NEURAL NETWORKS-BASED ALGORITHMS FOR OPTION PRICING

Master Thesis

to be defended on July 27, 2022

by

Jasper ROU
4664450

Financial Engineering
Master Applied Mathematics
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

SUMMARY

In this research, we consider neural network-algorithms for option pricing. We use the Black-Scholes model and the lifted Heston model. We derive the option pricing partial differential equation (PDE), which we solve with a neural network, and the conditional characteristic function of the stock price which leads to the option price with the COS method. We consider two neural network-algorithms: the Deep Galerkin Method (DGM) and the Time Deep Nitsche Method (TDNM). We extend the TDNM to be able to solve the option pricing PDE by splitting the PDE operator in a symmetric part and an asymmetric part. The splitting method is more stable than transforming the Black-Scholes option pricing PDE to the symmetric heat equation. The DGM can predict options prices perfectly in the Black-Scholes model even when r and σ are added as variables to the neural network. In the lifted Heston model, the DGM can predict option prices perfectly for small dimensions, but has larger errors for larger dimensions. The TDNM is as good as DGM for small times to maturity and volatilities, but has larger errors for large times to maturity and volatilities.

PREFACE

Once trained, the output of a neural network is an analytical function of the input, allowing fast option pricing even in high dimensions. In this research, neural networks-based algorithms for option pricing are developed. We are able to compute option prices in the Black-Scholes model and the lifted Heston model with few dimensions.

This research was done as Master Thesis to finish the Master of Applied Mathematics at the Delft University of Technology. Furthermore, the research is the first step of the Fast Track PhD position at the Delft Institute of Applied Mathematics, which I will continue the coming three years.

Thesis committee:

Prof. Dr. A. Papapantoleon,	Delft University of Technology, supervisor
Dr. F. Fang,	Delft University of Technology

Jasper Rou
Delft, July 19, 2022

CONTENTS

Summary iii

Preface v

1 Introduction 1

2 Theory 3
- 2.1 Option pricing . . . 3
  - 2.1.1 Partial differential equation . . . 4
  - 2.1.2 Characteristic function . . . 6
- 2.2 COS method . . . 7
- 2.3 Black-Scholes model . . . 9
  - 2.3.1 Black-Scholes PDE . . . 10
  - 2.3.2 Black-Scholes characteristic function . . . 10
- 2.4 (Rough) Heston model . . . 11
- 2.5 Lifted Heston model . . . 12

3 Methodology 15
- 3.1 Lifted Heston PDE . . . 15
- 3.2 Lifted Heston characteristic function . . . 16
- 3.3 Calculus of variations . . . 18
- 3.4 Splitting method . . . 20
  - 3.4.1 Black-Scholes model . . . 21
  - 3.4.2 Lifted Heston model . . . 22
- 3.5 Neural network algorithms . . . 23
  - 3.5.1 Deep Galerkin Method . . . 24
  - 3.5.2 Time Deep Nitsche Method . . . 24
  - 3.5.3 Implementation details . . . 26

4 Results 31
- 4.1 Hyperparameters . . . 31
  - 4.1.1 Activation function . . . 31
  - 4.1.2 Learning rate . . . 31
- 4.2 Splitting method . . . 33
- 4.3 Black-Scholes . . . 35
- 4.4 Lifted Heston . . . 36

5 Conclusion 41

Bibliography 43

1

INTRODUCTION

Every day, large amounts of financial derivatives trade at high frequency. One of the most popular derivatives are stock options. Many people and institutions are busy with modelling the prices of these options. The first successful attempt was by Black and Scholes in 1973 [25]. The big advantage of their model is that it has an analytical solution. However, the prices computed in the model are inconsistent with market prices.

Therefore, more realistic methods have appeared since. A more realistic class of financial models are stochastic volatility models such as the Heston model [19]. In contrast to the Black-Scholes model, most of these more realistic models do not have an analytical solution. Therefore, efficient numerical methods are necessary to rapidly price these derivatives. A first class of methods uses the often computable characteristic function of the stock price and transforms this function to the option price. Examples of this kind of method are the Fourier method [8] and the COS method [10]. A second class of methods uses the Feynman–Kac theorem to write the value of an option as the solution of a partial differential equation (PDE), which can be computed with finite difference or Monte Carlo methods [8].

Although some of these methods can calibrate option prices fast, they become slower as the number of dimensions increases. The slowing is due to the exponential growth of the number of grid points with the number of dimensions: this is the so-called curse of dimensionality (CoD). Furthermore, for each set of model parameters, the option price has to be calculated again.

A promising alternative are neural network algorithms. Neural networks do not suffer from the CoD, since they do not use a grid. Alternatively, neural networks sample randomly from the domain to train matching each set of input variables to the correct output. Once a neural network finished training, the output is an analytical function of the input. The price of an option for certain input variables can therefore be calculated fast. Furthermore, the number of input variables can be increased without making the network much slower. Therefore, neural network algorithms are suitable for high-dimensional problems. It is even possible to add the model parameters as input vari-

ables to the neural network. Then the neural network is able to compute the option price for random model parameter choices.

Neural networks received increasing attention in recent years. Some recently developed applications of neural networks are computing model-free price bounds for financial derivatives [23]; optimal control for stochastic optimal control problems [5, 26] and approximating derivative prices under Lévy-processes [17]. Neural networks can solve high-dimensional PDEs in financial applications such as credit valuation, portfolio allocation [15, 16], the Hamilton-Jacobi-Bellman PDE, the Burgers equation, option pricing PDEs, the Fokker-Planck equation, systemic risk and mean field games [5, 27].

This research develops neural networks-based algorithms for option pricing. We price the options in the Black-Scholes model, where the analytical solution exists, and the lifted Heston model, a stochastic volatility model without an analytical solution. In both models, we derive the characteristic function and use the COS method to compute the option prices. Furthermore, we derive the option pricing PDEs and use neural network algorithms to solve these PDEs.

We use two different but related algorithms. The first algorithm is the Deep Galerkin Method [27]. The second algorithm is the Time Deep Nitsche Method (TDNM) [14]. Originally, the TDNM could only be applied to a limited class of problems. However, we design a splitting method so that it can be applied to solving option pricing PDEs.

The structure of this report is as follows. Chapter 2 covers background theory about the models that are used in this research. Chapter 3 explains the methods to price the options. Chapter 4 presents the results of pricing the options with these methods. The report ends with the conclusions in Chapter 5.

2

THEORY

This chapter describes the theoretical knowledge required before we start our research. Section 2.1 explains what options are and which methods there are to price options. Section 2.2 describes the COS method, the numerical method to help price options. Section 2.3 explains how to price options in the famous Black-Scholes model. Section 2.4 expands to the more complicated Heston model and its rough variant. Section 2.5 describes the lifted Heston model, which is the model we will use in the following chapters.

2.1. OPTION PRICING

An option is a contract which gives the owner the right, but not the obligation, to buy or sell an asset at a specified price on or before a specified date. If the option gives the right to buy, it is a *call option*. If the option gives the right to sell, it is a *put option*. If the option can only be exercised at the maturity, it is *European*. If the option can be exercised at any time before or at maturity, it is *American*. The specified price is the strike price and denoted by K. The specified date is the maturity and denoted by T.

There are many options. The most popular options are stock options. In this research, we consider options where the asset is a single stock. If a stock option is on multiple stocks of the same type, we only need to multiply the price in our model with the number of stocks.

If at exercise time, the stock price S is higher than the strike price K, the owner can buy the stock for K and earn $S-K$. If at exercise time, the stock is worth less than the strike price K, the owner can do nothing and earn 0. This pay-off is summarized in the function $\Phi_{call}(S) = (S-K)^+$. Similarly, for a put option, the pay-off is $\Phi_{put}(S) = (K-S)^+$.

In this research, we will use two methods for pricing options. The first method uses a partial differential equation (PDE) and is explained in Subsection 2.1.1. The second method uses the conditional characteristic function and is explained is Subsection 2.1.2.

2.1.1. PARTIAL DIFFERENTIAL EQUATION

The first method for option pricing is to rewrite the option price as the solution of a PDE. This can be done with the Feynman-Kac theorem. Let $(\Omega, \mathscr{F}, \mathbb{P})$ be a probability space. Let $\mathbb{F} = (\mathscr{F}_t)_{t\in[0,T]}$ be a filtration of the sigma algebra $\mathscr{F}$. Let (S, Σ) be a measurable space and $X : [0, T] \times \Omega \to S$ a stochastic process. Then the process X is adapted to the filtration $\mathbb{F}$ if the random variable $X_t : \Omega \to S$ is an $(\mathscr{F}_t, \Sigma)$-measurable function for all $t \in [0, T]$. Let W be a Brownian motion adapted to the filtration.

Definition 2.1. *An Itô process is an adapted stochastic process that can be written as the sum of an integral with respect to a Brownian motion and an integral with respect to time:*

$$X_t = X_0 + \int_0^t \sigma_s dW_s + \int_0^t \mu_s ds, \tag{2.1}$$

for a predictable and W-integrable process σ and a predictable and integrable process μ.

If $\sigma_t \in L^2([0, T] \times \Omega)$, i.e. $\int_0^T \mathbb{E}\left[|\sigma_t|^2\right] dt < \infty$, then $\int_0^t \sigma_s dW_s$ for $t \in [0, T]$ is a martingale with respect to the filtration $\mathbb{F}$ [20]. The differential form of equation (2.1) is

$$X_t = \mu_t dt + \sigma_t dW_t. \tag{2.2}$$

Definition 2.2. *Let $0 = t_0 < t_1 < \ldots < t_n = t$ be a partition and $h = t_k - t_{k-1} = \frac{t}{n}$ the mesh size. The covariation of two processes X and Y is*

$$\langle X, Y\rangle_t = \lim_{h\to 0} \sum_{k=1}^{n} \left(X_{t_k} - X_{t_{k-1}}\right)\left(Y_{t_k} - Y_{t_{k-1}}\right).$$

The quadratic variation is the covariation of a process with itself.

For a Brownian motion, the quadratic variation is [20]

$$\langle W, W\rangle_t = \lim_{h\to 0} \sum_{k=1}^{n} \left(W_{t_k} - W_{t_{k-1}}\right)^2 = \lim_{h\to 0} \sum_{k=1}^{n} t_k - t_{k-1} = \lim_{h\to 0} t = t,$$

while for time t

$$\langle t, t\rangle_t = \lim_{h\to 0} \sum_{k=1}^{n} (t_k - t_{k-1})^2 = \lim_{h\to 0} th = 0.$$

More generally, an Itô process of the form (2.1) has quadratic variation

$$\langle X, X\rangle_t = \int_0^t \sigma_s^2 ds.$$

For Itô processes the famous Itô's formula [24] holds.

Lemma 2.1 (Itô's formula)**.** *Let X be an $\mathbb{R}^d$-valued Itô process and $f : \mathbb{R}^d \to \mathbb{R}$ a $\mathscr{C}^2$-function. Then*

$$df(X_t) = \sum_{i=1}^{d} \frac{\partial f}{\partial X_t^i} dX_t^i + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial X_t^i \partial X_t^j} d\langle X_t^i, X_t^j\rangle. \tag{2.3}$$

If f depends on t as well, then

$$df(t, X_t) = \frac{\partial f}{\partial t} dt + \sum_{i=1}^{d} \frac{\partial f}{\partial X_t^i} dX_t^i + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial X_t^i \partial X_t^j} d\langle X_t^i, X_t^j\rangle.$$

Substituting equation (2.2) in (2.3) gives

$$df(X_t) = \sum_{i=1}^{d} \frac{\partial f}{\partial X_t^i}\left(\mu_t^i dt + \sigma_t^i dW_t^i\right) + \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial X_t^i \partial X_t^j}\sigma_t^i \sigma_t^j dt$$

Definition 2.3. *The infinitesimal generator $\mathscr{A}$ of X_t is*

$$\mathscr{A}f = \lim_{t\downarrow 0}\frac{1}{t}\left(\mathbb{E}[f(X_t)|X_0 = x] - f(x)\right).$$

Suppose the function f is a $\mathscr{C}^2$-function. So it has a Taylor expansion:

$$f(y) = f(x) + \sum_{i=1}^{d}(y_i - x_i)\frac{\partial f}{\partial x_i} + \frac{1}{2}\sum_{i,j=1}^{d}(y_i - x_i)(y_j - x_j)\frac{\partial^2 f}{\partial x_i \partial x_j} + \ldots$$

Then the infinitesimal generator is [20]

$$\begin{aligned}\mathscr{A}f &= \lim_{t\downarrow 0}\frac{1}{t}\left(\sum_{i=1}^{d}\mathbb{E}\left[X_t^i - x_i\right]\frac{\partial f}{\partial x_i} + \frac{1}{2}\sum_{i,j=1}^{d}\mathbb{E}\left[\left(X_t^i - x_i\right)\left(X_t^j - x_j\right)\right]\frac{\partial^2 f}{\partial x_i \partial x_j}\right)\\ &= \sum_{i=1}^{d}\mu_t^i\frac{\partial f}{\partial x_i} + \frac{1}{2}\sum_{i,j=1}^{d}\sigma_t^i\sigma_t^j\frac{\partial^2 f}{\partial x_i \partial x_j}\end{aligned}$$

So we can rewrite $df(X_t) = \mathscr{A}f\,dt$+martingale. $\mathscr{A}$ is used in the following version of the Feynman-Kac Theorem [24].

Theorem 2.1 (Feynman-Kac Theorem)**.** *Let X be an $\mathbb{R}^d$-valued Itô process and $\mathscr{A}$ the infinitesimal generator of X. Then u satisfies the PDE*

$$\begin{aligned}&\frac{\partial u}{\partial t} + \mathscr{A}u = ru, \quad (t,x) \in [0,T]\times\mathbb{R}^d,\\ &u(T,x) = \Phi(x), \quad x \in \mathbb{R}^d\end{aligned} \tag{2.4}$$

if and only if

$$u(t,x) = e^{-r(T-t)}\mathbb{E}\left[\Phi(X_T)|X_t = x\right].$$

In an arbitrage-free market, there is no risk-free profit. Therefore, the price of a European option $V(t,S)$ that pays $\Phi(S_T)$ at maturity should at time t be equal to the expected pay-off under the equivalent martingale measure at maturity discounted at the risk-free rate r: $V(t,S) = e^{-r(T-t)}\mathbb{E}\left[\Phi(S_T)|S_t = S\right]$. Thus, the price of a European option satisfies PDE (2.4) with $\mathscr{A}$ the infinitesimal generator of the stock and r the risk-free rate.

