i^2 ||\boldsymbol{\zeta}_i||^2_{\mathbb{R}^d})t + \langle \beta_i \boldsymbol{\zeta}_i, \mathbf{W}_t \rangle_{\mathbb{R}^d}]_{i=1}^d)]$ satisfies for every $t \in [0, T], \mathbf{y} \in \mathbb{R}^d, v(0, \mathbf{y}) = u(0, \exp(\mathbf{y})) = \psi(\exp(\mathbf{y})) = \tilde{\psi}(\mathbf{y})$ and

$$(\partial_t v)(t, \mathbf{y}) = \frac{1}{2} \sum_{i,j=1}^d \beta_i \beta_j \langle \boldsymbol{\zeta}_i, \boldsymbol{\zeta}_j \rangle_{\mathbb{R}^d} (\partial^2_{y_i y_j} v)(t, \mathbf{y}) + \sum_{i=1}^d (\mu - \frac{1}{2}\beta_i^2 ||\boldsymbol{\zeta}_i||^2_{\mathbb{R}^d})(\partial_{y_i} v)(t, \mathbf{y}). \tag{3.22}$$

For this model, again we choose to train to non-transformed model and the complete setup is given by

$$\begin{cases} (\partial_t u)(t, \mathbf{x}) &= \frac{1}{2} \sum_{i,j=1}^d x_i x_j \beta_i \beta_j \langle \boldsymbol{\zeta}_i, \boldsymbol{\zeta}_j \rangle_{\mathbb{R}^d} (\partial^2_{x_i x_j} u)(t, \mathbf{x}) + \sum_{i=1}^d \mu x_i (\partial_{x_i} u)(t, \mathbf{x}), \quad (t, \mathbf{x}) \in \Omega, \\ u(t, \mathbf{x}) &= \frac{1}{N_s} \sum_{n=1}^{N_s} \psi([x^i \exp\Big((\mu - \frac{1}{2}\beta_i^2 ||\boldsymbol{\zeta}_i||^2_{\mathbb{R}^d})t + \langle \beta_i \boldsymbol{\zeta}_i, \mathbf{W}_t^n \rangle_{\mathbb{R}^d}\Big)]_{i=1}^d), \quad t \in (0, T), \mathbf{x} \in \partial D, \\ u(0, \mathbf{x}) &= \psi(\mathbf{x}), \quad \mathbf{x} \in D. \end{cases}$$

Note that we again make use of the Feynman-Kac solution to impose Dirichlet boundary conditions, where the expectation is replaced by a sample mean of size $N_s \in \mathbb{N}$. In addition to that, the differential operator $\mathcal{L}$ in (**??**) is given for all $u \in C^{1,2}([0,T]\times\mathbb{R}^d,\mathbb{R})$ by

$$\mathcal{L}u(t,\mathbf{x}) = (\partial_t u)(t,\mathbf{x}) - \frac{1}{2}\sum_{i,j=1}^{d} x_i x_j \beta_i \beta_j \langle \boldsymbol{\zeta}_i, \boldsymbol{\zeta}_j\rangle_{\mathbb{R}^d} (\partial^2_{x_i x_j} u)(t,\mathbf{x}) - \sum_{i=1}^{d} \mu x_i (\partial_{x_i} u)(t,\mathbf{x}).$$

For the uncorrelated and correlated Brownian motions cases, we use the analytical form of the exact solution (**??**) for $d \leq 10$ and thus avoiding the error induced by approximation the expectation by the sample mean. However, for $d > 10$ the evaluation of the multivariate normal cumulative distribution in (**??**) becomes extremely expensive and time consuming so that we resort the Monte-Carlo sampling for approximating the exact solution.

3.5 Heston Model

In this section we consider the multi-dimensional Heston model. Let $\delta \in \mathbb{N}$, $d = 2\delta$ and let $\alpha, \beta \in \mathbb{R}, \theta, \kappa > 0$ satisfying the Feller condition $2\kappa\theta > \beta^2$. The Heston model is defined for every $i \in \{1,\ldots,\delta\}$ by the system of stock and variance processes

$$dX_t^{2i-1} = \alpha X_t^{2i-1} dt + \sqrt{X_t^{2i}} X_t^{2i-1} dW_t^{2i-1}, \quad X_0^{2i-1} = x_0^{2i-1}, \quad \mathbb{P}-a.s., \tag{3.23}$$

$$dX_t^{2i} = \kappa(\theta - X_t^{2i})dt + \beta\sqrt{X_t^{2i}} d\tilde{W}_t^{2i}, \quad X_0^{2i} = x_0^{2i}, \quad \mathbb{P}-a.s, \tag{3.24}$$

The Feller condition ensures that the variance process X^{2i} remains strictly positive. Also, the two Brownian motions W^{2i-1} and $\tilde{W}^{2i}$ are correlated with a correlation coefficient $\rho \in (-1,1)$. We thus introduce a Brownian motion $W^{2i} : \Omega \times [0,T] \to \mathbb{R}$ independent from W^{2i-1} such that for all $t \in [0,T]$, it holds that $\tilde{W}_t^{2i} = \rho W_t^{2i-1} + \sqrt{1-\rho^2} W_t^{2i}$. The Heston model then becomes

$$\begin{aligned} dX_t^{2i-1} &= \alpha X_t^{2i-1} dt + \sqrt{X_t^{2i}} X_t^{2i-1} dW_t^{2i-1}, \quad X_0^{2i-1} = x_0^{2i-1}, \quad \mathbb{P}-a.s., \\ dX_t^{2i} &= \kappa(\theta - X_t^{2i})dt + \beta\rho\sqrt{X_t^{2i}} dW_t^{2i-1} + \beta\sqrt{1-\rho^2}\sqrt{X_t^{2i}} dW_t^{2i}, \quad X_0^{2i} = x_0^{2i}, \quad \mathbb{P}-a.s, \end{aligned} \tag{3.25}$$

In vector form, the Heston model can be written as

$$d\mathbf{X}_t = \mu(\mathbf{X}_t)dt + \sigma(\mathbf{X}_t)d\mathbf{W}_t, \quad \mathbf{X}_0 = \mathbf{x}_0, \quad \mathbb{P}-a.s., \tag{3.26}$$

where

$$\mathbf{X}_t = \begin{bmatrix} X_t^{2i-1} \\ X_t^{2i} \end{bmatrix}_{i=1}^{\delta}, \quad \mathbf{W}_t = \begin{bmatrix} W_t^{2i-1} \\ W_t^{2i} \end{bmatrix}_{i=1}^{\delta}.$$

and

$$\mu(\mathbf{X}_t) = \begin{bmatrix} \alpha X_t^{2i-1} \\ \kappa(\theta - X_t^{2i}) \end{bmatrix}_{i=1}^{\delta}, \quad \sigma(\mathbf{X}_t) = \operatorname{diag}\Big(\begin{bmatrix} X_t^{2i-1} & 0 \\ \beta\rho\sqrt{X_t^{2i}} & \beta\sqrt{1-\rho^2}\sqrt{X_t^{2i}} \end{bmatrix}_{i=1}^{\delta} \Big).$$

The solution to (**??**) [**?**] is given by

$$\begin{aligned}\mathbf{X}_t = \sum_{i=1}^{\delta} \Bigg(& \Big[X_0^{2i-1} \exp\Big((\alpha - \frac{X_0^{2i}}{2})t + W_t^{2i-1}\sqrt{X_0^{2i}} \Big) \Big] e_{2i-1} \\ & + \Big[\max\Big\{ \big[\max\{ \frac{\beta}{2}\sqrt{t}, \max\{ \frac{\beta}{2}\sqrt{t}, \sqrt{X_0^{2i}} \} + \frac{\beta}{2}(\rho W_t^{2i-1} + \sqrt{1-\rho^2} W_t^{2i}) \} \big]^2 \\ & + (\kappa\theta - \frac{\beta^2}{4} - \kappa X_0^{2i})t, 0 \Big\} \Big] e_{2i} \Bigg) \end{aligned} \tag{3.27}$$

where $e_i = (\dots, 0, \dots, 1 \dots) \in \mathbb{R}^d$ and the number 1 is located in the i-th entry. Let $\psi : \mathbb{R}^d \to \mathbb{R}$ a payoff function. The price of an option with payoff ψ and d underlying stocks is given by

$$u(t, \mathbf{x}) = \mathbb{E}[\psi(\mathbf{X}_t) | \mathbf{X}_0 = \mathbf{x}]. \tag{3.28}$$

The Feynman-Kac theorem states that the function u satisfies for every $t \in [0, T], \mathbf{x} \in \mathbb{R}^d, u(0, \mathbf{x}) = \psi(\mathbf{x})$ and

$$\begin{aligned}(\partial_t u)(t, \mathbf{x}) = & \Big[\sum_{i=1}^{\delta} (\alpha x_{2i-1}(\partial_{x_{2i-1}} u)(t, \mathbf{x})) + \kappa(\theta - x_{2i})(\partial_{x_{2i}} u)(t, \mathbf{x}) \Big] \\ & + \Big[\sum_{i=1}^{\delta} \frac{|x_{2i}|}{2} \big(x_{2i-1}^2 (\partial_{x_{2i-1}^2}^2 u)(t, \mathbf{x}) + 2x_{2i-1}\beta\rho(\partial_{x_{2i-1}x_{2i}}^2 u)(t, \mathbf{x}) + \beta^2(\partial_{x_{2i}^2}^2 u)(t, \mathbf{x}) \big) \Big]. \end{aligned} \tag{3.29}$$

Next, for $i \in \{1, \dots, \delta\}$ we define the process $Y^{2i-1} = \log\big(X^{2i-1}\big)$. Using Itö's formula, the dynamics of Y^{2i-1} are given by

$$dY_t^{2i-1} = (\alpha - \frac{1}{2}X_t^{2i})dt + \sqrt{X_t^{2i}} dW_t^{2i-1}, \quad Y_0^{2i-1} = y_0^{2i-1}, \quad \mathbb{P} - a.s. \tag{3.30}$$