AMERICAN OPTIONS

The price of an American option satisfies almost the same equation as the European option. Since the option can be exercised at any time, it has two values: the continuation value, that the European option has as well, and the intrinsic value. The continuation value is the value of the option if we do not exercise and let the stock continue following

the PDE. The intrinsic value is the value of the option if we exercise, which is Φ. Combining these values gives the following free boundary problem [22]:

$$\begin{cases} \max\left[\frac{\partial V}{\partial t} + \mathscr{A}V - rV, \Phi(S) - V(t,S)\right] = 0, \\ V(T,S) = \Phi(S). \end{cases}$$

This problem is equivalent to

$$\begin{cases} \frac{\partial V}{\partial t} + \mathscr{A}V - rV \leq 0, \\ V(t,S) \geq \Phi(S), \\ V(T,S) = \Phi(S), \\ \left(\frac{\partial V}{\partial t} + \mathscr{A}V - rV\right)(V(t,S) - \Phi(S)) = 0. \end{cases} \tag{2.5}$$

If the stock does not pay dividends and $r \geq 0$ the best exercise strategy for an American call option is to wait until maturity. Therefore, the price of an American call is the same as a European call. The same property holds for an American put option if $r \leq 0$ [22]. A non-dividend paying stock and a nonnegative interest rate are not unusual stylized assumptions. We will assume these two properties and only consider American put options.

2.1.2. CHARACTERISTIC FUNCTION

The second method for option pricing is computing the conditional characteristic function of the stock price. We can then use the conditional characteristic function to compute option prices using Fourier-based methods such as the COS method, as will be explained in Section 2.2. For affine models, it is relatively easy to compute the conditional characteristic function.

Definition 2.4. *An $\mathbb{R}^d$-valued process $\mathbf{X}$ is affine if it has the dynamics*

$$d\mathbf{X}_t = \beta(\mathbf{X}_t)dt + \sigma_1(\mathbf{X}_t)dB_t + \sigma_2(\mathbf{X}_t)dB_t^{\perp},$$

where B_t and $B_t^{\perp}$ are two independent Brownian motions, and

$$\beta(\mathbf{X}_t) = b(t) + \beta_1 X_1 + \ldots + \beta_d X_d$$

$$\sigma_1(\mathbf{X}_t)\sigma_1(\mathbf{X}_t)^T + \sigma_2(\mathbf{X}_t)\sigma_2(\mathbf{X}_t)^T = a(t) + \alpha_1 X_1 + \ldots + \alpha_d X_d.$$

We can use the following Theorem to compute the conditional moment generating function for affine models [11].

Theorem 2.2. *Suppose $\mathbf{X}$ is an $\mathbb{R}^d$-valued affine process. Then the conditional moment generating function of $\mathbf{X}$ is of the form*

$$\mathbb{E}\left[e^{\langle \mathbf{u}, \mathbf{X}_T\rangle} | \mathscr{F}_t\right] = e^{\phi(T-t,\mathbf{u}) + \langle \psi(T-t,\mathbf{u}), \mathbf{X}_t\rangle},$$

with $\mathbf{u} \in \mathbb{R}^d$ *if and only if and* ϕ *and* ψ *satisfy the Ricatti equations*

$$\begin{cases} \frac{\partial \psi_i(t,\mathbf{u})}{\partial t} = \langle \psi(t,\mathbf{u}), \beta_i \rangle + \frac{1}{2}\langle \psi(t,\mathbf{u}), \alpha_i \psi(t,\mathbf{u}) \rangle, \\ \psi_i(0,\mathbf{u}) = u_i, \\ \frac{\partial \phi(t,\mathbf{u})}{\partial t} = \langle \psi(t,\mathbf{u}), b(T-t) \rangle + \frac{1}{2}\langle \psi(t,\mathbf{u}), a(T-t)\psi(t,\mathbf{u}) \rangle, \\ \phi(0,\mathbf{u}) = 0. \end{cases} \tag{2.6}$$

Proof. Denote the conditional moment generating function of $\mathbf{X}$ by

$$M(t) = e^{\phi(T-t,\mathbf{u}) + \langle \psi(T-t,\mathbf{u}), \mathbf{X}_t \rangle}.$$

We know $e^{\phi(0,\mathbf{u}) + \langle \psi(0,\mathbf{u}), \mathbf{X}_T^n \rangle} = \mathbb{E}\left[e^{\langle \mathbf{u}, \mathbf{X}_T \rangle} | \mathscr{F}_T\right] = e^{\langle \mathbf{u}, \mathbf{X}_T \rangle}$. So $\psi_i(0,\mathbf{u}) = u_i$ and $\phi(0,\mathbf{u}) = 0$. Denote $\tau = T - t$. Then Itô's formula gives

$$\frac{dM(t)}{M(t)} = \left(-\partial_\tau \phi(\tau,\mathbf{u}) - \langle \partial_\tau \psi(\tau,\mathbf{u}), \mathbf{X}_t \rangle\right) dt + \langle \psi(\tau,\mathbf{u}), d\mathbf{X}_t \rangle + \frac{1}{2}\langle \psi(\tau,\mathbf{u}), \langle \mathbf{X} \rangle_t \psi(\tau,\mathbf{u}) \rangle =$$

$$\Big(-\partial_\tau \phi(\tau,\mathbf{u}) - \langle \partial_\tau \psi(\tau,\mathbf{u}), \mathbf{X}_t \rangle + \langle \psi(\tau,\mathbf{u}), b(\mathbf{X}_t) \rangle + \frac{1}{2}\big\langle \psi(\tau,\mathbf{u}), \sigma_1(\mathbf{X}_t)\sigma_1(\mathbf{X}_t)^T +$$

$$+ \sigma_2(\mathbf{X}_t)\sigma_2(\mathbf{X}_t)^T \psi(\tau,\mathbf{u}) \big\rangle\Big) dt + \langle \psi(\tau,\mathbf{u}), \sigma_1(\mathbf{X}_t) dB_t + \sigma_2(\mathbf{X}_t) dB_t^\perp \rangle =$$

$$\big(-\partial_\tau \phi(\tau,\mathbf{u}) - \langle \partial_\tau \psi(\tau,\mathbf{u}), \mathbf{X}_t \rangle + \langle \psi(\tau,\mathbf{u}), b(t) + \beta_1 X_1 + \ldots + \beta_d X_d \rangle +$$

$$+ \frac{1}{2}\big\langle \psi(\tau,\mathbf{u}), (a(t) + \alpha_1 X_1 + \ldots + \alpha_d X_d)\, \psi(\tau,\mathbf{u}) \big\rangle\Big) dt + \text{martingale}.$$

Since $M(t)$ is a martingale, the drift term should be 0. This should hold for all X_i. Separating the coefficients of these terms results in Equation (2.6).

Conversely, if Equation (2.6) is satisfied, $M(t)$ is a martingale. Furthermore $M(T) = e^{\langle \mathbf{u}, X_T \rangle}$. So $\mathbb{E}\left[e^{\langle \mathbf{u}, X_T \rangle} | \mathscr{F}_t\right] = \mathbb{E}\left[M(T) | \mathscr{F}_t\right] = M(t)$. □

The conditional characteristic function is the conditional moment generating function, substituting $i\mathbf{u}$ for $\mathbf{u}$. For Fourier-based methods, it is not as simple as for PDE methods to expand to American options. This pricing method is therefore not considered in the case of American option.

2.2. COS METHOD

In this section, we explain a recent method to price options from the conditional characteristic function called the COS method [10]. The method is based on Fourier-cosine expansions. Consider the density function $f(x)$ and its characteristic function $\phi(x)$ related by

$$\phi(\omega) = \int_{\mathbb{R}} e^{i\omega x} f(x)\, dx, \quad f(x) = \frac{1}{2\pi}\int_{\mathbb{R}} e^{-i\omega x} \phi(\omega)\, d\omega. \tag{2.7}$$

For a function on $[0,\pi]$ the cosine expansion is

$$f(\theta) = {\sum_{k=0}^{\infty}}' A_k \cos(k\theta), \quad A_k = \frac{2}{\pi}\int_0^\pi f(\theta)\cos(k\theta)\, d\theta,$$

where $\sum'$ means that the first summation term is divided by two. For a function on a general interval $[a,b]$ we apply the change of variables

$$\theta = \frac{x-a}{b-a}\pi.$$

2

The cosine expansion then becomes

$$f(x) = \sum_{k=0}^{\infty}{}' A_k \cos\left(k\frac{x-a}{b-a}\pi\right), \quad A_k = \frac{2}{b-a}\int_a^b f(x)\cos\left(k\frac{x-a}{b-a}\pi\right)dx.$$

We are interested in the price of a European option

$$V(t,S) = e^{-r(T-t)}\mathbb{E}\left[\Phi(S_T)|S_t = S\right] = e^{-r(T-t)}\int_{\mathbb{R}} \Phi(y) f(y|S_t = S)dy,$$

with $f(y|S_t)$ the probability density of S_T given $S_t = S$. Since this density decays to zero as $y \to \pm\infty$, we truncate the integration range

$$V(t,S) \approx e^{-r(T-t)}\int_a^b \Phi(y) f(y|S_t = S)dy.$$

Next we substitute the cosine expansion:

$$V(t,S) \approx e^{-r(T-t)}\int_a^b \Phi(y)\sum_{k=0}^{\infty}{}' A_k(S)\cos\left(k\frac{y-a}{b-a}\pi\right)dy =$$

$$e^{-r(T-t)}\sum_{k=0}^{\infty}{}' A_k(S)\int_a^b \Phi(y)\cos\left(k\frac{y-a}{b-a}\pi\right)dy = \frac{1}{2}(b-a)e^{-r(T-t)}\sum_{k=0}^{\infty}{}' A_k(S)V_k,$$

with

$$V_k = \frac{2}{b-a}\int_a^b \Phi(y)\cos\left(k\frac{y-a}{b-a}\pi\right)dy,$$

the cosine series coefficients of the pay-off function $\Phi(y)$. Since the coefficients are decaying, we truncate the sum and obtain

$$V(t,S) \approx \frac{1}{2}(b-a)e^{-r(T-t)}\sum_{k=0}^{N-1}{}' A_k(S)V_k.$$

Since the integrands in the Fourier transform (2.7) have to decay to 0, the truncated integral approximates the characteristic function well

$$\tilde{\phi}(\omega|S) = \int_a^b e^{i\omega x} f(x|S)dx \approx \int_{\mathbb{R}} e^{i\omega x} f(x|S)dx = \phi(\omega|S).$$

Then

$$A_k(S) = \frac{2}{b-a}\int_a^b f(x|S)\cos\left(k\frac{x-a}{b-a}\pi\right)dx = \frac{2}{b-a}\int_a^b f(x|S)\mathscr{R}\left(e^{ik\frac{x-a}{b-a}\pi}\right)dx =$$

$$\frac{2}{b-a}\mathscr{R}\left(\tilde{\phi}\left(\frac{k\pi}{b-a}\right)e^{-\frac{ik a\pi}{b-a}}\right) \approx \frac{2}{b-a}\mathscr{R}\left(\phi\left(\frac{k\pi}{b-a}\right)e^{-\frac{ik a\pi}{b-a}}\right).$$

Thus, the COS formula is

$$V(t,S) \approx e^{-r(T-t)}\sum_{k=0}^{N-1}{}'\mathscr{R}\left(\phi\left(\frac{k\pi}{b-a}\right)e^{-\frac{ik a\pi}{b-a}}\right)V_k.$$

If we denote $y = \log\left(\frac{S_T}{K}\right)$, then

$$\Phi(S_T) = \left(\alpha K\left(e^y - 1\right)\right)^+, \quad \alpha = \begin{cases} 1, & \text{for call,} \\ -1, & \text{for call.} \end{cases}$$

The cosine series coefficients of 1 are

$$\phi_k(c,d) = \int_c^d \cos\left(k\pi\frac{y-a}{b-a}\right)dy = \begin{cases} \left[\sin\left(k\pi\frac{d-a}{b-a}\right) - \sin\left(k\pi\frac{c-a}{b-a}\right)\right]\frac{b-a}{k\pi}, & k \neq 0, \\ (d-c), & k = 0. \end{cases}$$

The cosine series coefficients of e^y are

$$\chi_k(c,d) = \int_c^d e^y \cos\left(k\pi\frac{y-a}{b-a}\right)dy = \frac{1}{1+\left(\frac{k\pi}{b-a}\right)^2}\left[\cos\left(k\pi\frac{d-a}{b-a}\right)e^d - \cos\left(k\pi\frac{c-a}{b-a}\right)e^c + \right.$$

$$\left. + \frac{k\pi}{b-a}\sin\left(k\pi\frac{d-a}{b-a}\right)e^d - \frac{k\pi}{b-a}\sin\left(k\pi\frac{c-a}{b-a}\right)e^c\right].$$

Thus, the cosine series coefficients for the call and put option are

$$V_k^{call} = \frac{2}{b-a}K\left(\chi_k(0,b) - \psi_k(0,b)\right),$$

$$V_k^{put} = \frac{2}{b-a}K\left(-\chi_k(a,0) + \psi_k(a,0)\right).$$

For the truncation boundaries we choose $a = -L\sqrt{T}$ and $b = L\sqrt{T}$ for time to maturity T and truncation parameter L.

2.3. BLACK-SCHOLES MODEL

The most famous model for option pricing is the Black-Scholes model. In this section, we apply the two methods from Section 2.1 to the Black-Scholes model. In this model, the dynamics of the stock price S is a geometric Brownian motion:

$$dS_t = rS_t dt + \sigma S_t dW_t, \quad S_0 > 0, \tag{2.8}$$

with σ a parameter indicating how much the stock fluctuates, called the volatility. One of the reasons the model is so popular is that an exact solution of the option price exists.

Subsection 2.3.1 explains how to derive the option pricing PDE. Subsection 2.3.2 explains how to derive the conditional characteristic function of the (logarithm of the) stock price.

2.3.1. BLACK-SCHOLES PDE

We apply the theory from Subsection 2.1.1 to derive the option pricing PDE in the Black-Scholes model with dynamics (2.8). Let $f : \mathbb{R} \to \mathbb{R}$ be a $\mathscr{C}^2$-function. $\sigma S_t \in L^2([0,T] \times \Omega)$. Then by Itô's formula

$$df(S_t) = \frac{\partial f}{\partial S}(rS_t dt + \sigma S_t dW_t) + \frac{1}{2}\frac{\partial^2 f}{\partial S^2} d\langle S_t, S_t \rangle = \left(\frac{\partial f}{\partial S} rS_t + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S_t^2\right) dt + \text{martingale}.$$

So by the Feynman-Kac Theorem the price of a European option V with pay-off $\Phi(S_T)$ at maturity T satisfies the Black-Scholes PDE:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial S^2} + rS_t \frac{\partial V}{\partial S} - rV = 0, \quad V(T, S_T) = \Phi(S_T). \tag{2.9}$$

This PDE has an exact solution [24]:

$$\begin{aligned} V_{call}(t,S) &= S\mathcal{N}(d_1) - Ke^{-r\tau}\mathcal{N}(d_2), \\ V_{put}(t,S) &= Ke^{-r\tau}\mathcal{N}(-d_2) - S\mathcal{N}(-d_1). \end{aligned} \tag{2.10}$$

with $\mathcal{N}$ the standard normal distribution function, $\tau = T - t$ the time to maturity,

$$d_1 = \frac{\log\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}} \quad \text{and} \quad d_2 = d_1 - \sigma\sqrt{\tau},$$

The free boundary problem for the American option does not have an exact solution. We can approximate the solution using a finite difference method, stepping backwards through time and at each iteration taking the maximum of the continuation value and the intrinsic value.

2.3.2. BLACK-SCHOLES CHARACTERISTIC FUNCTION

We apply the theory from Subsection 2.1.2 to derive the characteristic function of the stock price in the Black-Scholes model (2.8). To apply the theory, we need an affine process. The S_t with dynamics (2.8) is not an affine process. To make the process affine, we transform to the log-domain. Define $X_t = \log(S_t)$. Then using Itô's formula

$$dX_t = \frac{1}{S_t}(rS_t dt + \sigma S_t dW_t) + \frac{1}{2}\frac{-1}{S_t^2}\sigma^2 S_t^2 dt = \left(r - \frac{1}{2}\sigma^2\right) dt + \sigma dW_t.$$

So the process X_t is affine with $b(t) = r - \frac{1}{2}\sigma^2$, $\beta = 0$, $a(t) = \sigma^2$ and $\alpha = 0$. The Ricatti equations therefore are

$$\begin{cases} \frac{\partial \psi}{\partial t} = 0, \\ \psi(0) = u, \\ \frac{\partial \phi}{\partial t} = (r - \sigma^2)\psi + \frac{1}{2}\sigma^2\psi^2, \\ \phi(0) = 0. \end{cases}$$

The solution to these equations is

$$\begin{cases} \psi(t,u) = u, \\ \phi(t,u) = (r - \sigma^2)ut + \frac{1}{2}\sigma^2 u^2 t \end{cases}$$

Thus, by Theorem 2.2 the conditional characteristic function of the log-price in the Black-Scholes model is equal to

$$\mathbb{E}\left[e^{iuX_T}|\mathscr{F}_t\right] = e^{iu(r-\frac{1}{2}\sigma^2)t-\frac{1}{2}\sigma^2u^2t+iuX_t}. \tag{2.11}$$

2.4. (ROUGH) HESTON MODEL

Despite its popularity, the Black-Scholes model is inconsistent with market data. Looking at the price of options in the market, we can use Equation (2.10) to compute the implied volatility σ resulting in this price. It turns out that this implied volatility is not a constant, but varies for different S and t. This results in the so-called implied volatility surface presented in Figure 2.1

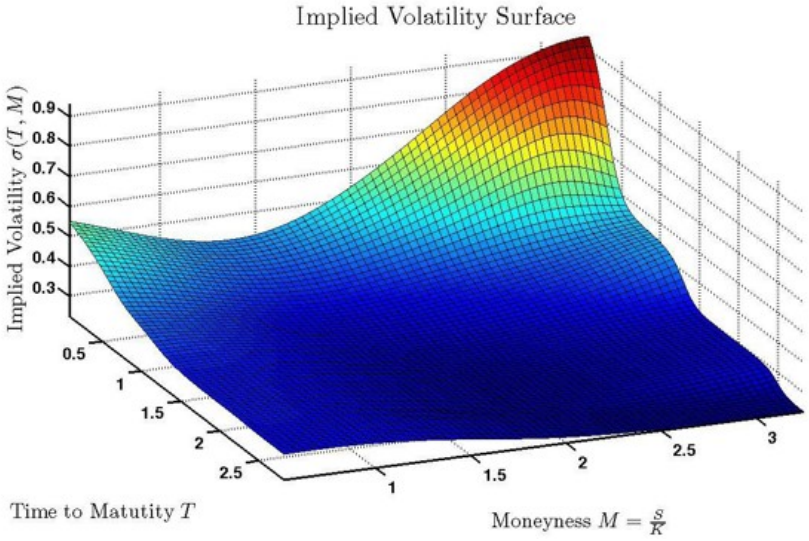


Figure 2.1: The implied volatility surface [4].

This problem can be solved by introducing stochastic volatility models where the volatility is not a constant but a stochastic process as well. The Heston model is a stochastic volatility model with dynamics

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{V_t}S_t dW_t, \quad S_0 > 0, \\ dV_t &= \kappa(\theta - V_t)dt + \eta\sqrt{V_t}dB_t, \quad V_0 > 0. \end{aligned}$$

Here V is the volatility process, B a Brownian motion correlated to W with correlation ρ, and $\kappa, \theta, \eta \in \mathbb{R}_+$. The Heston model can fit the implied volatility of an option at a specific time well. However, using estimated parameters to generate other option prices requires difficult modifications, such as making the parameters time dependent. Furthermore, the Heston model is not accurate for times close to maturity.

A rough fractional stochastic volatility model is able to reproduce the implied volatility surface with few constant parameters [7, 13]. In the rough Heston model, the Brownian motion in the volatility process becomes a fractional Brownian motion

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{V_t} S_t dW_t, \quad S_0 > 0, \\ V_t &= g(t) + \int_0^t K(t-s)\left(-\lambda V_s ds + \eta\sqrt{V_s} dB_s\right), \end{aligned} \tag{2.12}$$

with the fractional kernel $K(t) = \frac{t^{H-\frac{1}{2}}}{\Gamma\left(H+\frac{1}{2}\right)}$, $g(t) = V_0 + \int_0^t K(t-s)\theta(s)ds$, $\lambda \in \mathbb{R}_+$ and $H \in \left(0, \frac{1}{2}\right)$ the Hurst parameter. Although the process (S, V) mimics the real world data well, V is not a semimartingale. Moreover, due to the integral the process depends on all previous states instead of only the current state, meaning the process is not Markovian. Therefore, the Feynman-Kac Theorem cannot be applied anymore, giving trouble pricing options in the filtration generated by (W, B).

2.5. LIFTED HESTON MODEL

The lifted Heston model is a Markovian approximation of the rough Heston model [2, 3]. The lifted Heston model converges to the rough Heston model, and option prices in the lifted Heston model converge to the option prices in the rough Heston model. The essence of the model is rewriting the kernel as a Laplace transform and approximating this kernel with a discretization.

Define $c_H = \frac{1}{\Gamma\left(H+\frac{1}{2}\right)\Gamma\left(\frac{1}{2}-H\right)}$ and measure $\mu(dx) = c_H x^{-H-\frac{1}{2}} dx$. Then

$$\int_0^\infty e^{-xt}\mu(dx) = c_H \int_0^\infty e^{-xt} x^{-H-\frac{1}{2}} dx = c_H \int_0^\infty e^{-u}\left(\frac{u}{t}\right)^{-H-\frac{1}{2}} \frac{1}{t} du =$$

$$c_H t^{H-\frac{1}{2}} \int_0^\infty e^{-u} u^{-H-\frac{1}{2}} du = c_H t^{H-\frac{1}{2}} \Gamma\left(\frac{1}{2} - H\right) = K(t).$$

So $K(t)$ is written as the Laplace transform of μ. In the lifted Heston model, we replace μ by a finite sum of Dirac measures. The approximate kernel is

$$K^n(t) = \sum_{i=1}^n c_i^n e^{-\gamma_i t},$$

with

$$c_i^n = \left(r_n^{1-\alpha} - 1\right) \frac{r_n^{(\alpha-1)\left(1+\frac{n}{2}\right)} r_n^{(1-\alpha)i}}{\Gamma(\alpha)\Gamma(2-\alpha)}, \quad \gamma_i^n = \frac{1-\alpha}{2-\alpha} \frac{r_n^{2-\alpha}-1}{r_n^{1-\alpha}-1} r_n^{i-1-\frac{n}{2}}, \quad \text{and} \quad \alpha = H + \frac{1}{2}.$$

Theoretical results suggest taking $r_n = 1 + 10n^{-0.9}$. Numerical results suggest that $n = 20$ and $r_n = 2.5$ gives a good approximation to the rough Heston model [2]. Replacing $K(t)$ by $K^n(t)$ in (2.12) gives

$$\begin{aligned} dS_t^n &= rS_t^n dt + \sqrt{V_t^n} S_t^n dW_t, \quad S_0 > 0, \\ V_t^n &= g^n(t) + \int_0^t \sum_{i=1}^n c_i e^{-\gamma_i(t-s)} \left(-\lambda V_s^n ds + \eta\sqrt{V_s^n} dB_s\right) = g^n(t) + \sum_{i=1}^n c_i^n V_t^{i,n}, \end{aligned}$$

with

$$g^n(t) = V_0 + \sum_{i=1}^{n} c_i^n \int_0^t e^{-\gamma_i^n (t-s)} \theta(s) ds, \quad \text{and} \quad V_t^{i,n} = \int_0^t e^{-\gamma_i (t-s)} \left(-\lambda V_s ds + \eta \sqrt{V_s} dB_s \right).$$

The last term can be recognized as the solution of an Ornstein-Uhlenbeck process [12] with dynamics:

$$dV_t^{n,i} = -\left(\gamma_i^n V_t^{n,i} + \lambda V_t^n \right) dt + \eta \sqrt{V_t^n} dB_t, \quad V_0^{n,i} = 0.$$

Indeed, applying Itô's Formula to the function $f\left(V_t^{n,i}, t\right) = V_t^{n,i} e^{\gamma_i^n t}$ gives

$$df = \frac{\partial f}{\partial t} dt + \frac{\partial f}{\partial V_t^{n,i}} dV_t^{n,i} + \frac{1}{2} \frac{\partial^2 f}{\partial \left(V_t^{n,i}\right)^2} d\left\langle V_t^{n,i}, V_t^{n,i} \right\rangle = \gamma_i^n V_t^{n,i} e^{\gamma_i^n t} dt + e^{\gamma_i^n t} dV_t^{n,i} =$$

$$e^{\gamma_i^n t} \left(-\lambda V_t dt + \eta \sqrt{V_t} dW_t \right).$$

Integrating from 0 to t gives

$$V_t^{n,i} e^{\gamma_i^n t} - V_0^{n,i} = \int_0^t e^{\gamma_i^n s} \left(-\lambda V_s ds + \eta \sqrt{V_s} dB_s \right) \Longrightarrow$$

$$V_t^{n,i} = V_0^{n,i} e^{-\gamma_i^n t} + \int_0^t e^{-\gamma_i^n (t-s)} \left(-\lambda V_s ds + \eta \sqrt{V_s} dB_s \right).$$

We choose $\theta(t) = \theta\lambda$ as this makes computations easier without changing the essence of the problem [2].

3

METHODOLOGY

This chapter explains the methods to price options in the lifted Heston model. Section 3.1 derives the partial differential equation (PDE) of the option price in the lifted Heston model. Section 3.2 derives the characteristic function of the lifted Heston model. Section 3.3 explains the necessary variational calculus related to the pricing PDE. Section 3.4 explains the new splitting method to rewrite PDEs. Section 3.5 explains the neural network algorithms to solve the PDEs.

3.1. LIFTED HESTON PDE

In this section, we derive the option pricing PDE in the lifted Heston model. We apply the theory for deriving an option pricing PDE from Subsection 2.1.1 to the lifted Heston model from Section 2.5. Consider the lifted Heston dynamics that were derived in the previous chapter:

$$\begin{cases} dS_t^n = rS_t^n\,dt + \sqrt{V_t^n}S_t^n\,dW_t, \quad S_0 > 0, \\ V_t^n = g^n(t) + \sum_{i=1}^n c_i^n V_t^{i,n}, \\ dV_t^{n,i} = -\left(\gamma_i^n V_t^{n,i} + \lambda V_t^n\right)dt + \eta\sqrt{V_t^n}\,dB_t, \quad V_0^{n,i} = 0. \end{cases} \tag{3.1}$$

Let $f\left(S_t^n, V_t^{n,1}, ..., V_t^{n,n}\right) : \mathbb{R}^{n+1} \to \mathbb{R}$ be a $\mathscr{C}^2$-function. Denote $f_S = \frac{\partial f}{\partial S^n}$ and $f_{V^i} = \frac{\partial f}{\partial V^{n,i}}$. Then Itô's formula gives

$$df\left(S_t^n, V_t^{n,1}, ..., V_t^{n,n}\right) = f_S dS_t^n + \sum_{i=1}^n f_{V^i} dV_t^{n,i} + \frac{1}{2} f_{SS} d\left\langle S_t^n, S_t^n\right\rangle + \sum_{i=1}^n f_{SV^i} d\left\langle S_t^n, V_t^{n,i}\right\rangle +$$

$$+\frac{1}{2}\sum_{i,j=1}^n f_{V^iV^j} d\left\langle V_t^{n,i}, V_t^{n,i}\right\rangle = f_S r S_t^n dt - \sum_{i=1}^n f_{V^i}\left(\gamma_i^n V_t^{n,i} + \lambda V_t^n\right)dt + \frac{1}{2} f_{SS} V_t^n \left(S_t^n\right)^2 dt +$$

$$+\sum_{i=1}^n f_{SV^i}\eta\rho V_t^n S_t^n dt + \frac{1}{2}\sum_{i,j=1}^n f_{V^iV^j}\eta^2 V_t^n dt + \text{martingale}.$$

Then the Feynman-Kac Theorem gives that the price of a derivative V with payoff $\Phi(S_T)$ satisfies

$$\frac{\partial V}{\partial t} - \mathscr{A} V = ru, \quad V(T) = \Phi(S_T),$$

with

$$\mathscr{A} f = -rSf_S + \sum_{i=1}^{n} \left(\gamma_i^n V^i + \lambda V_t^n\right) f_{V^i} - \frac{1}{2} V_t^n S^2 f_{SS} - \eta\rho V_t^n S \sum_{i=1}^{n} f_{SV^i} - \frac{\eta^2}{2} V_t^n \sum_{i,j=1}^{n} f_{V^i V^j}.$$

Switching to time to maturity $\tau = T - t$ gives

$$\frac{\partial V}{\partial \tau} + \mathscr{A} V + ru = 0, \quad u(0) = \Phi(S_0), \tag{3.2}$$

In contrast to the Black-Scholes PDE (2.9), this PDE does not have an exact solution. Furthermore, for $n = 20$ the problem has 21 dimensions, making a finite difference method infeasible. We will solve this PDE using neural network algorithms, as is explained in Section 3.5.