In vector form, the transformed Heston model can be written as

$$d\mathbf{Y}_t = \mu(\mathbf{Y}_t)dt + \sigma(\mathbf{Y}_t)d\mathbf{W}_t, \quad \mathbf{Y}_0 = \mathbf{y}_0, \quad \mathbb{P} - a.s., \tag{3.31}$$

where

$$\mathbf{Y}_t = \begin{bmatrix} Y_t^{2i-1} \\ X_t^{2i} \end{bmatrix}_{i=1}^{\delta}, \quad \mathbf{W}_t = \begin{bmatrix} W_t^{2i-1} \\ W_t^{2i} \end{bmatrix}_{i=1}^{\delta}.$$

and

$$\mu(\mathbf{Y}_t) = \begin{bmatrix} \alpha - \frac{1}{2}X_t^{2i} \\ \kappa(\theta - X_t^{2i}) \end{bmatrix}_{i=1}^{\delta}, \quad \sigma(\mathbf{Y}_t) = \operatorname{diag}\Bigg(\begin{bmatrix} \sqrt{X_{t\,t}^{2i}} & 0 \\ \beta\rho\sqrt{X_t^{2i}} & \beta\sqrt{1-\rho^2}\sqrt{X_t^{2i}} \end{bmatrix}_{i=1}^{\delta} \Bigg).$$

The solution to (**??**) is given by

$$\begin{aligned}\mathbf{Y}_t = \sum_{i=1}^{\delta} \Big(&\Big[Y_0^{2i-1} + (\alpha - \frac{X_0^{2i}}{2})t + W_t^{2i-1}\sqrt{X_0^{2i}} \Big] e_{2i-1} \\ &+ \Big[\max\Big\{ \big[\max\{ \frac{\beta}{2}\sqrt{t}, \max\{ \frac{\beta}{2}\sqrt{t}, \sqrt{X_0^{2i}} \} + \frac{\beta}{2}(\rho W_t^{2i-1} + \sqrt{1-\rho^2} W_t^{2i}) \} \big]^2 \\ &+ (\kappa\theta - \frac{\beta^2}{4} - \kappa X_0^{2i})t, 0 \Big\} \Big] e_{2i} \Big) \end{aligned} \tag{3.32}$$

Let $v(t, \log \mathbf{x}^{2i-1}, \mathbf{x}^{2i}) := u(t, \mathbf{x}^{2i-1}, \mathbf{x}^{2i})$. We then find that $v(t, \mathbf{y}^{2i-1}, \mathbf{x}^{2i})$ satisfies for every $t \in [0, T], \mathbf{y} \in \mathbb{R}^d, v(0, \mathbf{y}^{2i-1}, \mathbf{x}^{2i}) = u(0, \exp(\mathbf{y}^{2i-1}), \mathbf{x}^{2i}) = \psi(\exp(\mathbf{y}^{2i-1}), \mathbf{x}^{2i}) = \tilde{\psi}(\mathbf{y}^{2i-1}, \mathbf{x}^{2i}) = \tilde{\psi}(\mathbf{y})$ and

$$\begin{aligned}(\partial_t v)(t, \mathbf{y}) = &\Big[\sum_{i=1}^{\delta} (\alpha - \frac{1}{2}x_{2i})(\partial_{y_{2i-1}} v)(t, \mathbf{y})) + \kappa(\theta - x_{2i})(\partial_{x_{2i}} v)(t, \mathbf{y}) \Big] \\ &+ \Big[\sum_{i=1}^{\delta} \frac{|x_{2i}|}{2} \big((\partial^2_{y^2_{2i-1}} v)(t, \mathbf{y}) + 2\beta\rho(\partial^2_{y_{2i-1}x_{2i}} v)(t, \mathbf{y}) + \beta^2(\partial^2_{x^2_{2i}} v)(t, \mathbf{y}) \big) \Big]. \end{aligned} \tag{3.33}$$

For the Heston model, we also observed that training the non-transformed model is more performant than the transformed one, and hence the complete setup is given by equation (**??**) with exact solutions imposed on the boundaries (Dirichlet boundary conditions) and initial condition given by the function ψ.

Chapter 4

Numerical Experiments

In this section, we present numerical results for the forward problem, the Greeks and the inverse problem for the corresponding price models discussed in the previous section. More precisely we focus on pricing European options in very high dimensions under the Black-Scholes and Heston models. We then consider approximating the Greeks using automatic differentation. Finally, we consider calibration problems, where we try to estimate the volatility function and option prices from observed noiseless price measurements. For each problem and each dimension, we show the results of the ensemble training, that is we present the neural networks' best setups (number of hidden layers, number of neurons, regulization parameters, etc ...) along with the corresponding training errors, generalization errors and computational times. We note again that the non-transformed models were trained, based on the fact that the underlying solutions are easier to learn. We run the optimization using the second order LBFGS optimizer [?], with mini batches of size 8192 and between 30 and 40 epochs where for each epoch and for each batch, the LBFGS optimizier performs 20 iterations. Thus, the total number of iterations performed is given by $N_{epochs} \times N_{batches} \times 20$.

4.1 Forward Problem

For the forward problem, we consider pricing multi-assets European options under the famous multi-dimensional Black-Scholes and Heston models. For each model, we fix the training set size and the number of epochs the optimizer performs. We then show empirically that PINNs overcome the curse of dimension, that is the exponential growth of the error with the dimension. Of course for this study, the computational time increases with the dimension d as the training algorithm rely on computing the derivatives (gradients and hessians) in the corresponding PDEs, and the number of derivatives to be computed is directly proportional to the dimension d of the problem.

4.1.1 Black-Scholes Model

We consider the d-dimensional Black-Scholes in the domain $\mathcal{D} = [0, T] \times [90, 110]^d$ for pricing European options. We also set the final time $T = 1$ and the risk-free interest rate $r = 0.05$.

4.1.1.1 The case of $d \in \{1, 2\}$

For $d \in \{1, 2\}$, we let $\sigma_1 = \frac{1}{10} + \frac{1}{200}, \sigma_2 = \frac{1}{10} + \frac{2}{200}$ and $\mu = -0.05$. We also consider the initial condition $\psi(\mathbf{x}) = \max(\max(x_1, x_2) - K, 0)$ with $K = 100$. Figure (**??**) shows

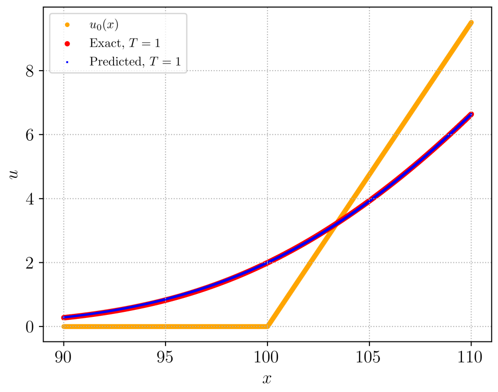
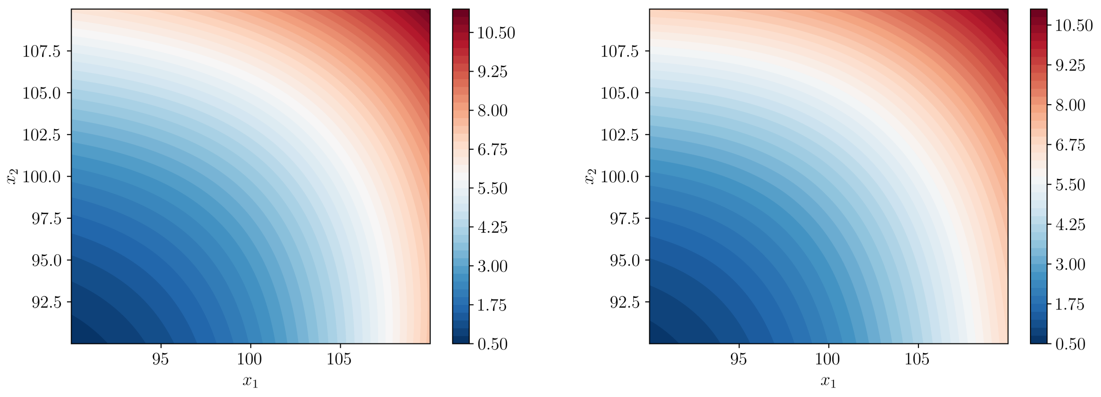


Figure 4.1: Call Option.

a comparison between the PINNs and the Feynman-Kac solutions for the 1-dimensional case. Also, figure (**??**) presents a similar comparison for the 2 dimensional case. Clearly, one can see that the PINNS generate very accurate solutions to the corresponding Black-Scholes model.



(a) PINN Solution.

(b) Feynman-Kac Solution.

Figure 4.2: Comparison of the PINN solution(left) and the Feynman-Kac solution(right) for the 2-dimensional Black-Scholes model with uncorrelated noise.

4.1.1.2 Multi-dimensional Case

For the multi-dimensional case, we consider the **correlated** and **uncorrelated** noise problems in the domain $\mathcal{D} = [0, T] \times [90, 110]^d$.

Uncorrelated noise: We consider the initial condition $\psi(\mathbf{x}) = \max(\max_{1 \leq i \leq d}(x_i) - 100, 0)$. We also let $\mu = -0.05$ and $\sigma_i = \frac{1}{10} + \frac{i}{200}$, $i \in \{1, \dots, d\}$. In table (**??**), the best

d	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}_B\rvert + \lvert\mathcal{S}_I\rvert$	Hidden Layers	Neurons	λ_{res}	λ_{L^2}	ϵ_{T}	ϵ_{G}	Training Time
1	32768	32768	4	28	0.1	0	0.0052	0.74%	3m 30s
2	32768	32768	4	24	0.1	0	0.0767	1.0%	4m 46s
10	32768	32768	4	28	0.1	10^{-6}	0.0467	1.55%	11m 13s
20	32768	32768	4	32	1	10^{-6}	0.0254	1.51%	18m 23s
50	32768	32768	4	28	1	0	0.0097	1.62%	40m 59s
100	32768	32768	4	28	0.1	0	0.0045	1.51%	1h 18m 28s
200	32768	32768	4	32	0.1	0	0.0015	2.28%	2h 28m 55s

Table 4.1: Black-Scholes model with uncorrelated noise

setups with the corresponding training errors, generalization errors and training times are presented for increasing the dimension d. One can see that the behavior of generalization error as a function of the dimension is approximately constant. Slight variations in ϵ_{G} are captured for increasing d and that is indeed an empiricial proof of overcoming the curse of dimension, where for traditional methods the generalization error increases exponentially with the dimension.

Correlated noise: In this case, we consider $\psi(\mathbf{x}) = \max(110 - \min_{1 \leq i \leq d}(x_i), 0)$, $\mu = -0.05$ and $\beta_i = \frac{1}{10} + \frac{i}{200}$ for $i \in \{1, \dots, d\}$. We also set the lower triangular matrix $\mathbf{\Sigma}$ to be the cholesky factor of the matrix $\mathbf{Q} \in \mathbb{R}^{d \times d}$ given by $\mathbf{Q}_{ij} = 1$ for $i = j$ and $\mathbf{Q}_{ij} = 0.5$ for $i \neq j$.
An important issue to address in this case is the expensive computation of the double sum in the corresponding model (**??**). More precisely, we define a scalar ν to be

$$\nu := \sum_{i=1}^{d} \sum_{j=1}^{d} \mathbf{Z}_{ij}, \quad \mathbf{Z}_{ij} = \beta_i \beta_j \langle \boldsymbol{\zeta}_i, \boldsymbol{\zeta}_j \rangle_{\mathbb{R}^d} (\partial^2_{x_i x_j} u)(t, \mathbf{x}), \quad \mathbf{Z}_{ij} = \mathbf{Z}_{ji} \ \forall i, j. \tag{4.1}$$

The matrix $\mathbf{Z} = (\mathbf{Z}_{ij})_{ij} \in \mathbb{R}^{d \times d}$ is symmetric and hence only the upper triangular part, including the diagonal is required to compute the scalar ν. Thus, equation (**??**) can be rewritten as

$$\nu = \sum_{i=1}^{d} \sum_{j=i+1}^{d} 2\mathbf{Z}_{ij} + \sum_{i=1}^{d} \mathbf{Z}_{ii} \tag{4.2}$$

For $d \in \mathbb{N}$, computing ν in the form of (**??**) requires computing the d^2 elements of $\mathbf{Z}$, however one needs to evaluate only $\frac{d}{2}(d+1)$ entries of $\mathbf{Z}$ by adapting equation (**??**) and hence saving $\frac{d}{2}(d-1)$ computations. For large d, that makes a big difference in training

PINNs (computational time-wise) since ν has to be computed in each training iteration. The following plot illustrates the number of computed entries as the dimension d grows. Clearly, as d grows the difference in the number of $\mathbf{Z}$ entries computed between (**??**) and

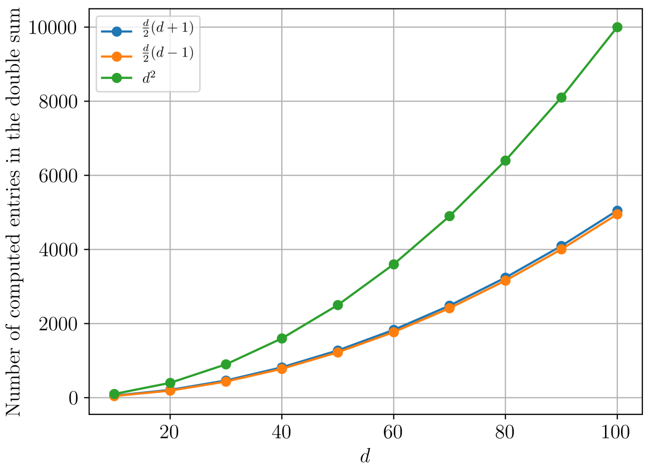


Figure 4.3: The number of $\mathbf{Z}$ entries computed as d grows.

(**??**) increases and hence rendering optimizations necessary.
Another important step towards an efficient implementation of ν is the efficient formulation of the dot product $\langle \boldsymbol{\zeta}_i, \boldsymbol{\zeta}_j \rangle_{\mathbb{R}^d}$. Since $\boldsymbol{\Sigma}$ is lower triangular, $\mathrm{nnz}(\boldsymbol{\zeta}_i) = i + 1$ for all i, where nnz denotes the number of non-zero elements in the input. The dot product can then be formulated as

$$\langle \boldsymbol{\zeta}_i, \boldsymbol{\zeta}_j \rangle_{\mathbb{R}^d} = \langle (\boldsymbol{\zeta}_i)_{0:m}, (\boldsymbol{\zeta}_j)_{0:m} \rangle_{\mathbb{R}^m}, \quad m = 1 + \min(i, j) \leq d. \tag{4.3}$$

Equation (**??**) is useful in our case, especially when one of the vectors in the dot products is sparse, i.e it has only few non-zero elements in its head. Again, table (**??**) shows that

d	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}_B\rvert + \lvert\mathcal{S}_I\rvert$	Hidden Layers	Neurons	λ_{res}	λ_{L^2}	ϵ_{T}	ϵ_{G}	Training Time
2	32768	32768	4	24	1	0	0.0040	0.14%	3m 34s
10	32768	32768	4	28	0.1	10^{-6}	0.0277	1.11%	8m 32s
20	32768	32768	4	24	1	0	0.0198	1.18%	20m 03s
50	32768	32768	4	28	1	0	0.0099	0.69%	38m 03s
100	32768	32768	4	28	0.1	0	0.0064	0.77%	1h 53m 23s

Table 4.2: Black-Scholes model with correlated noise

PINNs are performant in terms of generalization errors for the Black-Scholes model with correlated noise structure. In addition to that, we also observe slight variations in the generalization error ϵ_{G} as function of the dimension d.

4.1.2 Heston Model

In this section, we present numerical results for the multi-dimensional Heston model in the domain $\mathcal{D} = [0,T] \times_{i=1}^{\delta} ([90,110] \times [0.02,0.2])$. We set $T = 1, \alpha = \frac{1}{20}, \beta = \frac{1}{5}, \kappa = \frac{6}{10}, \theta = \frac{1}{25}$ and $\rho = -\frac{1}{5}$. For this model, we choose the initial condition $\psi(\mathbf{x}) = \max(110 - \frac{1}{\delta}\sum_{i=1}^{\delta} x_{2i-1}, 0)$, where $\delta = d/2$.

d	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}_B\rvert + \lvert\mathcal{S}_I\rvert$	Hidden Layers	Neurons	λ_{res}	λ_{L^2}	ϵ_{T}	ϵ_{G}	Training Time
2	32768	32768	4	24	0.1	0	0.0080	1.06%	5m 1s
10	32768	32768	4	28	0.1	0	0.0073	0.81%	10m 56s
30	32768	32768	4	24	0.1	10^{-6}	0.0057	0.89%	20m 32s
50	32768	32768	4	28	0.1	10^{-6}	0.0036	1.33%	1h 5m 27s
100	32768	32768	4	28	0.1	0	0.0030	1.37%	2h 6m 11s

Table 4.3: Heston model

Again for the high dimensional heston model, it is evident that we observe slight variations in the generalization error ϵ_{G} when increasing the spatial dimension d. Also the training time is higher than for the Black-Scholes model. That is because the Heston PDE (**??**) has a more complicated structure compared to the Black-Scholes PDE.

4.2 The Greeks

In computational finance, the Greeks play an important role quantifying the sensitivities of options' prices with respect to model parameters. We list therefore the widely used Greeks, in the one dimensional Black-Scholes setting:

- Delta, $(\Delta) = \partial_x u$, representing the sensitivity of the option price u to the price x of the underlying.
- Gamma, $(\gamma) = \partial_{x^2} u$, representing the sensitivity of Δ to the price x of the underlying.
- Theta, $(\Theta) = -\partial_t u$, representing the sensitivity of the option price u to the time-to-maturity t.
- Rho, $(\rho) = \partial_r u$, representing the sensitivity of the option price u to the risk-free interest rate r.
- Vega, $(\nu) = \partial_\sigma u$, representing the sensitvity of the option price u to the volatility σ.

4.2.1 Computations In The Forward Problem

We consider the 1-dimensional Black-Scholes model given by

$$(\partial_t u)(t,x) - rx(\partial_x u)(t,x) - \frac{\sigma^2}{2}x^2(\partial_{x^2}u)(t,x) + ru(t,x) = 0, \quad (t,x) \in (0,T] \times [0.5, 1.5], \tag{4.4}$$

with initial condition $u(0,x) = \max(x-K, 0)$, where the strike price $K = 1$. We set $T = 1, r = 0.06$ and $\sigma = 60\%$. The solution to (**??**) is given by

$$u(t,x) = xN(d_1) - K\mathrm{e}^{-rt}N(d_2), \quad d_1 = \frac{\log\left(\frac{x}{K}\right) + (r + \frac{1}{2}\sigma^2)t}{\sigma\sqrt{t}}, \quad d_2 = d_1 - \sigma\sqrt{t}, \tag{4.5}$$

where N is the cumulative distribution function of a standard normal random variable. In this case, analytical solutions for the Greeks exist by simple differentiation and are given by

$$\Delta = N(d_1), \tag{4.6}$$

$$\gamma = \frac{1}{x\sigma\sqrt{t}}\Phi(d_1), \tag{4.7}$$

$$\Theta = -rK\mathrm{e}^{rt}N(d_2) - \frac{\sigma}{2\sqrt{t}}x\Phi(d_1), \tag{4.8}$$

$$\nu = x\sqrt{t}\Phi(d_1), \tag{4.9}$$

$$\rho = Kt\mathrm{e}^{-rt}N(d_2), \tag{4.10}$$

where Φ the normal density function. Note that the Greeks are functions of t and x, but we ommit this dependence for the simplicity of notations. Although computing the Greeks is not the main feature of this project, we show a straight-forward way of computing some of the sensitivities. For instance, one can easily observe that Θ, Δ and γ can be already obtained while solving the forward problem for $u(t,x)$, since equation (??) can be reformulated as a relation between the greeks given by

$$-\Theta - rx\Delta - \frac{\sigma^2}{2}\gamma + ru = 0.$$

The last two greeks ν and ρ are not a straight forward consequence of the forward problem, but it is shown that the following relations [?] holds

$$\nu = \sigma t x^2 \gamma, \tag{4.11}$$

$$\rho = -t(u - x\Delta). \tag{4.12}$$

The following table summarizes the generalization errors for the solution of (??) and the Greeks. One could see that the ν is the most problematic term, but this due of course

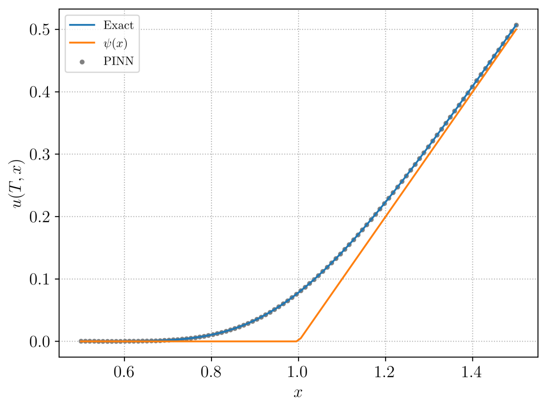
u	Δ	γ	Θ	ν	ρ
0.030%	0.104%	0.876%	0.829%	1.017%	0.147%

Table 4.4: L_2 relative test error of u and the Greeks.