3.2. LIFTED HESTON CHARACTERISTIC FUNCTION

In this section, we derive the characteristic function in the lifted Heston model. We apply the method to derive a characteristic function from Subsection 2.1.2 to the lifted Heston model (3.1). To apply the method from Subsection 2.1.2, we need an affine model. The S_t with the lifted Heston dynamics (3.1) is not affine. As in the Black-Scholes case in Subsection 2.3.2 we transform to the logarithm of the stock price. Defining $X_t^n = \log(S_t^n)$ transforms (3.1) to

$$\begin{cases} dX_t^n = \left(r - \frac{1}{2} V_t^n\right) dt + \sqrt{V_t^n} dW_t, \quad X_0 \in \mathbb{R}, \\ V_t^n = g^n(t) + \sum_{i=1}^n c_i^n V_t^{i,n}, \\ dV_t^{n,i} = -\left(\gamma_i^n V_t^{n,i} + \lambda V_t^n\right) dt + \eta \sqrt{V_t^n} dB_t, \quad V_0^{n,i} = 0. \end{cases} \tag{3.3}$$

Since W_t and B_t are two Brownian motions with correlation ρ, we can write $W_t = \rho B_t + \sqrt{1-\rho^2} dB_t^\perp$, with $B_t^\perp$ a Brownian motion independent of B_t. In matrix notation, (3.3) becomes

$$d\mathbf{X}_t = d \begin{bmatrix} X_t^n \\ V_t^{n,1} \\ \vdots \\ V_t^{n,n} \end{bmatrix} = \beta(\mathbf{X}_t) dt + \sigma_1(\mathbf{X}_t) dB_t + \sigma_2(\mathbf{X}_t) dB_t^\perp,$$

with

$$\beta(\mathbf{X}_t) = \begin{bmatrix} r - \frac{1}{2}\left(g^n(t) + \sum_{k=1}^n c_k^n V_t^{k,n}\right), \\ -\gamma_1^n V_t^{n,1} - \lambda\left(g^n(t) + \sum_{k=1}^n c_k^n V_t^{k,n}\right) \\ \vdots \\ -\gamma_n^n V_t^{n,n} - \lambda\left(g^n(t) + \sum_{k=1}^n c_k^n V_t^{k,n}\right) \end{bmatrix},$$

$$\sigma_1(\mathbf{X}_t) = \begin{bmatrix} \rho\sqrt{g^n(t) + \sum_{i=1}^n c_i^n V_t^{i,n}} \\ \eta\sqrt{g^n(t) + \sum_{i=1}^n c_i^n V_t^{i,n}} \\ \vdots \\ \eta\sqrt{g^n(t) + \sum_{i=1}^n c_i^n V_t^{i,n}} \end{bmatrix}, \quad \text{and}$$

$$\sigma_2(\mathbf{X}_t) = \begin{bmatrix} \sqrt{1-\rho^2}\sqrt{g^n(t) + \sum_{i=1}^n c_i^n V_t^{i,n}} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Then

$$\beta(\mathbf{X}_t) = b(t) + \beta_0 X_t^n + \beta_1 V_t^{n,1} + ... + \beta_n V_t^{n,n},$$
$$\sigma_1(\mathbf{X}_t)\sigma_1(\mathbf{X}_t)^T + \sigma_2(\mathbf{X}_t)\sigma_2(\mathbf{X}_t)^T = a(t) + \alpha_0 X_t^n + \alpha_1 V_t^{n,1} + ... + \alpha_n V_t^{n,n},$$

with

$$b(t) = \begin{bmatrix} r - \frac{1}{2}g^n(t) \\ -\lambda g^n(t) \\ \vdots \\ -\lambda g^n(t) \end{bmatrix}, \quad \beta_0 = \mathbf{0}, \quad \beta_1 = \begin{bmatrix} -\frac{1}{2}c_1^n \\ -\gamma_1^n - \lambda c_1^n \\ -\lambda c_1^n \\ \vdots \\ -\lambda c_1^n \end{bmatrix}, ..., \beta_n = \begin{bmatrix} -\frac{1}{2}c_n^n \\ -\lambda c_n^n \\ \vdots \\ -\lambda c_n^n \\ -\gamma_n^n - \lambda c_n^n \end{bmatrix},$$

$$a(t) = g^n(t)\Sigma, \quad \alpha_0 = \mathbf{0} \quad \text{and} \quad \alpha_i = c_i^n\Sigma \quad \text{for} \quad i = 1, ..., N,$$

where

$$\Sigma = \begin{bmatrix} 1 & \eta\rho & ... & \eta\rho \\ \eta\rho & \eta^2 & ... & \eta^2 \\ \vdots & \vdots & \ddots & \vdots \\ \eta\rho & \eta^2 & ... & \eta^2 \end{bmatrix}.$$

So the process X_t is affine. Then by Theorem 2.2 the moment generating function of the lifted Heston models is

$$\mathbb{E}\left[e^{\langle \mathbf{u}, \mathbf{X}_T\rangle} | \mathscr{F}_t\right] = e^{\phi(T-t,\mathbf{u}) + \langle \psi(T-t,\mathbf{u}), \mathbf{X}_t\rangle},$$

where ϕ and ψ satisfy the Ricatti equations. For ψ_0

$$\frac{\partial \psi_0(t,\mathbf{u})}{\partial t} = 0, \quad \psi_0(0,\mathbf{u}) = u_0 \implies \psi_0(t,u) = u_0.$$

Then for $i = 1, ..., N$

$$\frac{\partial \psi_i(t,\mathbf{u})}{\partial t} = -\gamma_i^n \psi_i(t,\mathbf{u}) - \lambda c_i^n \sum_{j=1}^{n} \psi_j(t,\mathbf{u}) - \frac{1}{2} c_i^n u_0 + c_i^n \left(\frac{1}{2} u_0^2 + u_0 \eta \rho \sum_{j=1}^{n} \psi_j(t,\mathbf{u}) + \right.$$

$$\left. + \frac{\eta^2}{2} \left(\sum_{j=1}^{n} \psi_j(t,\mathbf{u}) \right)^2 \right) = -\gamma_i^n \psi_i(t,\mathbf{u}) + c_i^n F\left(u_0, \sum_{j=1}^{n} \psi_j(t,\mathbf{u}) \right), \quad \psi_i(0,\mathbf{u}) = u_i,$$

with

$$F(u,v) = \frac{1}{2}\left(u^2 - u\right) + \left(u\eta\rho - \lambda\right) v + \frac{\eta^2}{2} v^2.$$

Then finally

$$\frac{\partial \phi(t,\mathbf{u})}{\partial t} = r u_0 + g^n(T-t) F\left(u_0, \sum_{j=1}^{n} \psi_j(t,\mathbf{u}) \right), \quad \phi(0,\mathbf{u}) = 0.$$

To compute derivative prices, we solve these ordinary differential equations with the Runge-Kutta method and then obtain the price from the characteristic function by the COS method as in Section 2.2.

3.3. CALCULUS OF VARIATIONS

In this section, we explain the variational calculus which will be used in the Time Deep Nitsche Method (TDNM) in Subsection 3.5.2. Calculus of variations is a technique to solve nonlinear problems by rewriting them as minimization problems [9]. Let $u(x_1, ..., x_d) : \Omega \to \mathbb{R}$ be a sufficiently smooth function and denote $u_i = \frac{\partial u}{\partial x_i}$. Suppose we have the PDE

$$\begin{cases} \mathscr{L} u = 0, & \text{in } \Omega, \\ u = g, & \text{on } \partial\Omega \end{cases} \tag{3.4}$$

We try to find an energy functional I such that $\mathscr{L} u = I'[u]$. Then PDE (3.4) rewrites to

$$I'[u] = 0,$$

which is finding critical points of I. Now assume the energy functional has the form

$$I[u] = \int_\Omega L(u_1, ..., u_d, u, x_1, ..., x_d)\, dx.$$

The operator $L : \mathbb{R}^n \times \mathbb{R} \times \Omega \to \mathbb{R}$ is the Lagrangian. Suppose u minimizes I and let v be a smooth function that is 0 on $\partial\Omega$. Consider the function

$$i(\tau) = I(u + \tau v) = \int_\Omega L(\nabla u + \tau \nabla v, u + \tau v, \mathbf{x}) dx, \quad \tau \in \mathbb{R}.$$

Since u minimizes I and $u + \tau v = g$ on $\partial\Omega$, $\tau = 0$ minimizes i. So

$$0 = i'(0) = \left[\int_\Omega \sum_{i=1}^{n} \frac{\partial L}{\partial u_i} (\nabla u + \tau \nabla v, u + \tau v, \mathbf{x}) \frac{\partial v}{\partial x_i} + \frac{\partial L}{\partial u} (\nabla u + \tau \nabla v, u + \tau v, \mathbf{x})\, v dx \right]_{\tau=0}$$

$$= \int_\Omega \sum_{i=1}^{n} \frac{\partial L}{\partial u_i} (\nabla u, u, \mathbf{x}) \frac{\partial v}{\partial x_i} + \frac{\partial L}{\partial u} (\nabla u, u, \mathbf{x})\, v dx.$$

Using integration by parts gives

$$0 = \int_\Omega \left[-\sum_{i=1}^{n} \frac{\partial}{\partial x_i} \left(\frac{\partial L}{\partial u_i} (\nabla u, u, \mathbf{x}) \right) + \frac{\partial L}{\partial u} (\nabla u, u, \mathbf{x}) \right] v dx.$$

Since this equality must hold for all v, u satisfies the PDE

$$-\sum_{i=1}^{d} \frac{\partial}{\partial x_i} \left(\frac{\partial L}{\partial u_i} \right) + \frac{\partial L}{\partial u} = 0.$$

This equation is the Euler-Lagrange equation. Suppose $\mathscr{L}$ is of the form

$$\mathscr{L} u = -\nabla \cdot (A \nabla u) + r u. \tag{3.5}$$

Then defining

$$L = \frac{1}{2} \left((\nabla u)^T A \nabla u + r u^2 \right)$$

gives

$$-\sum_{i=1}^{d} \frac{\partial}{\partial x_i} \left(\frac{\partial L}{\partial u_i} \right) + \frac{\partial L}{\partial u} = \mathscr{L} u.$$

However, the Black-Scholes PDE (2.9) and the Lifted Heston PDE (3.2) do not have the form (3.5). It is possible to transform the Black-Scholes PDE to the heat equation with the following theorem [8].

Theorem 3.1. *Define* $y = \log\left(\frac{S}{K}\right)$, $\tau = \frac{1}{2}\sigma^2 (T-t)$, $q = \frac{2r}{\sigma^2}$,

$$C = \frac{1}{K} \exp\left(\frac{1}{2}(q-1)y + \left(\frac{1}{4}(q-1)^2 + q \right) \tau \right) > 0,$$

$$W(\tau, y) = C V(t, S) \quad \textit{and} \quad g(\tau, y) = C \Phi(K e^y).$$

V satisfies the Black-Scholes equation (2.9) if and only if W satisfies

$$\frac{\partial}{\partial \tau} W(\tau, y) - \frac{\partial^2}{\partial y^2} W(\tau, y) = 0.$$

Furthermore, V satisfies the free boundary problem (2.5) for the American option in the Black-Scholes case if and only if W satisfies

$$\begin{cases} \left(\frac{\partial}{\partial \tau} W(\tau, y) - \frac{\partial^2}{\partial y^2} W(\tau, y) \right) \left(W(\tau, y) - g(\tau, y) \right) = 0, \quad \frac{\partial}{\partial \tau} W(\tau, y) - \frac{\partial^2}{\partial y^2} W(\tau, y) \geq 0, \\ W(\tau, y) \geq g(\tau, y), \quad W(0, y) = g(0, y). \end{cases}$$

Proof.

$$\frac{\partial}{\partial \tau} W(\tau, y) - \frac{\partial^2}{\partial y^2} W(\tau, y) = C \left(\left(\frac{1}{4}(q-1)^2 + q \right) V(t, S) + \frac{\partial V(t, S)}{\partial \tau} \right) -$$

$$\frac{1}{K} \frac{\partial}{\partial y} \left(\exp\left(\frac{1}{2}(q-1)y + \left(\frac{1}{4}(q-1)^2 + q \right) \tau \right) \left(\frac{1}{2}(q-1) V(t, S) + \frac{\partial V(t, S)}{\partial y} \right) \right) =$$

$$C\left(\left(\frac{1}{4}(q-1)^2+q\right)V(t,S)+\frac{\partial V(t,S)}{\partial \tau}-\frac{1}{4}(q-1)^2V(t,S)-(q-1)\frac{\partial V(t,S)}{\partial y}-\frac{\partial^2 V(t,S)}{\partial y^2}\right).$$

The term without C is equal to

$$qV(t,S)+\frac{\partial V(t,S)}{\partial t}\frac{\partial t}{\partial \tau}-(q-1)\frac{\partial V(t,S)}{\partial S}\frac{\partial S}{\partial y}-\frac{\partial^2 V(t,S)}{\partial S^2}\left(\frac{\partial S}{\partial y}\right)^2-\frac{\partial V(t,S)}{\partial S}\frac{\partial^2 S}{\partial y^2}=$$

$$qV(t,S)-\frac{\partial V(t,S)}{\partial t}\frac{2}{\sigma^2}-q\frac{\partial V(t,S)}{\partial S}Ke^y-\frac{\partial^2 V(t,S)}{\partial S^2}K^2e^{2y}=$$

$$-\frac{2}{\sigma^2}\left(\frac{\partial V(t,S)}{\partial t}+rS\frac{\partial V(t,S)}{\partial S}+\frac{1}{2}\sigma^2S^2\frac{\partial^2 V(t,S)}{\partial S^2}-rV(t,S)\right).$$

Since $-\frac{2C}{K\sigma^2}\neq 0$, the first part of the Theorem holds. More specifically, $-\frac{2C}{K\sigma^2}<0$, so

$$\frac{\partial}{\partial \tau}W(\tau,y)-\frac{\partial^2}{\partial y^2}W(\tau,y)\geq 0 \iff \frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV\leq 0.$$

The rest of the second part follows by multiplying each side of the (in)equality sign by C. □

If we for instance have a European call option, the initial condition becomes:

$$W(0,y)=\frac{1}{K}\exp\left(\frac{1}{2}(q-1)y\right)V(T,Ke^y)=\exp\left(\frac{1}{2}(q-1)y\right)(e^y-1)^+$$
$$=\left(\exp\left(\frac{1}{2}(q+1)y\right)-\exp\left(\frac{1}{2}(q-1)y.\right)\right)^+.$$

In the heat equation $\mathscr{L}u=-\frac{\partial^2 u}{\partial y^2}$. This operator is of the same type as (3.5) with $A=1\in\mathbb{R}$ and $r=0$. So the Lagrangian corresponding to the heat equation is $L=\frac{1}{2}|\frac{\partial u}{\partial y}|^2$.

For the lifted Heston model, such a transformation method is not possible. We therefore use another method, as explained in the next section.