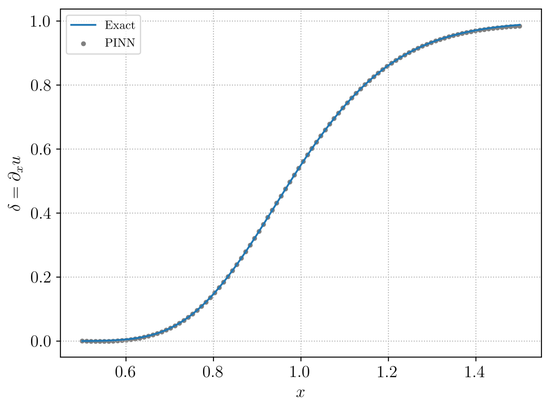
to regularities considerations that are beyond the scope of this project.

The previous procedure of calculating the Greeks Δ, γ, Θ through PINNs generalizes to any model in consideration. However analytical formulas and relations do not always exist. For example, calculating ν assuming that relations (??) do not exist could be done by learning the solution as a function of σ, that is $u_\theta(t,x,\sigma)$ and using automatic differentiation u_θ with respect to σ. This is addressed in the following subsection.

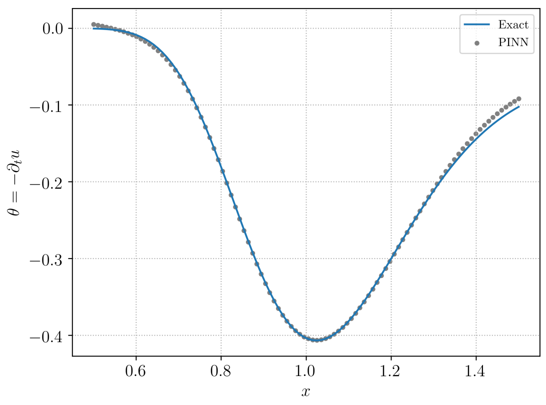
To illustrate the main advantage of PINNs, we provide next a 2-dimensional example of computing the Greeks. In practice, the previous analytical formulas for the Greeks in one dimension do not generalize for higher dimensions, and it becomes hard, sometimes impossible, to derive analytical formulas even for the simple multi-dimensional Black-Scholes model. In addition to that, even deriving analytical formulas for the one dimensional case for complex payoff functions might also be impossible. On the other hand, PINNs are able to approximate accurately the Greeks for arbitrary payoff functions and for multi-dimensional cases.
Figure (??) presents plots of the approximated Greeks against the exact ones and table (??) provides generalization errors for the 2-dimensional Black-Scholes model in the domain $\mathcal{D} = [0,1] \times [0.5, 1.5]^2$ with independent Brownian parts and model parameters $r = 0.06, \sigma_1 = \sigma_2 = 0.6$ and initial condition $\psi(x,y) = \max(\max(x,y) - 1, 0)$. Once again, one can see that the Greeks are approximated accurately in the 2-dimensional case, but slight problems arise near the boundaries.



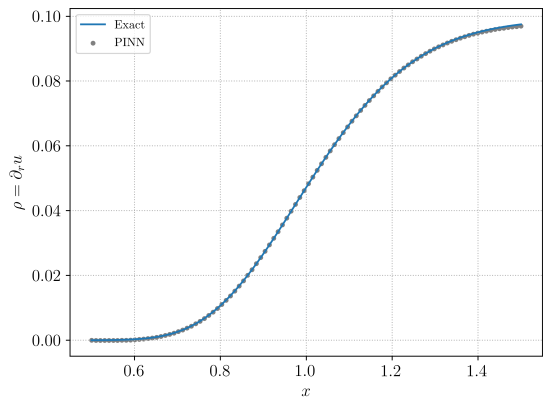
(a) $u(T/2,x)$



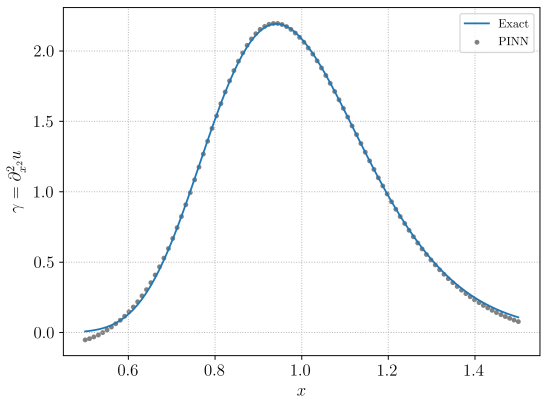
(b) $\Delta(T/2,x)$



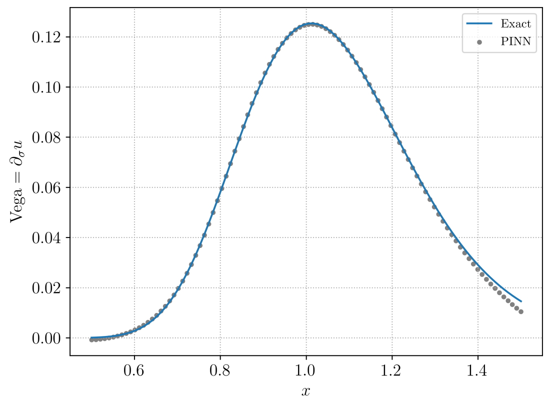
(c) $\Theta(T/2,x)$



(d) $\rho(T/2,x)$



(e) $\gamma(T/2,x)$



(f) $\nu(T/2,x)$

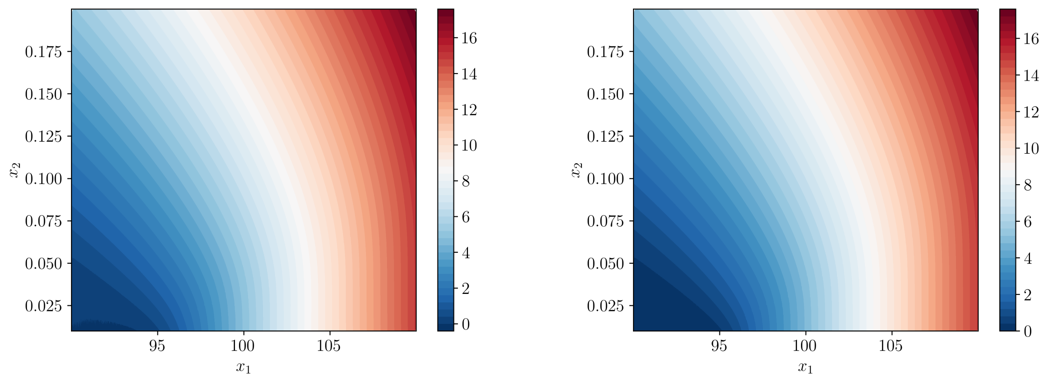
Figure 4.4: Comparisons between the exact and the PINN solutions for the exact solution $u(T,x)$ and the greeks $\Delta, \Theta, \rho, \gamma$ and ν at time T.

u	$\Delta_1 = \partial_{x_1} u$	$\Delta_2 = \partial_{x_2} u$	$\Theta = -\partial_t u$
0.040%	0.305%	0.281%	0.52%

Table 4.5: L_2 relative test error of u and the Greeks Δ_1, Δ_2 and Θ.

4.2.2 Learning The Solution Function of Model's Volatility

We consider the same setting as before, however we learn the solution of (**??**) as a function of the volatility σ. Clearly, our problem is now a 2-dimensional problem, the first dimension being the price $x_1 \in [90, 110]$ and the second being $x_2 = \sigma \in [0.01, 0.2]$. For the greeks computations, we calculate using automatic differentiation Δ, γ, Θ and ν, and we use the relation in (**??**) to compute ρ. Of course, one could also learn the solution as a function ρ, but we do not follow this approach in this example.
The following figure shows the PINN solution $u_\theta(T, x_1, \sigma)$ and exact solution $u(T, x_1, \sigma)$. The generalization error for the solution u in this case is 0.04% with 4 hidden layers and 20 neurons each.



(a) PINN solution $u_\theta(T, x_1, x_2 = \sigma)$ (b) Exact solution $u(T, x_1, x_2 = \sigma)$

Figure 4.5: Comparisons between the exact and the PINN solutions for the solution u at time T.

In this case errors are summarized in the following table (**??**).

u	Δ	γ	Θ	ν	ρ
0.04%	0.08%	0.70%	0.3%	0.2%	0.09%

Table 4.6: L_2 relative test error of u and the Greeks at the final time T and $\sigma = 0.105$.

4.2.3 General Algorithm For Computing The Greeks

In the previous two subsections, we showed two approaches for the Greeks computations in the Black-Scholes model. In the first approach, we made use of some available analytical formulas, i.e. equations (**??**) to compute the option price sensitivities with respect to the model's volatility and risk-free interest rate. In the second approach, we learned the Black-Scholes solution as a function of time, space and additionally the volatility σ. That was beneficial for computing the Greek ν if its analytical formulation in (**??**) was not known.
Next, we present a general algorithm (**??**) for computing the Greeks. Algorithm (**??**) is model independent, that is it works for any model considered. Without loss of generality, we assume that we would like to compute the sensitivity of the option price with respect to the space, time variables and other parameters α, β occuring in the model.