3.4. SPLITTING METHOD

In this section, we explain how to apply the calculus of variations technique to a more general operator. This allows us to apply the TDNM to more general problems than in [14]. Suppose we have the general operator

$$\mathscr{A}=\sum_{i=1}^{d}\beta^i u_i-\sum_{i,j=1}^{d}a^{ij}u_{ij}+ru. \tag{3.6}$$

For most values of β and a this operator cannot be rewritten in the form (3.5). Therefore, we split the operator into two parts. A symmetric part that can be written as equation (3.5) and an asymmetric part containing all other terms of the operator. Using the chain

rule, we can rewrite (3.6) to

$$\begin{aligned}\mathscr{A} &= \sum_{i=1}^{d} \beta^i u_i - \sum_{i,j=1}^{d} \frac{\partial}{\partial x_j}\left(a^{ij} u_i\right) + \sum_{i,j=1}^{d} \frac{\partial a^{ij}}{\partial x_j} u_i + r u \\ &= \sum_{i=1}^{d} \left(\beta^i + \sum_{j=1}^{d} \frac{\partial a^{ij}}{\partial x_j}\right) u_i - \sum_{i,j=1}^{d} \frac{\partial}{\partial x_j}\left(a^{ij} u_i\right) + r u.\end{aligned}$$

Define $b^i = \beta^i + \sum_{j=1}^{d} \frac{\partial a^{ij}}{\partial x_j}$. Then

$$\mathscr{A} = \mathscr{L} u + F(u)$$

with $\mathscr{L}$ as in (3.5) and

$$F(u) = \sum_{i=1}^{d} b^i u_i = \mathbf{b} \cdot \nabla u,$$

for

$$A = \begin{bmatrix} a^{11} & a^{21} & \dots & a^{d1} \\ a^{12} & a^{22} & \dots & a^{1d} \\ \vdots & \vdots & \ddots & \vdots \\ a^{1d} & a^{2d} & \dots & a^{dd} \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b^1 \\ b^2 \\ \vdots \\ b^d \end{bmatrix}.$$

We apply this splitting method to the Black-Scholes PDE (2.9) in Subsection 3.4.1 and to the Lifted Heston PDE (3.2) in Subsection 3.4.2.

3.4.1. BLACK-SCHOLES MODEL

First, we apply the splitting method to the Black-Scholes PDE (2.9). Switching to time to maturity $\tau = T - t$, the Black-Scholes PDE is

$$V_\tau - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rS\frac{\partial V}{\partial S} + rV = 0, \quad V(0,S) = \Phi(S).$$

Applying the chain rule gives

$$0 = V_\tau - \frac{\partial}{\partial S}\left(\frac{1}{2}\sigma^2 S^2 \frac{\partial V}{\partial S}\right) + \sigma^2 S \frac{\partial V}{\partial S} - rS\frac{\partial V}{\partial S} + rV = V_\tau + \mathscr{L} V + F(V),$$

with

$$F = (\sigma^2 - r) S \frac{\partial V}{\partial S} \tag{3.7}$$

and

$$\mathscr{L} V = -\frac{\partial}{\partial S}\left(\frac{1}{2}\sigma^2 S^2 \frac{\partial V}{\partial S}\right) + rV.$$

The corresponding Lagrangian is

$$L = \frac{1}{4}\sigma^2 S^2 \left(\frac{\partial V}{\partial S}\right)^2 + \frac{1}{2} r V^2. \tag{3.8}$$

In Section 4.2 a comparison of the splitting method and the transformation method based on Theorem 3.1 is presented.

3.4.2. LIFTED HESTON MODEL

Second, we apply the splitting method to the lifted Heston PDE (3.2). Applying the chain rule gives

$$0 = u_\tau - rSu_S + \sum_{i=1}^{n}\left(\gamma_i^n V^i + \lambda V_t^n\right)u_{V^i} - \frac{1}{2}V_t^n S^2 u_{SS} - \eta\rho V_t^n S\sum_{i=1}^{n} u_{SV^i} - \frac{\eta^2}{2}V_t^n \sum_{i,j=1}^{n} u_{V^iV^j} +$$

$$+ ru = u_\tau - rSu_S + \sum_{i=1}^{n}\left(\gamma_i^n V^i + \lambda V_t^n\right)u_{V^i} - \left(\frac{1}{2}V_t^n S^2 u_S\right)_S + V_t^n S u_S - \frac{1}{2}\sum_{i=1}^{n}\left(\eta\rho V_t^n S u_{V^i}\right)_S +$$

$$+\frac{1}{2}\eta\rho V_t^n \sum_{i=1}^{n} u_{V^i} - \frac{1}{2}\sum_{i=1}^{n}\left(\eta\rho V_t^n S u_S\right)_{V^i} + \frac{1}{2}\eta\rho S\sum_{i=1}^{n} c_i^n u_S - \sum_{i,j=1}^{n}\left(\frac{\eta^2}{2}V_t^n u_{V^i}\right)_{V^j} +$$

$$+\frac{\eta^2}{2}\sum_{i,j=1}^{n} c_j^n u_{V^i} + ru = u_\tau + \left(V_t^n S - rS + \frac{1}{2}\eta\rho S\sum_{i=1}^{n} c_i^n\right)u_S + \sum_{i=1}^{n}\left(\gamma_i^n V^i + \lambda V_t^n + \frac{1}{2}\eta\rho V_t^n +\right.$$

$$\left.+\frac{\eta^2}{2}\sum_{j=1}^{n} c_j^n\right)u_{V^i} - \left(\frac{1}{2}V_t^n S^2 u_S\right)_S - \frac{1}{2}\sum_{i=1}^{n}\left(\eta\rho V_t^n S u_{V^i}\right)_S - \frac{1}{2}\sum_{i=1}^{n}\left(\eta\rho V_t^n S u_S\right)_{V^i} -$$

$$-\sum_{i,j=1}^{n}\left(\frac{\eta^2}{2}V_t^n u_{V^i}\right)_{V^j} + ru = u_\tau + \mathscr{L}V + F(V),$$

with

$$\mathscr{L}V = -\left(a^{00}u_S\right)_S - \sum_{i=1}^{n}\left(a^{0i}u_{V^i}\right)_S - \sum_{i=1}^{n}\left(a^{i0}u_S\right)_{V^i} - \sum_{i,j=1}^{n}\left(a^{ij}u_{V^i}\right)_{V^j} + ru,$$

$$F(u) = b^0 u_S + \sum_{i=1}^{n} b^i u_{V^i}. \tag{3.9}$$

and

$$\begin{cases} a^{00} = \frac{1}{2}V_t^n S^2, \\ a^{i0} = a^{0i} = \frac{1}{2}\eta\rho V_t^n S, \\ a^{ij} = \frac{\eta^2}{2}V_t^n, \\ b^0 = \left(V_t^n - r + \frac{1}{2}\eta\rho\sum_{i=1}^{n} c_i^n\right)S, \\ b^i = \left(\gamma_i^n V^{n,i} + \lambda V_t^n + \frac{1}{2}\eta\rho V_t^n + \frac{\eta^2}{2}\sum_{j=1}^{n} c_j^n\right). \end{cases}$$

The corresponding Lagrangian is

$$L = \frac{1}{2}\left(a^{00}u_S^2 + 2\sum_{i=1}^{n} a^{0i}u_{V^i}u_S + \sum_{i,j=1}^{n} a^{ij}u_{V^i}u_{V^j} + ru^2\right). \tag{3.10}$$

3.5. NEURAL NETWORK ALGORITHMS

This section explains the neural network algorithms that we use to solve the option pricing PDEs. A neural network is a graph consisting of layers, a collection of vertices, that are chained together like neurons in a brain, hence the name. A schematic overview of a neural network is in Figure 3.1. The input travels through the layers and results in the output. Each edge has a weight, making each vertex a weighted sum of all vertices from the previous layer. Furthermore, each layer has an activation function which allows nonlinearity. The number of layers of the graph is the *depth*. The number of neurons per layer is the *width*. The parameters of the network are all weights and biases and denoted by θ. The activation function is a hyperparameter.

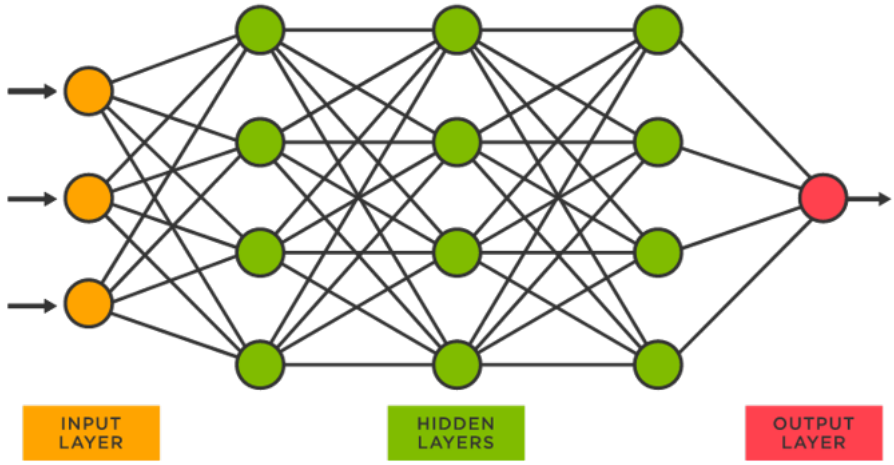


Figure 3.1: Schematic overview of a neural network [28].

The aim is to train the neural network such that for each set of input variables $\mathbf{x}$, the network returns the correct output. Training the neural network means we update the parameters θ such that we minimize a loss function $L(\theta;\mathbf{x})$ determining the performance of the network.

The most used approach for minimizing a loss function is stochastic gradient descent [5]. We start with an initial guess θ_0 and compute the gradient of the loss function at that point. This gives the direction with the largest increase in loss function. Gradient descent updates θ by moving away from that direction: $\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta L(\theta;\mathbf{x})$. The hyperparameter α_n is the step-size of our update, called the *learning rate*. Often these loss functions involve an integral, making it computationally expensive to compute. Stochastic gradient descent means randomly sampling points $\mathbf{x}_i$ from the domain and applying a Monte Carlo approximation of L.

An overview of the neural network algorithm is in Algorithm 1. In the third step, we take random points from the domain instead of grid points that finite difference methods use. Therefore, neural network algorithms do not suffer from the curse of dimensionality like finite difference methods. In this research, we use two different neural

Algorithm 1 Neural network algorithm

1: Initialize θ_0.
2: **for** each sampling stage **do**
3: Generate random points $\mathbf{x}^i$ for training.
4: Calculate the cost functional $\tilde{L}(\theta_n;\mathbf{x}^i)$ for the sampled points.
5: Take a descent step $\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta L(\theta_n;\mathbf{x}^i)$.
6: **end for**

network-algorithms. First is the Deep Galerkin Method (DGM) explained in Subsection 3.5.1. Second is the TDNM explained in Subsection 3.5.2. The implementation details of the neural network-algorithms are in Subsection 3.5.3

3.5.1. DEEP GALERKIN METHOD

The first neural network algorithm to solve PDEs we consider is the DGM [27]. Suppose we have the following PDE:

$$\begin{cases} (\partial_t + \mathscr{A})u(t,\mathbf{X}) = 0, \quad (t,\mathbf{X}) \in [0,T]\times\Omega, \\ u(0,\mathbf{X}) = \Phi(\mathbf{X}), \quad \mathbf{X}\in\Omega, \\ u(t,\mathbf{X}) = g(t,\mathbf{X}), \quad (t,\mathbf{X}) \in [0,T]\times\partial\Omega. \end{cases} \tag{3.11}$$

The goal is to approximate the function u with a neural network $f(t,\mathbf{X};\theta)$ with parameter set θ. The DGM approach is to define a loss function measuring how well the neural network satisfies PDE (3.11):

$$\begin{aligned} L(\theta;t,x,w,\tau,z) = &\left\| (\partial_t + \mathscr{A}) f(t,x;\theta) \right\|^2_{[0,T]\times\Omega} + \left\| f(0,w;\theta) - \Phi(w) \right\|^2_{\Omega} \\ &+ \left\| f(\tau,z;\theta) - g(\tau,z) \right\|^2_{[0,T]\times\partial\Omega}. \end{aligned}$$

For a u satisfying the PDE (3.11) the loss function is 0. Applying a Monte Carlo approximation to the integrals, we define the loss functional

$$\begin{aligned} \tilde{L}(\theta;t_i,x_i,w_i,\tau_i,z_i) = &\frac{1}{N_I}\sum_{i=1}^{N_I}\left((\partial_t + \mathscr{A})f(t_i,x_i;\theta)\right)^2 + \frac{1}{N_T}\sum_{i=1}^{N_T}\left(f(0,w_i;\theta) - \Phi(w_i)\right)^2 \\ &+ \frac{1}{N_B}\sum_{i=1}^{N_B}\left(f(\tau_i,z_i;\theta) - g(\tau_i,z_i)\right)^2, \end{aligned} \tag{3.12}$$

with N_I the number of samples on the interior domain, N_T the number of samples on the terminal domain and N_B the number of samples from the boundary domain. The DGM algorithm is to apply Algorithm 1 with loss functional (3.12).

3.5.2. TIME DEEP NITSCHE METHOD

The second neural network algorithm to solve PDEs we consider is the TDNM [14]. We develop a new version of the TDNM by combining it with the splitting method from Section 3.4 so that it can solve option pricing PDEs. Instead of using time as an input variable for the neural network, we train a neural network for each time step. The neural

network for the previous time step is then a good initial guess for the neural network in the current time step, reducing the number of training stages necessary. In the first step the neural network then approximates the initial condition which is a known function not needing differentiation of the network.

Again, consider PDE (3.11). Using the splitting method, we write $\mathscr{A}u = \mathscr{L}u + F(u)$, with $\mathscr{L}u = -\nabla\cdot(A\nabla u) + ru$ and $F(u)$ the rest term. Divide $[0,T]$ in N_t intervals $(t_{k-1}, t_k]$ with $h = t_k - t_{k-1} = \frac{1}{N_t}$. We seek approximations $f^k(\mathbf{X};\theta)$ such that

$$\frac{f^k - f^{k-1}}{h} + \mathscr{L}f^k + F\left(f^{k-1}\right) = 0.$$

We use f^{k-1} in F to avoid applying gradient descent to the nonlinear function F. Using the theory from Section 3.3 this goal is equivalent to minimizing

$$\frac{1}{2}\|w - f^{k-1}\|^2_{L^2(\Omega)} + h\left(\int_\Omega L(w) + F\left(f^{k-1}\right)w\,dx\right),$$

with L the Lagrangian corresponding to $\mathscr{L}$. For the boundary condition, we add the following terms [14]:

$$\frac{1}{2}\|w - f^{k-1}\|^2_{L^2(\Omega)} + h\left(\int_\Omega L(w) + F\left(f^{k-1}\right)w\,dx - \int_{\partial\Omega}\mathbf{n}\cdot A\nabla w(w-g)\,ds + \right.$$

$$\left. + \int_{\partial\Omega}\frac{\gamma}{2}(w-g)^2\,ds\right),$$

with penalty parameter γ. Applying a Monte Carlo approximation to the integrals, we define the loss functional

$$\begin{aligned}\tilde{L}(\theta;x_i,z_i) = {}& \frac{1}{N_I}\sum_{i=1}^{N_I}\left[\left(f^k(x_i;\theta) - f^{k-1}(x_i)\right)^2 + h\left(L(x_i) + F\left(f^{k-1}(x_i)\right)f^k(x_i)\right)\right] \\ & + \frac{|\partial\Omega|}{|\Omega|N_B}\sum_{i=1}^{N_B}\left[-\mathbf{n}\cdot A\nabla f^k(z_i;\theta)\left(f^k(z_i;\theta) - g(t,z_i)\right) + \frac{\gamma^i}{2}\left(f^k(z_i;\theta) - g(t,z_i)\right)^2\right],\end{aligned} \tag{3.13}$$

with N_I the number of samples on the interior domain and N_B the number of samples on the boundary domain. For the penalty parameter, we choose [14]

$$\gamma^i = \max_{z_i} 500\frac{|\partial\Omega|N_I\lambda_d|\nabla f^{k-1}(z_i)|^2}{|\Omega N_B|\nabla f^{k-1}(x_i)}.$$

An overview of the TDNM is in Algorithm 2.

Algorithm 2 Time Deep Nitsche Method

1: Initialize θ_0^0.
2: Initialize a neural network approximating the initial condition

$$f^0 = \min_{w \in H^1(\Omega)} \|w - \Phi(\mathbf{X})\|_{L_2(\Omega)}.$$

3: **for** each time step $k = 1, ..., N_t$ **do**
4: Apply Algorithm 1 with initial parameter guess $\theta_0^k = \theta^{k-1}$ and loss function $\tilde{L}$ from (3.13).
5: **end for**

3.5.3. IMPLEMENTATION DETAILS

We use the following neural network architecture [27]:

$$\begin{aligned}
S^1 &= \sigma_1\left(W^1\mathbf{x} + b^1\right), \\
Z^l &= \sigma_2\left(U^{z,l}\mathbf{x} + W^{z,l}S^l + b^{z,l}\right), && l = 1, ..., L, \\
G^l &= \sigma_2\left(U^{g,l}\mathbf{x} + W^{g,l}S^1 + b^{g,l}\right), && l = 1, ..., L, \\
R^l &= \sigma_2\left(U^{r,l}\mathbf{x} + W^{r,l}S^l + b^{r,l}\right), && l = 1, ..., L, \\
H^l &= \sigma_3\left(U^{h,l}\mathbf{x} + W^{h,l}\left(S^l \odot R^l\right) + b^{h,l}\right), && l = 1, ..., L, \\
S^{l+1} &= \left(1 - G^l\right) \odot H^l + Z^l \odot S^l, && l = 1, ..., L, \\
f(t, \mathbf{x}; \theta) &= \sigma_4\left(WS^{L+1} + b\right),
\end{aligned} \tag{3.14}$$

with L the number of layers, σ_i an activation function and $\odot$ denoting element-wise multiplication. We use a depth of 3 and a width of 50 [27].

We take 600 samples for every variable of the neural network [14]. So if we take $600d$ samples on the interior domain for the DGM, we take $600(d-1)$ samples on the terminal and boundary domain and for the TDNM $600(d-1)$ samples on the interior domain and $600(d-2)$ samples on the boundary domain.

We use 100,000 sampling stages for the DGM. For the TDNM, we take the number of intervals $N_t = 100$. So for a fair comparison, we use 100,000 sampling stages on the initial network and $\frac{100{,}000}{N_t} = 1000$ for the subsequent networks. The training was performed on supercomputer Delft Blue [1].

The last hyperparameters we have to choose are the activation functions and the learning rate. For the activation functions [27] used $\sigma_1 = \sigma_2 = \sigma_3 = \tanh$ and no σ_4. A new and promising activation function is the Gaussian Error Linear Unit (GELU) [18]. We test replacing each of the σ_i by the GELU to determine which network to use.

For the learning rate, [27] proposed the following schedule:

$$\alpha_n = \begin{cases} 10^{-4}, & 5000 \geq n, \\ 5 \times 10^{-5}, & 5000 < n \leq 10000, \\ 10^{-5}, & 10000 < n \leq 30000, \\ 5 \times 10^{-6}, & 20000 < n \leq 30000, \\ 10^{-6}, & 30000 < n \leq 40000, \\ 5 \times 10^{-7}, & 40000 < n \leq 45000, \\ 10^{-7}, & 45000 < n. \end{cases} \tag{3.15}$$

[5] and [14] used a larger learning rate. Therefore, we compare learning rate schedule (3.15) to the same learning rate schedule but with every learning rate multiplied by ten:

$$\alpha_n = \begin{cases} 10^{-3}, & 5000 \geq n, \\ 5 \times 10^{-4}, & 5000 < n \leq 10000, \\ 10^{-4}, & 10000 < n \leq 30000, \\ 5 \times 10^{-5}, & 20000 < n \leq 30000, \\ 10^{-5}, & 30000 < n \leq 40000, \\ 5 \times 10^{-6}, & 40000 < n \leq 45000, \\ 10^{-6}, & 45000 < n. \end{cases} \tag{3.16}$$

The results of these hyperparameter comparisons are in Section 4.1.

After choosing the right hyperparameters, we apply both neural network-methods to option pricing. The output of the neural network is the option price. In the Black-Scholes model, the input variables are S and t for the DGM and only S for the TDNM.

Considering the moneyness $\frac{S}{K}$ instead of the stock price S, will return the relative option price $\frac{V}{K}$. So we can compute the option prices for all combinations of S and K with only one parameter. In all computations, we substitute the moneyness for the stock price and keep the notation S.

We wish to test our models for time to maturity t between 0 and 1 and moneyness S between 0 and 2. When training, we sample t uniformly between 0 and 1 and S uniformly between 0 and 3 for call options and between 0 and 2 for put options [5].

After training the model for a fixed r and σ, we add σ and r to our input variables [21]. Then we can train the neural network to model option prices for random σ and r instead of training the network for each set of combinations of σ and r. We sample r uniformly between 0.0 and 0.1 and σ uniformly between 0.01 and 1 [21]. The results for the Black-Scholes model are in Section 4.3.

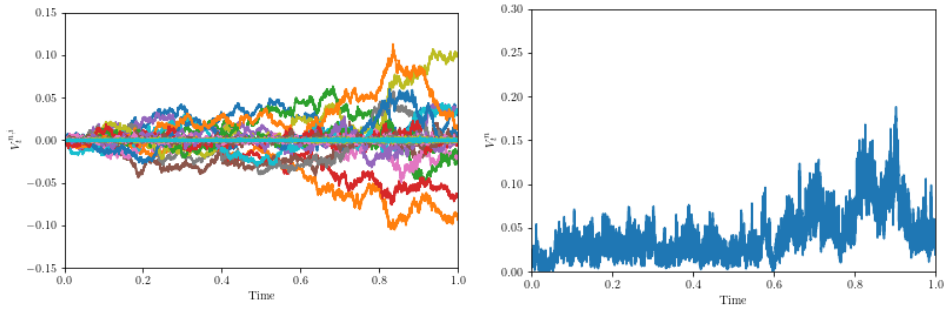
In the lifted Heston model, the input variables are S, t and $V^{n,i}$ for the DGM and S and $V^{n,i}$ for the TDNM. The sampling of the $V^{n,i}$ is explained below. The results for the lifted Heston model are in Section 4.4.

To help the algorithm to converge, we can add boundary conditions imposing what the limiting behavior should be. As the moneyness becomes large, the probability that

the put option becomes worth money goes to zero. Therefore, we add the boundary condition $V_{put} = 0$ at the upper S-boundary. Using the put-call parity, $V_{call} = S - Ke^{rt}$ at the upper S-boundary. Similarly, $V_{call} = 0$ and $V_{put} = Ke^{-rt} - S$ at the lower S-boundary.

SAMPLING OF VARIANCE PROCESS

Although the variance process V^n should remain nonnegative, some factors might become negative [2]. In Figure 3.2 we plot a Monte Carlo simulation of the individual and total variance processes. Indeed, the individual factor become negative while the total variance process always stays nonnegative.

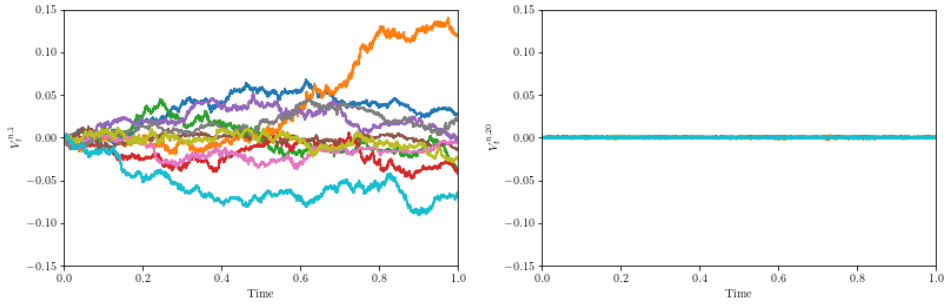


(a) The factors $V^{20,i}$ of the variance process.

(b) The total variance process V^{20}.

Figure 3.2: Monte Carlo simulation of the variance process for $H = 0.1$, $\lambda = 0.3$, $\eta = 0.3$, $\theta = 0.02$ and $V_0 = 0.02$.

In Figure 3.3 we plot 10 samples of a Monte Carlo simulation for $V^{20,1}$ and $V^{20,20}$. The $V^{n,1}$ become larger than $V^{n,n}$, because γ_1^n is smaller than γ_n^n. In Figure 3.4 we plot the maximum of $V^{20,i}$ and $V^{5,i}$ until time $T = 1$ in 100 Monte Carlo simulations against i. The maximum decreases almost linearly with i.

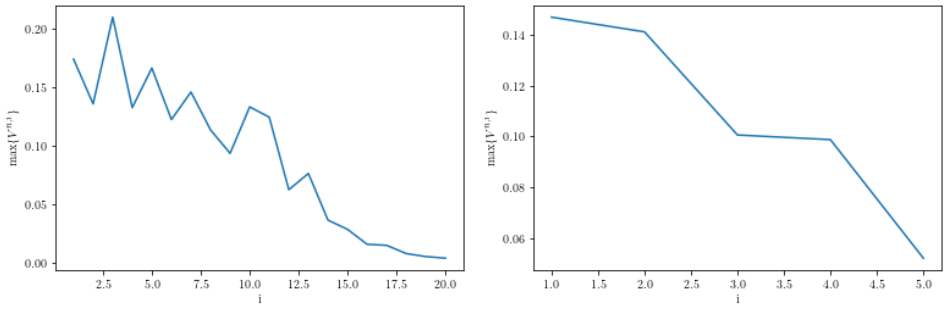


(a) $V^{20,1}$

(b) $V^{20,20}$

Figure 3.3: Monte Carlo simulation of the first and last factor of the variance process for $H = 0.1$, $\lambda = 0.3$, $\eta = 0.3$, $\theta = 0.02$ and $V_0 = 0.02$.

We choose to sample $V^{n,1}$ uniformly between -0.5 and 0.5 and decrease these boundaries linearly with i until $\frac{-0.5}{n}$ and $\frac{0.5}{n}$ for $V^{n,n}$. We then calculate V^n and discard all



(a) $V^{20,i}$ (b) $V^{5,i}$

Figure 3.4: The maximum of the Monte Carlo simulation of $V^{n,i}$ until $T = 1$ against i for $H = 0.1$, $\lambda = 0.3$, $\eta = 0.3$, $\theta = 0.02$ and $V_0 = 0.02$.

combinations which give negative V^n. Empirically more than half of the samples gives nonnegative V^n leaving enough samples to train the neural network.

4

RESULTS

This chapter presents the results of pricing options using the methods explained in the previous chapter. Section 4.1 selects good choices for the hyperparameters of the model. Section 4.2 compares the splitting method to the transformation method in the Black-Scholes model. Section 4.3 presents the results in the Black-Scholes model and Section 4.4 in the lifted Heston model.

4.1. HYPERPARAMETERS

Before we use our model, we test what good choices for the hyperparameters are. In Subsection 4.1.1 we test what a good activation function is. In Subsection 4.1.2 we test what a good learning rate is.

4.1.1. ACTIVATION FUNCTION

The first hyperparameter that is important to our model is the activation function. We test replacing each of the standard activation functions $\sigma_1 = \sigma_2 = \sigma_3 = \tanh$ and no σ_4 in (3.14) by the GELU to determine which network to use. We train the Deep Galerkin Method (DGM) with only 1000 sampling stages and compute the error compared to the exact solution of a Black-Scholes call option. We use fewer sampling stages so that the errors are larger and the comparisons clearer. The results are in Table 4.1. Using only GELU activation function gives the lowest error.

In Figure 4.1 we plot the DGM solution for the standard activation functions and only GELU activation functions together with the exact solution. We can see that the DGM with the optimal activation functions is indeed closer to the exact solution than with the standard activation functions. Therefore, we choose to use only GELU activation functions.

4.1.2. LEARNING RATE

The second hyperparameter that is important to our model is the learning rate. We compare small learning rate (3.15) to large learning rate (3.16). In Figure 4.2 we present the

	σ_1	tanh		GELU	
$\sigma 3$	σ_4 \ σ_2	tanh	GELU	tanh	GELU
tanh	none	1.61	0.77	1.15	1.35
	GELU	45.28	1.20	45.27	45.25
GELU	none	0.70	0.76	2.12	0.85
	GELU	0.64	1.15	1.90	0.37

Table 4.1: Total absolute error of the DGM using only 1000 sampling stages with different activation functions compared to the exact solution for the European call option in the Black-Scholes model for $r = 0.05$ and $\sigma = 0.25$. The results are evaluated on 41 evenly spaced points on the domain $S = 0$ to $S = 2$ for times to maturity 0.01, 0.34, 0.67 and 1.