Algorithm 3: Greeks Computation

```
Input: Training dataset $(t_i, \mathbf{x}_i, \alpha_i, \beta_i)_i$, number of epochs $N_{epochs}$, number of batches
  $N_{batches}$, learning rate $\eta$ and a differential operator $\mathcal{L}$.
Output: $u_\theta$ approximating the solution $u$ of the model and the greeks
  $\partial_t u_\theta,\ \nabla_{\mathbf{x}} u_\theta,\ \partial_\alpha u_\theta,\ \partial_\beta u_\theta$ approximations of $\partial_t u,\ \nabla_{\mathbf{x}} u,\ \partial_\alpha u,\ \partial_\beta u$.
  Intialize the weights $\theta$ of the network.
  for $i = 1$ to $N_{epochs}$ do
    for $j = 1$ to $N_{batches}$ do
      Forward pass the batch $\mathcal{B}_j$ through the network to get $u_\theta$.
      Compute the residual $\mathcal{L}(u_\theta) - f$ using automatic differentation.
      Compute the loss $L(\theta) := J(\theta) + \lambda_{L^q}||\theta||_q^q$ according to (??) and (??).
      Perform a backward propagation to compute $\nabla_\theta L(\theta)$.
      Update the weights according to $\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$.
    end for
  end for
  Compute $\partial_t u_\theta,\ \nabla_{\mathbf{x}} u_\theta,\ \partial_\alpha u_\theta,\ \partial_\beta u_\theta$ using automatic differentiation.
```

Comparing algorithm (**??**) to algorithm (**??**), one can clearly see that both are exactly the same. Algorithm (**??**) has an additional line for computing the Greeks using automatic differentiation, and hence the Greeks computations are a straightforward result of the forward problem. Worth noting that requiring computing the option price sensitivities with respect to model parameters requires adding one dimension per parameter, and hence increasing the problem dimensionality, but that would not be problematic as the main goal of PINNs is to overcome the obstacles faced by standard numerical methods for high dimensional problems.

4.3 Inverse Problem

Inverse problems are typical problems encountered in computational finance, such as calculating implied volatilties and calibrating financial models. More precisely, given a set of observed market prices $\mathbb{U} = \{u_i \in \mathbb{R}_+, i \in \mathbb{N}\}$, the aim is to estimate model parameters that replicate the observed market prices. This section is divided into 3 main parts.

For the first part, we consider the problem of calculating the implied volatility in a 1-dimensional setting, considering a set of prices $\mathbb{U}$ generated from the Heston model or from the one dimensional local volatility model. More precisely, for price and variance processes $(X_t)_t, (Y_t)_t$ and two independent Brownian motions $(W_t)_t$ and $(Z_t)_t$, the 2-dimensional Heston model is given by

$$\begin{aligned} dX_t &= \alpha X_t dt + \sqrt{Y_t} X_t dW_t, \quad X_0 = x_0, \\ dY_t &= \kappa(\theta - Y_t)dt + \beta\rho\sqrt{Y_t}dW_t + \beta\sqrt{1-\rho^2}\sqrt{Y_t}dZ_t, \quad Y_0 = y_0, \end{aligned} \tag{4.13}$$

with model parameters $\alpha, \kappa, \theta, \beta$ and ρ. For $T > 0$, the price $u : [0,T] \times \mathbb{R}^2_+ \to \mathbb{R}$ of a European option with payoff $\psi : \mathbb{R} \to \mathbb{R}$ satisfies based on the Feynman-Kac formula (**??**) the pricing parabolic partial differential equation

$$\begin{aligned} (\partial_t u)(t,x,y) &- \alpha x(\partial_x u)(t,x,y) - \kappa(\theta - y)(\partial_y u)(t,x,y) \\ &- \frac{1}{2} y x^2 (\partial^2_{xx} u)(t,x,y) - \frac{1}{2} y\beta^2 (\partial^2_{xx} u)(t,x,y) \\ &- \beta\rho y x(\partial^2_{xy} u)(t,x,y) + ru(t,x,y) = 0, \quad (t,x,y) \in [0,T] \times \mathbb{R}^2_+, \end{aligned} \tag{4.14}$$

with initial condition $u(0,x,y) = \psi(x)$. On the other hand, the local volatility model is defined as

$$dX_t = rX_t dt + s(t, X_t) X_t dW_t, \quad X_0 = x_0, \tag{4.15}$$

where $r \in \mathbb{R}$ is the risk-free rate. The function $s : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ is called the local volatility function. Existence and uniqueness of solutions to (**??**) is guaranteed provided that $(t,x) \mapsto xs(t,x)$ is Lipschitz continuous and satisfies a linear growth condition. For the special case where the local volatility function is given by $s(t,x) \equiv \sigma x^{\rho-1}$ for some $\sigma > 0$ and $\rho \geq 0$, the corresponding model is called the Constant Elasticity of Variance (CEV) model and SDE (**??**) is written as

$$dX_t = rX_t dt + \sigma X_t^{\rho} dW_t, \quad X_0 = x_0. \tag{4.16}$$

For $\rho = 1$, the CEV model boils down to the Black-Scholes model, and for $0 < \rho < 1$ it models the leverage effect [**?**]. In addition to that, for $0 < \rho < 1$ the function $x \mapsto xs(x)$ is not Lipschitz continuous and so existence and uniqueness of solutions to (**??**) is not obvious. However the Yamada-Watanabe [**?**] conditions prove existence and uniqueness of solutions in the latter case.

The pricing equation is given by

$$(\partial_t u)(t,x) - rx(\partial_x u)(t,x) - \sigma^2 x^{2\rho}(\partial^2_{xx} u)(t,x) + ru(t,x) = 0, \quad (t,x) \in [0,T] \times \mathbb{R}_+, \tag{4.17}$$

with initial condition $u(0,x) = \psi(x)$. The implied volatility problem is the estimation of the volatility parameter in the Black-Scholes model that replicates given generated prices based on the CEV (**??**) or the Heston models (**??**).

In the second part, we consider the problem of the one dimensional parametric calibration, that is we estimate the parameters σ and ρ in the one dimensional CEV PDE (**??**) based on generated CEV prices. Also, we perform multi-dimensional parametric calibarion by estimating d-volatilites $\sigma_1, \dots, \sigma_d$ in the d-dimensional Black-Scholes model

$$(\partial_t u)(t,\mathbf{x}) - \sum_{i=1}^{d} rx_i(\partial_{x_i} u)(t,\mathbf{x}) - \frac{1}{2}\sum_{i=1}^{d} \sigma_i^2 x_i^2 (\partial^2_{x_i^2} u)(t,\mathbf{x}) + ru(t,\mathbf{x}) = 0, \quad (t,\mathbf{x}) \in [0,T] \times \mathbb{R}^d_+. \tag{4.18}$$

In the third part, we perform multi-dimensional non-parametric calibration of (**??**) by estimating the local volatility $\mathbf{x} \mapsto s(\mathbf{x}) = \mathrm{diag}(\sigma_1 x_1, \dots, \sigma_d x_d)$, from generated Black-Scholes prices.

4.3.1 Implied Volatility

Given a set of market prices $\mathbb{U} = \{u_i \in \mathbb{R}_+, i \in \mathbb{N}\}$, the implied volatility problem is to estimate the volatility parameter σ in the Black-Scholes model

$$dX_t = rX_t dt + \sigma X_t dW_t, \quad X_0 = x_0, \tag{4.19}$$

such that prices generated using (**??**) and the estimated implied volatility match the observed market prices $\mathbb{U}$. In our case, σ is typically a function of the price, i.e $\sigma \equiv \sigma(x)$. More precisely, we seek to approximate the inverse of the function $\sigma \mapsto u_{BS}(\sigma)$ where u_{BS} is the Black-Scholes price for a given parameter σ. It is well known that the price $\sigma \mapsto u_{BS}(\sigma)$ is strictly increasing in σ so that the inverse is well defined and the implied volatility exists and is unique. Note that we do not use any Tikhonov regularization for the implied volatility problem.

4.3.1.1 CEV Model Implied Volatility

The CEV implied volatility function $x \mapsto \sigma_{imp}(x)$ is the volatility parameter that should be inserted in the Black-Scholes model for the prices generated from the Black-Scholes model to meet the prices generated from the CEV model. One can clearly see that $\sigma_{imp}(x) = \sigma x^{\rho-1}$ where σ and ρ are the two parameters in the CEV model. The implied volatility estimation is then an inverse problem and we make use of PINNs to calculate the CEV implied volatility and at the same time price options through the Black-Scholes model with the learned implied volatility. Note that both learning σ_{imp} and the solution

u of the Black-Scholes PDE are done simulteanously. The general approach works as follows:

1. Run the forward problem to learn the solution u of (**??**) in the domain $\mathcal{D} = [0,1] \times [0.5, 1.5]$ with model parameters $r = 0.06, \sigma = 0.06,\ \rho \in \{0.35, 0.6, 0.85, 1\}$, $K = 1$ and initial condition $\psi(x) = \max(K - x, 0)$. Denote the learned solution by the network u_θ^{CEV}.

▼

2. Use the network u_θ^{CEV} to generate a set $\mathbb{U}$ of CEV measurements.

▼

3. Run the inverse problem to learn the solution u of the Black-Scholes PDE and the implied volatility σ_{imp}.

Table 4.7: Steps for estimating the CEV implied volatility.

Table (**??**) summarizes the results for different values of ρ with $\lambda_{res} = 0.1$ and $\lambda_{L^2} = 0$. We show the best network setups with the corresponding training errors ϵ_{T}, solution u generalization errors ϵ_{G}, implied volatility generalization errors and the training times for each case. Clearly, PINNs were abel to provide low generalization errors for both the solution u and the implied volatility σ.

In addition, CEV models are well known for generating volatility skews and hence fitting

$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}_I\rvert$	$\lvert\mathcal{S}_M\rvert$	ρ	Hidden Layers	Neurons	ϵ_{T}	ϵ_{G}	ϵ_σ	Training Time
16384	16384	4096	0.35	8	20	0.00025	0.070%	0.99%	6m 50s
16384	16384	4096	0.6	8	24	0.00026	0.045%	0.69%	5m 32s
16384	16384	4096	0.85	8	24	0.00020	0.110%	1.2%	5m 28s
16384	16384	4096	1	8	24	0.00022	0.016%	0.3%	5m 49s

Table 4.8: The CEV implied volatilities and the Black-Scholes prices results for different values of the CEV parameter ρ.

market data well. Indeed figure (**??**) shows the learned implied volatility σ_θ as a function of moneyness x/K. The skewness clearly depends on the ρ parameter in such a way that decreasing ρ increases the skewness of the implied volatility curve. Also, note that the Black-Scholes models assumes constant volatility and that is well approximated for the case $\rho = 1$ (blue points in figure (**??**)).