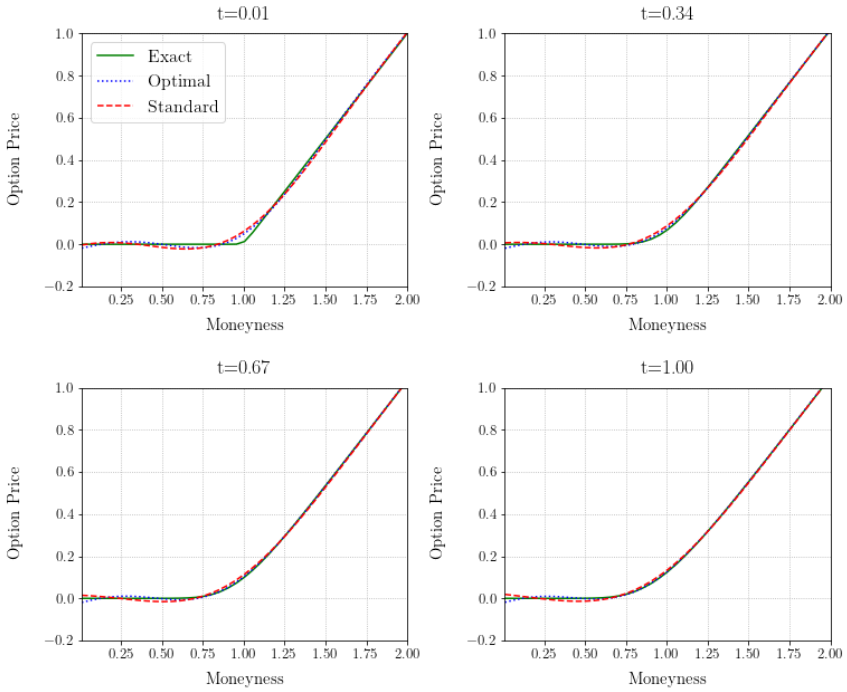


Figure 4.1: Comparison of the DGM with standard activation functions as in [27] and optimal GELU activation functions to the exact solution for a European call option in the Black-Scholes model with $r = 0.05$ and $\sigma = 025$.

result of pricing a European call option using the DGM with both learning rates. The larger learning rate (3.16) gives values closer to the exact solution.

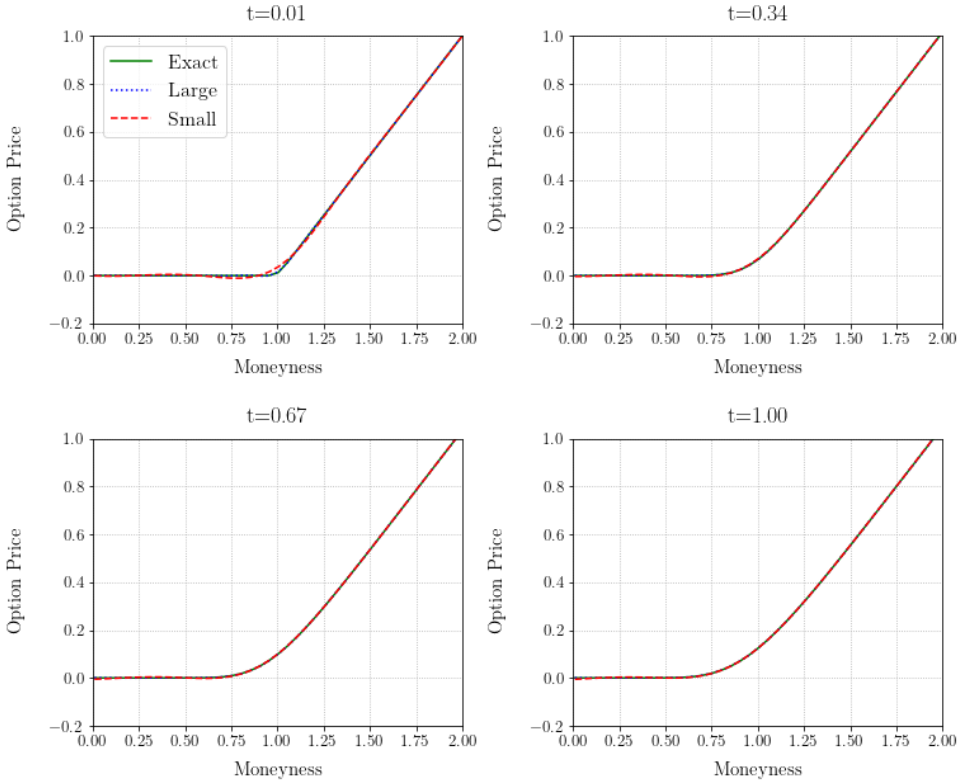
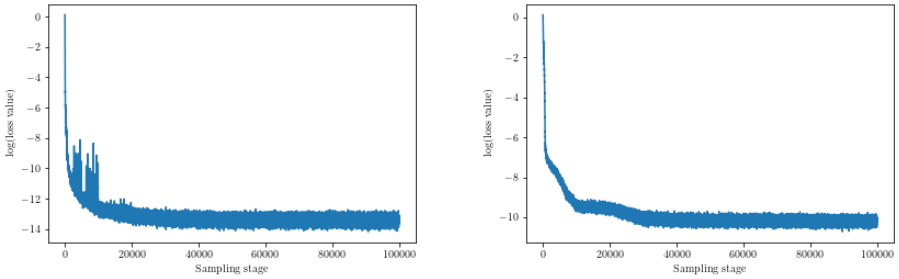


Figure 4.2: Comparison of the DGM with large learning rate (3.16) and the small learning rate (3.15) to the exact solution for a European call option in the Black-Scholes model with $r = 0.05$ and $\sigma = 025$.

In Figure 4.3 we plot the loss values at each time step for both learning rates. Although the large learning rate is fluctuating more, the loss values eventually become lower compared to the small learning rate. Therefore, we use learning rate (3.16).

4.2. SPLITTING METHOD

In this section, we compare the two methods for dealing with the asymmetry in the Black-Scholes model: the splitting method from Section 3.4 and the transformation to the heat equation based on Theorem 3.1. This comparison for a European call option is in Figure 4.4. Sampling small values of S leads to large negative values in the y-domain, which gives unstable results in the training of the neural network. Therefore, we choose to split the partial differential equation (PDE).



(a) Large learning rate

(b) Small learning rate

Figure 4.3: Loss values of the large learning rate (3.16) and the small learning rate (3.15) using the DGM algorithm.

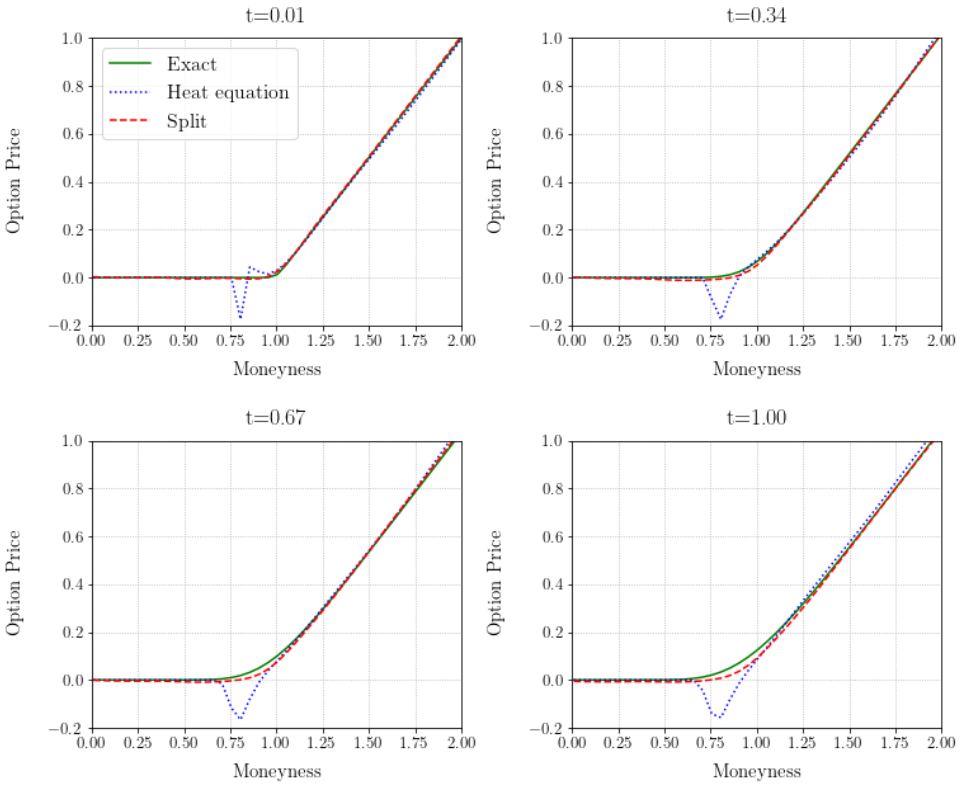


Figure 4.4: Comparison of the two methods for dealing with asymmetry for a European call option in the Black-Scholes model with $r = 0.05$ and $\sigma = 0.25$.

4.3. BLACK-SCHOLES

In this section, we test the performance of our neural network methods in the Black-Scholes model. We substitute the Black-Scholes PDE (2.9) in the loss function (3.12) for the DGM and the Black-Scholes nonsymmetric function (3.7) and Lagrangian (3.8) in the loss function (3.13) for the Time Deep Nitsche Method (TDNM). For European call options, we compare the neural network performance with the exact solution (2.10) and the COS method applied to the Black-Scholes characteristic function (2.11). For American put options, we compare the neural network performance with a finite difference method.

First, we test the performance of our neural network methods on a European call option in the Black-Scholes model. We apply the DGM and the TDNM to the Black-Scholes PDE (2.9). $r = 0.05$ and $\sigma = 0.25$ gives as a result Figure 4.5. The DGM is indistinguishable from the exact solution. The TDNM is indistinguishable for small times, but has a larger error for larger times.

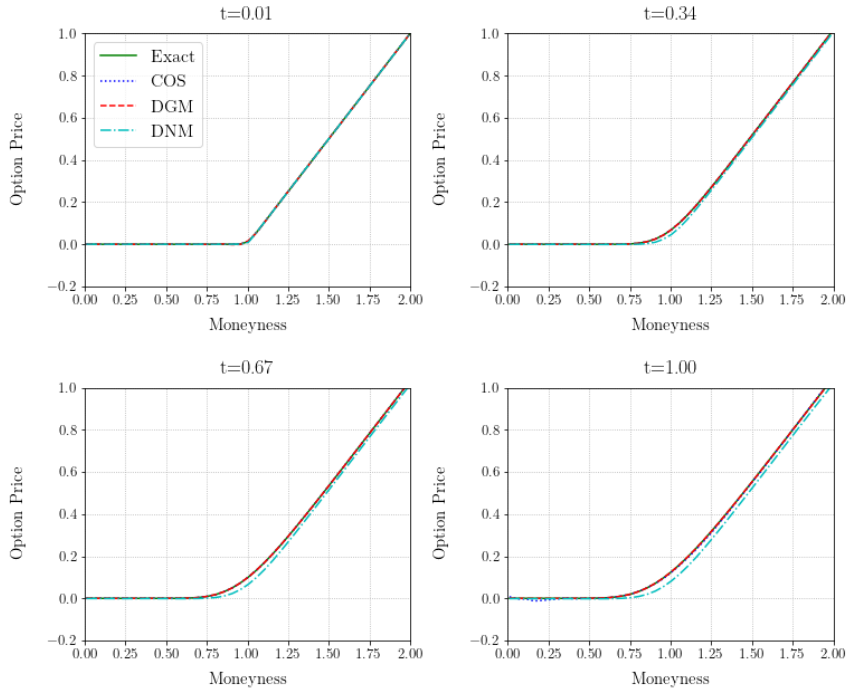


Figure 4.5: Relative European call option prices in the Black-Scholes model against the moneyness of the stock, computed using the COS Method with $N = 128$, $L = 30$ for $t = 0.01$ and $L = 10$ for the other times and the two neural network methods, compared to the exact solution for four different times to maturity. $r = 0.05$ and $\sigma = 0.25$.

Second, we test the performance of our neural network methods on an American put option in the Black-Scholes model. We choose $\sigma = 0.5$ to enlarge the difference with a European option and keep $r = 0.05$. This choice gives Figure 4.6. The DGM is indistinguishable from the finite difference method.

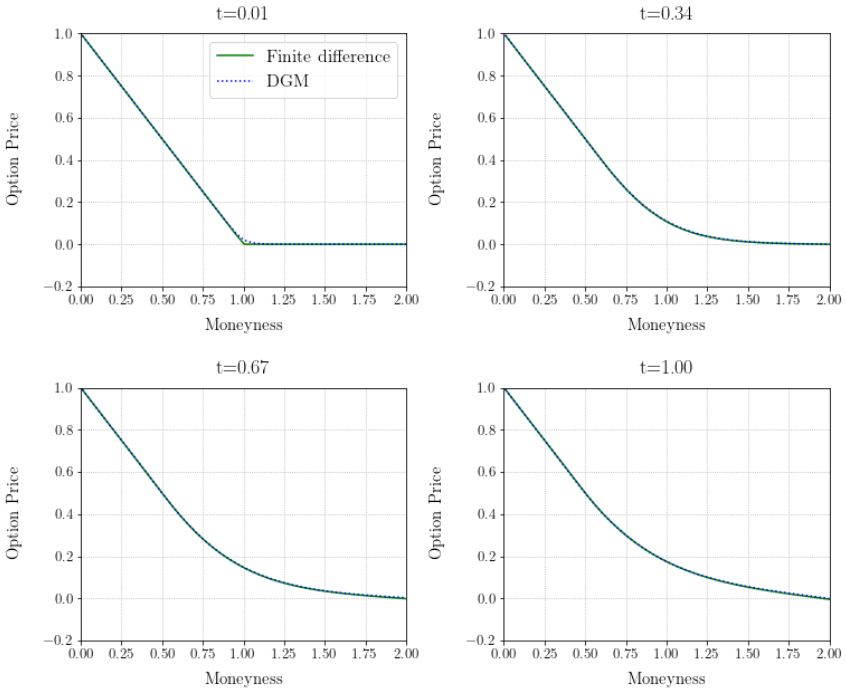


Figure 4.6: Relative American put option prices in the Black-Scholes model against the moneyness of the stock, computed using the DGM compared to finite difference method with 10000 steps for four different times to maturity. $r = 0.05$ and $\sigma = 0.5$.

Third, we test the performance of our neural network methods on a European call option in the Black-Scholes model with bonus variables σ and r. The results are in Figure 4.7. The DGM is indistinguishable from the exact solution for any t, σ and r. The TDNM performs almost as good for small t and σ, but is off for large t and σ.

4.4. LIFTED HESTON

In this section, we test the performance of our neural network methods in the lifted Heston model. We substitute the lifted Heston PDE (3.2) in the loss function (3.12) for the DGM and the lifted Heston nonsymmetric function (3.9) and Lagrangian (3.10) in the loss function (3.13) for the TDNM. We compare the neural network performance with COS method applied to the lifted Heston characteristic function from Subsection 3.2.