4.3.1.2 Heston Model Implied Volatility

Another important model to consider for computing the implied volatility is the Heston model with price and variance processes $(X_t)_t$ and $(Y_t)_t$, respectively. The Heston model is known for producing volatility smiles. For that reason, we proceed by computing call option prices using the COS method [**?**] and then we estimate the corresponding volatility

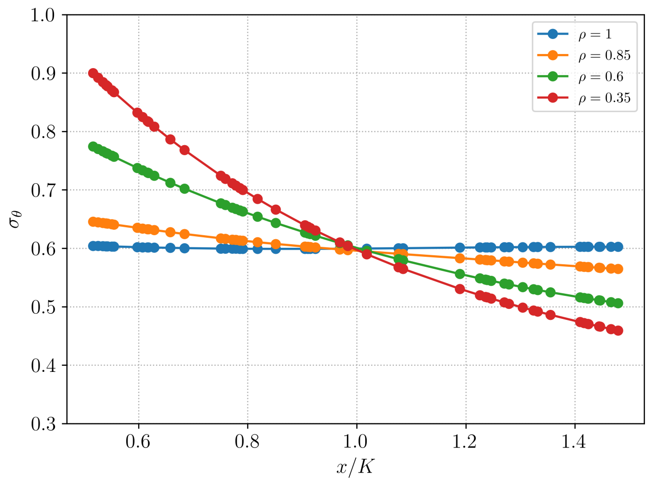


Figure 4.6: The PINN CEV implied volatility σ_θ for different values of ρ.

smile while also pricing options using the Black-Scholes model and the estimated implied volatility. More precisely, the approach is:

1. Compute a set $\mathbb{U}$ of Heston call option prices using the COS method with strike $K = 1$ and model parameters $\alpha = 0.02, \rho = -0.05, \beta = 0.3, \theta = 0.1, \kappa = 1.5$ and initial variance $Y_0 = 0.1$. We consider the time and stock price intervals $[0.5, 1]$ and $[0.7, 1.3]$, respectively.

▼

2. Run the inverse problem to learn the solution u of the Black-Scholes PDE and the volatility smile $\sigma_{imp}(x/K)$ at time $\mathrm{T} = 1$.

Table 4.9: Steps for estimating the Heston implied volatility.

Table (**??**) summarizes the obtained results, with $\lambda_{L^2} = 0$ and $\lambda_{res} = 1$. Again, the

$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}_I\rvert$	$\lvert\mathcal{S}_M\rvert$	Hidden Layers	Neurons	ϵ_T	ϵ_G	ϵ_σ	Training Time
16384	8192	4096	8	24	3.40×10^{-5}	0.081%	1.57%	1m 50s

Table 4.10: The Heston implied volatilities and the Black-Scholes prices.

PINNs prove to be performant in this case where the solution u is approximated very accurately along with the implied volatility. Note that the exact implied volatility σ_{imp} is calculated using the MATLAB function **bsimpv**.

4.3.2 Parametric Calibration

4.3.2.1 The 1-dimensional case

Another type of inverse problems is parametric calibration. In the context of local volatility models, one could fix the local volatility function in (**??**) to some known function and tries to estimate some of its parameters. More precisely, we consider again the CEV model, and hence fixing the local volatility to be the function $x \mapsto s(x) = \sigma x^{\rho}$ with $\sigma = 0.6$ and $\rho \in \{0.35, 0.6, 0.85, 1\}$. We then make use of PINNS to learn the two parameters σ and ρ from observed price values while pricing at the same time options using the estimated values of the parameters. This inverse problem is ill-posed as different combinations of σ and ρ could match the observed prices. For that reason we add a Tikhonov regularization in (**??**) to ensure well-posedness with the Tikhonov functional given by $\Phi(\sigma,\rho) = ||\nabla_x \sigma||^2_{L^2} + ||\nabla_x \rho||^2_{L^2}$. The general approach works as follows:

1. Run the forward problem to learn the solution u of (**??**) in the domain $\mathcal{D} = [0,1] \times [0.5, 1.5]$ with model parameters $r = 0.06, \sigma = 0.06$, $\rho \in \{0.35, 0.6, 0.85, 1\}$, $K = 1$ and initial condition $\psi(x) = \max(K - x, 0)$. Denote the learned solution by the network u_θ^{CEV}.

▼

2. Use the network u_θ^{CEV} to generate a set $\mathbb{U}$ of CEV measurements.

▼

3. Run the inverse problem to learn the solution u of CEV PDE (**??**) and the parameters σ and ρ.

Table 4.11: Steps for the parametric calibration of the one dimensional CEV model.

$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}_I\rvert$	$\lvert\mathcal{S}_M\rvert$	ρ	Hidden Layers	Neurons	λ_Φ	ϵ_{T}	ϵ_{G}	ϵ_σ	ϵ_ρ	Training Time
16384	8192	4096	0.35	8	20	10^{-6}	0.00014	0.02%	0.14%	0.83%	8m 40s
16384	8192	4096	0.6	8	20	10^{-6}	0.00026	0.04%	0.15%	0.70%	5m 57s
16384	8192	4096	0.85	8	20	10^{-6}	0.00023	0.05%	0.64%	0.55%	10m 5s
16384	8192	4096	1	8	20	10^{-6}	0.00030	0.05%	0.85%	0.27%	6m 34s

Table 4.12: Parametric Calibration of the one dimensional CEV model.

It is evident that PINNs were able to provide very low generalization erros for both parameters σ and ρ.

4.3.2.2 The multi-dimensional case

Next we consider the parametric calibration of the multi-dimensional Black-Scholes model with uncorrelated Brownian parts. More precisely, we estimate the solution u and the volatility matrix $\sigma(x) = \text{diag}(\sigma_1, \ldots, \sigma_d)$. The problem then boils down to estimating the solution u and d constant volatilities $(\sigma_i)_{i=1}^d$ in the d-dimensional setting. For this problem, we run a case study where we chose the Tikhonov functional to be

$$\Phi(\sigma_1,\dots,\sigma_d) = \sum_{i=1}^{d} ||\nabla\sigma_i||_{L^2}^2 = \sum_{i=1}^{d} \int_{\mathcal{D}} (\partial_t\sigma_i)^2 + \sum_{j=1}^{d} (\partial_{x_j}\sigma_i)^2 d\mathbf{x}dt, \tag{4.20}$$

and for each $\lambda_\Phi \in \{0, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 1, 10\}$, we run an ensemble training and pick the best setup based on the best training error ϵ_{T}. The procedure for the corresponding inverse problems works as follows:

1. Run the forward problem to learn the solution u of (**??**) in the domain $\mathcal{D} = [0,1] \times [0.5, 1.5]^d$ with model parameters $r = 0.06, \sigma_i = 0.06$ for all $1 \leq i \leq d$, $K = 1$ and initial condition $\psi(x) = \max(\max_i x_i - K, 0)$. Denote the learned solution by the network u_θ.

▼

2. Use the network u_θ to generate a set $\mathbb{U}$ of price measurements.

▼

3. Run the inverse problem to learn the solution u of (**??**) and the parameters $\sigma_1, \dots, \sigma_d$.

Table 4.13: Steps for the parametric calibration of the d-dimensional Black-Scholes model.

Tables (**??**) and (**??**) present the case study results. More importantly, we present for each λ_Φ the price u generalization error ϵ_{G} and the volatility s generalization error ϵ_s. It is evident from the tables that the price u is very-well approximated while the volatility function estimation is not really affected by the Tikhonov function we are using. For almost all the values of λ_Φ except for large ones ($\lambda_\Phi = 10$), the generalization errors ϵ_s do not vary significantly and so we decide to run the problem without any Tikhonov regularization and focus on the accurate estimation of the price u.

λ_Φ	λ_{L^2}	λ_{res}	ϵ_{T}	ϵ_{G}	ϵ_s
0	0.1	0	0.00152	0.05%	10%
10^{-10}	0.1	0	0.0014	0.05%	7.5%
10^{-8}	1	10^{-6}	0.007	0.11%	6.7%
10^{-6}	0.1	0	0.017	0.08%	6%
10^{-4}	0.1	0	0.0036	0.125%	11.4%
10^{-2}	0.1	10^{-6}	0.0076	0.22%	11.4%
10^{-1}	0.1	10^{-6}	0.0077	0.25%	11.8%
1	0.1	0	0.005	0.26%	12.9%
10	0.1	10^{-6}	0.011	0.56%	59%

Table 4.14: Case study, d=3

λ_Φ	λ_{L^2}	λ_{res}	ϵ_T	ϵ_G	ϵ_s
0	0.1	10^{-6}	0.01	0.22%	10.37%
10^{-10}	0.1	10^{-6}	0.0054	0.18%	7.87%
10^{-8}	0.1	10^{-6}	0.010	0.22%	11.8%
10^{-6}	0.1	10^{-6}	0.0086	0.12%	10%
10^{-4}	0.1	0	0.0075	0.2%	12.4%
10^{-2}	0.1	0	0.010	0.32%	14%
1	0.1	0	0.0124	0.36%	13.9%
10	0.1	10^{-6}	0.021	0.7%	88%

Table 4.15: Case study, d=5

With that in mind, we present in table (**??**) the corresponding results while increasing the space dimension d. In this case, one can clearly see that PINNs turned out to be

d	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}_I\rvert$	$\lvert\mathcal{S}_M\rvert$	Hidden Layers	Neurons	λ_{L^2}	λ_{res}	ϵ_T	ϵ_G
2	16384	8192	8192	8	28	0	0.1	0.00050	0.02%
3	32768	16384	16384	8	28	0	0.1	0.00196	0.08%
5	32768	16384	16384	8	28	10^{-6}	0.1	0.0081	0.12%
10	32768	16384	16384	8	32	10^{-6}	1	0.0121	0.8%
20	32768	16384	16384	8	32	10^{-6}	0.1	0.0128	0.85%
50	32768	16384	16384	4	36	0	0.1	0.0129	1.6%
100	32768	16384	16384	4	32	0	0.1	0.0164	2.4%

Table 4.16: Parametric calibration of the d-dimensional Black-Scholes model.

performant in estimating the prices accurately even in very high dimensions.

4.3.3 Non-Parametric Calibration

4.3.3.1 The 1-dimensional case

Yet also another type of inverse problems is non-parametric calibration. One could use equation (**??**) without assuming any particular form on the local volatility function $x \mapsto s(x)$ and tries to estimate it for all x in some specified domain. Again, we consider the CEV model for $\sigma = 0.6$ and $\rho \in \{0.35, 0.6, 0.85, 1\}$. We then make use of PINNS to estimate the function $s(x) = \sigma x^{\rho}$ in the domain $[0.5, 1.5]$ from observed CEV prices. The general approach works as (**??**), but the last stop is modified so that we learn the volatility function $x \mapsto s(x) = \sigma x^{\rho}$.

$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}_I\rvert$	$\lvert\mathcal{S}_M\rvert$	ρ	Hidden Layers	Neurons	ϵ_{T}	ϵ_{G}	ϵ_s	Training Time
16384	8192	4096	0.35	8	28	0.00027	0.042%	1.12%	7m 1s
16384	8192	4096	0.6	8	28	0.00025	0.070%	1.10%	8m 9s
16384	8192	4096	0.85	8	28	0.00019	0.063%	0.84%	8m 11s
16384	8192	4096	1	8	28	0.00025	0.078%	1.5%	6m 38s

Table 4.17: Non-parametric calibration of the one dimensional CEV model.

In this case, PINNs estimated the solution and the volatility function s accurately. However, we expected the increase in the error on s since we do not provide any addition info on its structure as in the parameteric case.

4.3.3.2 The multi-dimensional case

For this case, the general procedure is similar to (**??**). However the last step is modified so that we learn the whole volatility mapping, that is the map $\mathbf{x} \mapsto s(\mathbf{x}) = \mathrm{diag}(\sigma_1 x_1, \ldots, \sigma_d x_d)$. In addition to that, we also run a case study similar to that in the previous section, that we vary the parameter λ_Φ and for each one we run an ensemble training in order to obtain the best setup. Tables (**??**) and (**??**) show that still the generalization errors ϵ_s do not vary significantly while the price u is still approximated very-well, so that we decide to run the inverse problem without any Tikhonov regularization

In the following table (**??**) we present the results for varying the space dimension d and with choosing the Tikhonov parameter $\lambda_\Phi = 0$. Again, for the non-parametric calibration, the PINNs prove to be performant in the estimation of the prices, with generalization errors smaller than 3% even for 100 dimensions.

However, generalization errors on the volatility function are not shown in the previous tables for $d > 1$, as PINNs are not able yet to approximate the true underlying volatility accurately. This is of course due to the ill-posed nature of the corresponding inverse problem, escpecially in high dimensions. More study and research will be done on the type of regularization that one should include in the loss function so that the volatility function is well approximated for $d > 1$. However, this will not be explored in the current thesis and will be done as a future work. For the case of $d = 1$, previous results show

λ_Φ	λ_{L^2}	λ_{res}	ϵ_{T}	ϵ_{G}	ϵ_s
0	0.1	0	0.0005	0.052%	12.2%
10^{-10}	0.1	0	0.0004	0.046%	20.74%
10^{-8}	1	0	0.0005	0.06%	14.8%
10^{-6}	0.1	0	0.0006	0.08%	20.1%
10^{-4}	0.1	0	0.0009	0.144%	22.4%
10^{-2}	0.1	0	0.0013	0.28%	21.5%
10^{-1}	0.1	0	0.0022	0.67%	78.5%
1	0.1	10^{-6}	0.006	0.54%	57.5%
10	0.1	10^{-6}	0.0063	0.76%	27.8%

Table 4.18: Case study, d=2

λ_Φ	λ_{L^2}	λ_{res}	ϵ_{T}	ϵ_{G}	ϵ_s
0	0.1	0	0.0018	0.075%	24.7%
10^{-10}	0.1	0	0.0018	0.063%	23.1%
10^{-8}	0.1	0	0.0019	0.071%	22.5%
10^{-6}	0.1	0	0.0024	0.10%	22.1%
10^{-4}	0.1	0	0.00264	0.155%	26.4%
10^{-2}	0.1	10^{-6}	0.0069	0.25%	28.4%
10^{-1}	0.1	10^{-6}	0.0074	0.25%	27.3%
1	0.1	10^{-6}	0.0096	0.64%	30.5%
10	0.1	10^{-6}	0.0095	0.63%	30.8%

Table 4.19: Case study, d=3

d	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}_I\rvert$	$\lvert\mathcal{S}_M\rvert$	Hidden Layers	Neurons	λ_{L^2}	λ_{res}	ϵ_{T}	ϵ_{G}
2	32768	16384	8192	8	28	0	0.1	0.0005	0.052%
3	32768	16384	16384	8	28	0	0.1	0.0018	0.075%
5	32768	16384	16384	8	24	10^{-6}	0.1	0.0087	0.16%
10	32768	16384	16384	8	32	10^{-6}	0.1	0.0127	1.1%
20	32768	16384	16384	8	32	10^{-6}	0.1	0.0130	0.76%
50	32768	16384	16384	8	36	10^{-6}	0.1	0.0125	1.57%
100	32768	16384	16384	8	36	0	0.1	0.0155	2.35%

Table 4.20: Non-parametric calibration of the d-dimensional Black-Scholes model.

that PINNs are able to reconstruct to the volatility or the CEV parameters ρ and σ accurately, sometimes with the help of a corresponding Tikhonov regularizaton.

4.3.4 Out-Of-Sample Pricing

In this section we consider the out-of-sample calibration. More precisely, we generate a set $\mathbb{U}$ of observed prices in the domain $[0.5, 1.5]^d \times [0, \frac{T}{2}]$ and we train the PINN to learn the solution u and the volatility function s in $\mathcal{D} = [0.5, 1.5]^d \times [0, T]$. From table (**??**), it is

d	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}_I\rvert$	$\lvert\mathcal{S}_M\rvert$	Hidden Layers	Neurons	λ_{L^2}	λ_{res}	ϵ_{T}	ϵ_{G}
2	32768	16384	8192	8	28	0	0.1	0.0004	0.27%
5	32768	16384	16384	8	28	10^{-6}	0.1	0.0084	1.56%
10	32768	16384	16384	8	28	10^{-6}	0.1	0.0184	3.70%
20	32768	16384	16384	8	36	10^{-6}	1	0.0164	5.73%
30	32768	16384	16384	8	32	0	0.1	0.0291	11.9%
50	32768	16384	16384	8	36	10^{-6}	0.1	0.0522	22.1%

Table 4.21: Out-Of-Sample parametric calibration of the d-dimensional Black-Scholes model.

evident that PINNs were able to generate accurate prices even for low spatial dimension d and for moderately high d. However, one cannot expect that the peformance of PINNs in the out-of-sample experiment to be same as in the previous calibration experiments, since we test the network in an unseen time domain $[\frac{1}{2}, T]$. The reason behind that is that even if the true volatilities $\sigma_1, \dots, \sigma_d$ are constants, the network is still learning them as a function of time and so affecting the prices outside the time domain $[0, \frac{1}{2}]$. We have already observe this problem in the previous calibration problems, for example in the parametric calibration of the high dimensional Black-Scholes model. Choosing a suitable and conveniant Tikhonov regularizer that obliges the network to learn the volatilites as constants will certainly improve the ouf-of-sample results as no time dependence is induced while testing in the time interval $[\frac{1}{2}, T]$.

Chapter 5

Extension To Non-Linear Models

We study in this chapter the extension of PINNs to non-linear models. More precisely, we consider the application of PINNs to the non-linear Hamilton-Jacobi-Bellman equation.

5.0.1 Hamilton-Jacobi-Bellman Equation

We consider the non-linear Hamilton-Jacobi-Bellman equation given by:

$$(\partial_t u)(t,\mathbf{x}) + (\Delta u)(t,\mathbf{x}) - \kappa ||(\nabla u)(t,\mathbf{x})||^2 = 0, \tag{5.1}$$

with the terminal condition $u(T,\mathbf{x}) = g(\mathbf{x}) = \log\big(0.5 * (1 + ||\mathbf{x}||^2)\big)$. Performing time variable transform, the corresponding forward problem is formulated as

$$(\partial_t u)(t,\mathbf{x}) - (\Delta u)(t,\mathbf{x}) + \kappa ||(\nabla u)(t,\mathbf{x})||^2 = 0, \tag{5.2}$$

with initial condition $u(0,\mathbf{x}) = g(\mathbf{x})$. The exact solution to (**??**) is given by [**?**]

$$u(t,\mathbf{x}) = -\frac{1}{\kappa}\log\Big(\mathbb{E}[\exp\Big(\kappa g(\mathbf{x} + \sqrt{2}\mathbf{W}_t)\Big)]\Big). \tag{5.3}$$

We finally consider the Hamilton-Jacobi-Bellman model and we let $T = 1$ and $\mathcal{D} = (0,1) \times (-2,2)$. Following the approach in [**?**], we compute the generalization error at the point $(t,\mathbf{x}) = (0,\mathbf{0})$. We refer to the parameter κ as the control strength and to the solution $u(0.\mathbf{0})$ as the optimal cost. For the case $d = 50$, we chose 4 layers with 20 neurons each, $\lambda_{res} = 0.01$ and $\lambda_{L^2} = 10^{-6}$. The following table summarizes the generalization errors as a function of the parameter κ. Intuitevely, the optimal cost

κ	1	3	5	7	10
ϵ_G	1.31%	1.92%	1.33%	0.5%	0.17%

Table 5.1: The generalization error ϵ_G as a function of the parameter κ for the 50-dimensional Hamilton-Jacobi-Bellman equation.

decreases as the control strength increases. The following plot confirms this intuition for the 50-dimensional case.

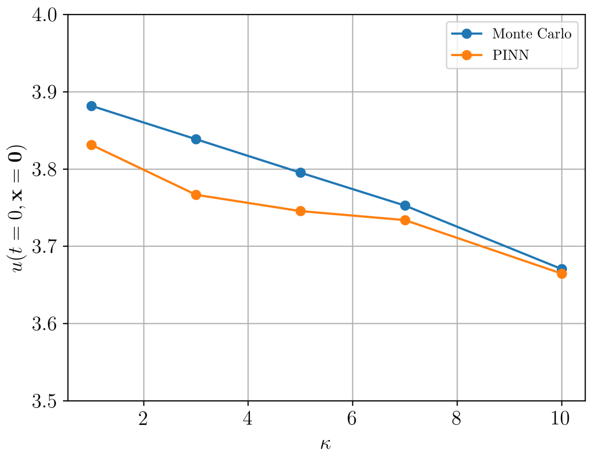


Figure 5.1: Comparison of $u(0, \mathbf{0})$ vs. κ between the PINN solution and the exact solution approximated by the Monte-Carlo method.

For the case of $d = 100$ and $\kappa = 1$, the generalization error obtained is 5.1% for 4 layers and 24 neurons in each layer, $\lambda_{L^2} = 10^{-6}$ and $\lambda_{res} = 0.1$.

Chapter 6

Conclusion

Physics Informed Neural Networks (PINNs) have been recently used as numerical methods for approximating solutions to partial differential equations (PDEs) in different contexts, for instance see [?, ?, ?]. The main focus of this paper is to apply PINNs in the context of high dimensional forward and inverse option pricing. More precisely, the Feynman-Kac (??) is a transition from a probabilistic formulation of the option price to a deterministic formulation, where the option price satisfies the Kolmogorov equation, a linear parabolic differential equation, with suitable initial condition depending on the option type. Since high dimensional option pricing is of huge interest in practice, existing numerical methods suffer from the curse of dimensionality problem and hence renders solving such high dimensional problems impossible.

We present numerical results and illustrate our approach for pricing European options under the high dimensional Black-Scholes and Heston models, where both lead to linear parabolic differential equations. Using the *ensemble training* technique, we show the best neural networks' architectures and the corresponding training and generalization errors, along with the computational times. For our case, the exact solutions were available through the Feynman-Kac probabilistic representation of the underlying solutions, approximated by Monte-Carlo sampling. We were able to show that, at least for the Black-Scholes and Heston models, that PINNS as a numerical method overcome the curse of dimensionality that other numerical methods suffer from. Similar work is present for example in [?], however they consider European and American options with spatial dimension $d \leq 3$. In our paper, we consider only European options with going up to 100 dimensions. On the other hand, neural networks have been used in [?] to solve the same Black-Scholes and Heston models, however they do not take advantage of the availibility of the corresponding PDE and follow a supervised technique approach, requiring the availibility of the exact underlying PDE solution. The main advantage of PINNS is that training can be performed without data points of the exact solution.

We also show that the Greeks, important quantities in finance, can be easily computed after running the forward problem. Finally, we illustrate the applications of PINNs for inverse problems, namely implied volatility, parametric and non-parametric calibration problems. The advantage of PINNS for inverse problems is that the latter can be solved with the same computational cost and efficiency as the forward problem. We were able to recover the volatility function and the corresponding option prices with very high accuracy for one dimensional problems. However as we increase the dimension, the ill-posedness of the inverse problem is more expressed and hence we were able to recover the prices with very high accuracy, but not the volatility function even after adding a Tikhonov regularization term.

Finally, our approach has advantages as well some limitations that should be addressed:

- We were able to overcome the curse of dimensionality and solve the high dimensional Black-Scholes and Heston models for up to 100 dimensions. However our implementation relied on imposing the exact solution on the spatial boundaries (Dirichlet boundary conditions) for $d \geq 2$. For high dimensional problems, literature do not provide any guidance on the treatment of the boundary conditions and so this could be extended and studied more in depth for later improvement.

- For the inverse problem, especially for the high dimensional parametric and non-parametric calibrations of the Black-Scholes model, we were able to recover option prices accurately, but not the volatility function. Tikhonov regularization in the form that we use turned out not to be sufficient for uniqueness. Therefore this also could be studied in more depth to which form of regulizarers should be used to recover the exact volatility in high dimensions.

The computations were performed on the Leonhard cluster of ETH Zürich, with GPU support. The source code for this project is available at https://github.com/ramytanios/refurbished_pinn.
Special thanks to Roberto as his code served as the basis for my extended code and implementations.

Appendix A

Appendix

A.1 Scale Invariance of Black-Scholes and Heston Partial Differential Equations

In this section, we prove an notable feature of the Black-Scholes and Heston PDEs. For instance, stock prices can be usually of high order of magnitude, for instance stock prices could lie in the interval $[90, 110]$. It turned out that training PINNs in such intervals results in poor performances. We show next that one could normalize the price domain and still solve the same partial differential equation with scaled initial conditions. This certainly results in much better performance compared to the non-scaled domain.

A.1.1 Black-Scholes Model

Without loss of generality, we consider the 1-dimensional Black-Scholes model. All the following proofs generalize to higher dimensions, correlated and uncorrelated noise cases. We hence consider the domain $\mathcal{D} = [0, T] \times [L, R]$ where we assume that L and R are relatively large values, of the order of 100. The price $u(t, x)$ satisfies for all $(t, x) \in (0, T] \times [L, R]$

$$(\partial_t u)(t, x) + rx(\partial_x u)(t, x) + \frac{\sigma^2}{2} x^2 (\partial^2_{x^2} u)(t, x) - ru(t, x) = 0 \tag{A.1}$$

with initial condition $u(0, x) = \psi(x)$. We also consider general model parameters r, μ, σ and we define a new space independent variable $\hat{x} = \frac{x}{R}$ and the function $v(t, \hat{x}) = u(t, R\hat{x})$. One can easily show that if u satisfies (**??**), then v satisfies for all $(t, \hat{x}) \in (0, T] \times [\frac{L}{R}, 1]$ the equation

$$(\partial_t u)(t, \hat{x}) + r\hat{x}(\partial_{\hat{x}} u)(t, \hat{x}) + \frac{\sigma^2}{2} \hat{x}^2 (\partial^2_{\hat{x}^2} u)(t, \hat{x}) - ru(t, \hat{x}) = 0, \tag{A.2}$$

with initial condition $v(0, \hat{x}) = \psi(R\hat{x})$.

A.1.2 Heston Model

For the Heston model, we again consider without loss of generality the 2-dimensional case. The price $u(t,x,y)$ satisfies for all $(t,x,y) \in (0,T] \times [L,R] \times [V_L, V_R]$ the equation

$$\begin{aligned}(\partial_t u)(t,x,y) - &\alpha x(\partial_x u)(t,x,y) - \kappa(\theta - y)(\partial_y u)(t,x,y) \qquad (A.3)\\ &- \frac{|y|}{2}(x^2(\partial^2_{x^2} u)(t,x,y) + 2\rho\beta x(\partial^2_{xy} u)(t,x,y) + \beta^2(\partial^2_{x^2} u)(t,x,y)) = 0 \qquad (A.4)\end{aligned}$$

with initial condition $u(0,x,y) = \psi(x)$. We consider general parameters $\kappa, \beta, \theta, \rho$ and we define a new space variable $\hat{x} = \frac{x}{R}$ and the funtion $v(t,\hat{x},y) = u(t,R\hat{x},y)$. Once again, one can show that if u satisfies (**??**) then v satisfies for all $(t,\hat{x}) \in (0,T] \times [\frac{L}{R}, 1] \times [V_L, V_R]$ the equation

$$\begin{aligned}(\partial_t u)(t,\hat{x},y) - &\alpha \hat{x}(\partial_{\hat{x}} u)(t,\hat{x},y) - \kappa(\theta - y)(\partial_y u)(t,\hat{x},y) \qquad (A.5)\\ &- \frac{|y|}{2}(\hat{x}^2(\partial^2_{\hat{x}^2} u)(t,\hat{x},y) + 2\rho\beta \hat{x}(\partial^2_{\hat{x}y} u)(t,\hat{x},y) + \beta^2(\partial^2_{\hat{x}^2} u)(t,\hat{x},y)) = 0 \qquad (A.6)\end{aligned}$$

with initial condition $v(0,\hat{x},y) = \psi(R\hat{x})$.

A.2 Analytical Solution Of The d-dimensional Black-Scholes Model On The Maximum Of d Risky Assets

For d stocks with volatilties $\sigma_1, \sigma_2, \dots, \sigma_d$ and correlation matrix $(\rho_{ij})_{i,j=1}^d$, the analytical solution u of the d-dimensional Black-Scholes model with payoff $\psi(\mathbf{x}) = \max(\max_i x_i - K, 0)$ where K is the strike price, is given by [**?**, **?**]

$$\begin{aligned}u(t,\mathbf{x}) = &\, x_1 N_d(d_1(x_1,K,\sigma_1^2), d_1'(x_1,x_2,\sigma_{12}^2), \dots, d_1'(x_1,x_n,\sigma_{1n}^2), \rho_{112}, \rho_{113}, \dots)\\ &+ x_2 N_d(d_1(x_2,K,\sigma_2^2), d_1'(x_2,x_1,\sigma_{12}^2), \dots, d_1'(x_2,x_n,\sigma_{2n}^2), \rho_{212}, \rho_{223}, \dots)\\ &+ \dots\\ &+ x_n N_d(d_1(x_n,K,\sigma_n^2), d_1'(x_n,x_1,\sigma_{1n}^2), \dots, d_1'(x_n,x_{n-1},\sigma_{n-1n}^2), \rho_{n1n}, \rho_{n2n}, \dots)\\ &- Ke^{-rt}(1 - N_d(-d_2(x_1,K,\sigma_1^2), -d_2(x_2,K,\sigma_2^2), \dots, -d_2(x_n,K,\sigma_n^2), \rho_{12}, \rho_{13}, \dots)), \qquad (A.7)\end{aligned}$$

where

$$\sigma_{ij}^2 = \sigma_i^2 + \sigma_j^2 - 2\rho_{ij}\sigma_i\sigma_j, \tag{A.8}$$

$$\rho_{iij}^2 = \frac{\sigma_i - \rho_{ij}\sigma_j}{\sigma_{ij}}, \tag{A.9}$$

$$\rho_{ijk}^2 = \frac{\sigma_i^2 - \rho_{ij}\sigma_i\sigma_j - \rho_{ik}\sigma_i\sigma_k + \rho_{jk}\sigma_j\sigma_k}{\sigma_{ij}\sigma_{ik}}, \tag{A.10}$$

$$d_1(x, K, \sigma^2) = \frac{\log\left(\frac{x}{K}\right) + (r + \frac{1}{2}\sigma^2)t}{\sigma\sqrt{t}}, \tag{A.11}$$

$$d_2(x, K, \sigma^2) = d_1(x, K, \sigma^2) - \sigma\sqrt{t}, \tag{A.12}$$

$$d_1'(x, y, \sigma^2) = \frac{\log\left(\frac{x}{y}\right) + \frac{1}{2}\sigma^2 t}{\sigma\sqrt{t}}, \tag{A.13}$$

and N_d is the d-th variate standard cumulative normal.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

PHYSICS INFORMED NEURAL NETWORKS IN COMPUTATIONAL FINANCE: HIGH DIMENSIONAL FORWARD & INVERSE OPTION PRICING.

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
TANIOS	RAMY

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, 21-02-2021	[illegible]

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.