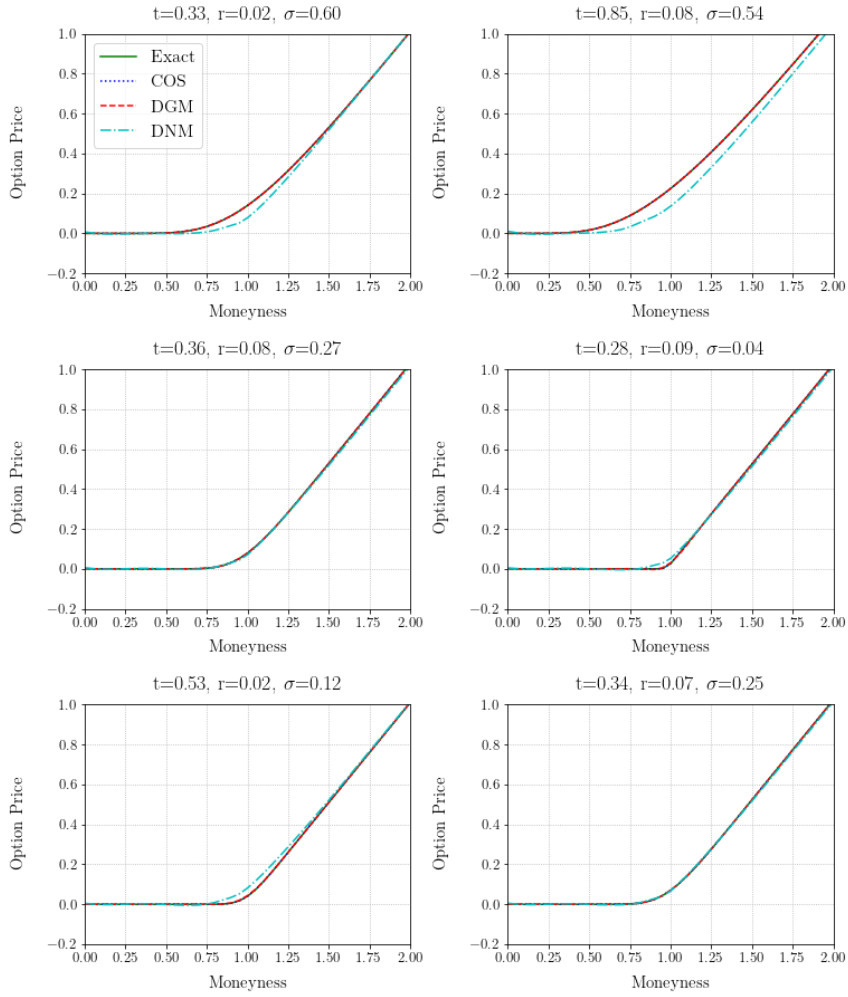


Figure 4.7: Relative European call option prices in the Black-Scholes model against the moneyness of the stock computed using the COS Method with $N = 1028$ and $L = 20$ and the two neural network methods compared to the exact solution for different times to maturity, interest rates and volatilities.

First, we test the performance of our neural network methods in the Lifted Heston model with $n = 1$, which is equivalent to the regular Heston model. We apply the DGM and the TDNM to price a European call option. We choose $r = 0.0$, $H = 0.1$, $\lambda = 0.3$, $\eta = 0.3$, $\rho = -0.7$, $\theta = 0.02$ and $V_0 = 0.02$ [2]. This choice of parameters gives Figure 4.8 as a result. The DGM is indistinguishable from the COS method. The TDNM is a little off for larger times.

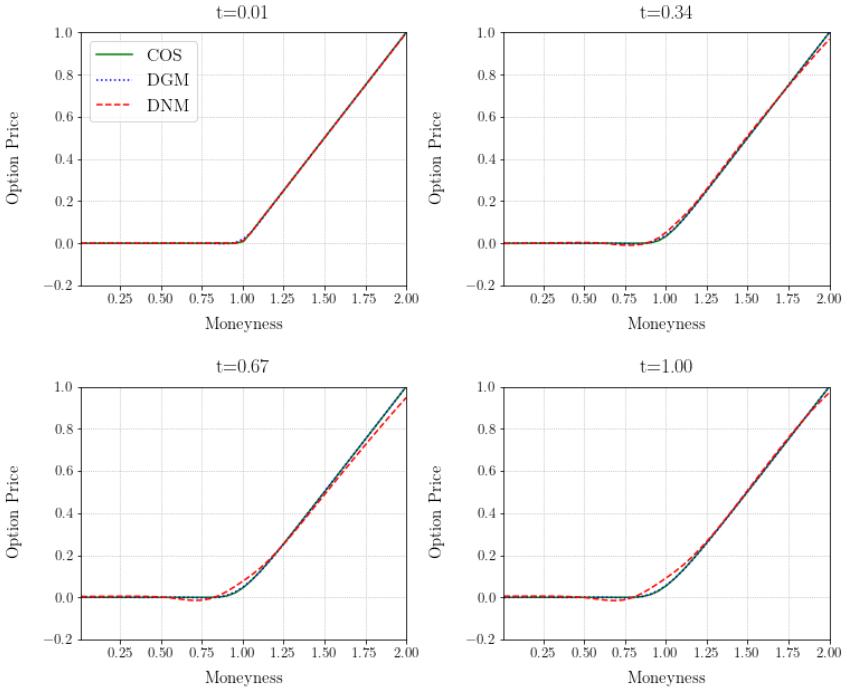


Figure 4.8: Relative European call option prices in the lifted Heston model with $n = 1$ against the moneyness of the stock computed using the COS Method with $N = 512$, $L = 30$ for $t = 0.01$ and $L = 10$ for the other times compared to the two neural network methods for different times to maturity, $r = 0.0$, $H = 0.1$, $\lambda = 0.3$, $\eta = 0.3$, $\rho = -0.7$,$\theta = 0.02$ and $V_0 = 0.02$.

Second, we test the performance of our neural network methods in the Lifted Heston model with $n = 5$. We keep $r = 0.0$, $H = 0.1$, $\lambda = 0.3$, $\eta = 0.3$, $\rho = -0.7$, $\theta = 0.02$ and $V_0 = 0.02$. This choice gives Figure 4.9 as a result. The neural networks perform similar as for $n = 1$.

Third, we test the performance of our neural network methods in the Lifted Heston model with $n = 20$. We keep $r = 0.0$, $H = 0.1$, $\lambda = 0.3$, $\eta = 0.3$, $\rho = -0.7$, $\theta = 0.02$ and $V_0 = 0.02$. This choice gives Figure 4.10 as a result. The DGM is less accurate than in lower dimensions. The TDNM is still accurate for small t but inaccurate for larger t.

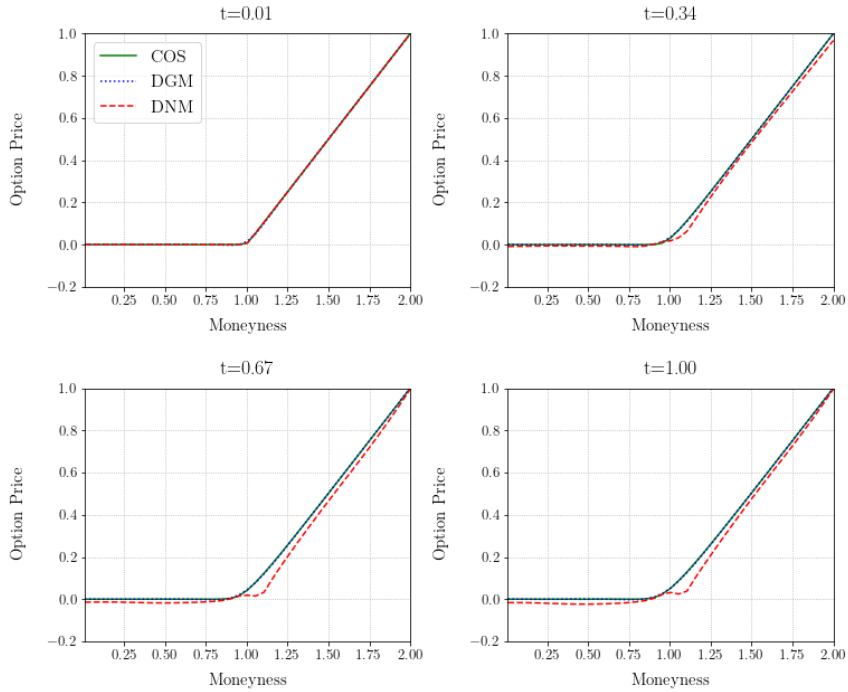


Figure 4.9: Relative European call option prices in the lifted Heston model with $n = 5$ against the moneyness of the stock computed using the COS Method with $N = 512$, $L = 30$ for $t = 0.01$ and $L = 10$ for the other times compared to the two neural network methods for different times to maturity, $r = 0.0$, $H = 0.1$, $\lambda = 0.3$, $\eta = 0.3$, $\rho = -0.7$,$\theta = 0.02$ and $V_0 = 0.02$.

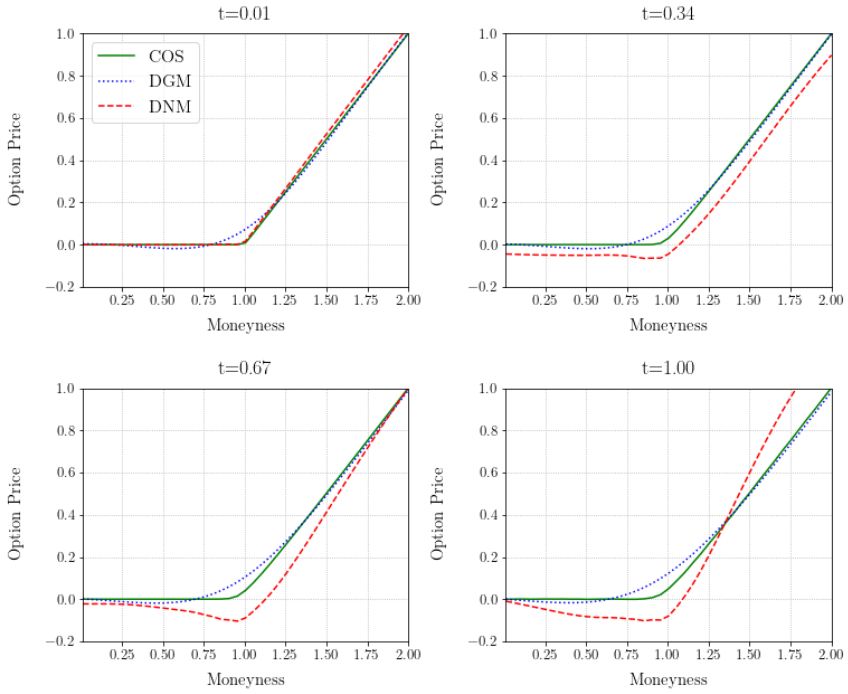


Figure 4.10: Relative European call option prices in the lifted Heston model with $n = 20$ against the moneyness of the stock computed using the COS Method with $N = 512$, $L = 30$ for $t = 0.01$ and $L = 10$ for the other times compared to the two neural network methods for different times to maturity, $r = 0.0$, $H = 0.1$, $\lambda = 0.3$, $\eta = 0.3$, $\rho = -0.7$,$\theta = 0.02$ and $V_0 = 0.02$.

5

CONCLUSION

In this research, we considered neural network-algorithms for option pricing. We used two different models for the stock price. First, the Black-Scholes model, for which an analytical solution exists. Second, the lifted Heston model which is a Markovian approximation of the rough Heston model, a stochastic volatility model. We applied two different methods to price options. First, we derived the option pricing partial differential equation (PDE), which we solved with a neural network. Second, we derived the conditional characteristic function of the stock price which lead to the option price with the COS method.

Once trained, the output of a neural network is an analytical function of the input, allowing fast option pricing even in high dimensions. We considered two neural network-algorithms. First, the Deep Galerkin Method (DGM), which uses a loss function measuring how well the neural network satisfies the option pricing PDE. Second, the Time Deep Nitsche Method (TDNM), which uses variational calculus to rewrite a PDE as a minimization problem. Initially, the TDNM could only be applied to symmetric PDEs. We extended the TDNM by splitting the PDE operator in a symmetric part and an asymmetric part. This splitting allows applying the TDNM to the option pricing PDE.

When training the neural network, we chose a learning rate decaying from 10^{-3} to 10^{-6}. As activation function, we chose the Gaussian Error Linear Unit. We found that this combination gave the best results.

We compared our splitting method to transforming the Black-Scholes option pricing PDE to the symmetric heat equation. We found that the splitting method was more stable.

We used the neural networks to price options in the Black-Scholes model. The DGM can predict both European and American options prices perfectly. When r and σ are added as variables to the neural network, it remained accurate. The TDNM can predict European option prices perfectly for small times to maturity and volatilities, but has larger errors for large times to maturity and volatilities.

We used our the neural networks to price options in the lifted Heston model as well.

The DGM can predict European options prices perfectly for $n = 5$, but has larger errors for $n = 20$. The TDNM can predict European option prices perfectly for small times to maturity, but has larger errors for large times to maturity and volatilities.

Some improvements on the research are still possible. First, for the sampling of the variance processes in the lifted Heston model, we sampled uniformly between two bounds. However, the variance processes are denser around 0 than at larger values. Furthermore, the variance increases with time. A better sampling method can be considered.

Second, the constants chosen in the lifted Heston model to approximate the rough Heston model are not optimal. Better estimates exist, enabling even better modelling of reality [6]. These estimates require smaller n for the lifted Heston model to be as close to the rough Heston model as the standard estimates that we used. As the constants do not change the essence of the research, we did not consider the optimal choices here.

Third, the hyperparameters of the model were optimized for DGM. For TDNM, other hyperparameters might be better, improving the fit. However, theoretically the activation function should not matter and in general the method converges as long as the learning rate drops over time.

Fourth, in the splitting method, we use the previous neural network for the asymmetric part. To be accurate, it is therefore important that the previous network does not differ too much from the current network. Therefore, it might help to take more intervals so that we have a smaller time step.

For further research, there are several interesting possibilities to explore. First, in the lifted Heston the parameters r, η, H. λ, ρ, θ and V_0 can be added as variables to the neural network. Then we can train the neural network to model option prices for random parameter choices, as we successfully did in the Black-Scholes model.

Second, it is interesting to see if the TDNM can be extended to free boundary PDEs. Then, we can use the algorithm to solve American option prices as well.

Third, it might be interesting to model implied volatility as well, besides the option price. Many people and institutions trading assets are interested in the implied volatility of the options they trade, as this enables them to hedge the risk they are exposed to.

BIBLIOGRAPHY

[1] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 1)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1. 2022.

[2] Eduardo Abi Jaber. "Lifting the Heston model". In: *Quantitative Finance* 19.12 (2019), pp. 1995–2013.

[3] Eduardo Abi Jaber and Omar El Euch. "Multifactor approximation of rough volatility models". In: *SIAM Journal on Financial Mathematics* 10.2 (2019), pp. 309–349.

[4] Craig Anthony. *The volatility surface explained*. May 2022. URL: https://www.investopedia.com/articles/stock-analysis/081916/volatility-surface-explained.asp.

[5] Ali Al-Aradi et al. "Solving nonlinear and high-dimensional partial differential equations via deep learning". In: *arXiv preprint arXiv:1811.08782* (2018).

[6] Christian Bayer and Simon Breneis. "Makovian approximations of stochastic Volterra equations with the fractional kernel". In: *arXiv preprint arXiv:2108.05048* (2021).

[7] Christian Bayer, Peter Friz, and Jim Gatheral. "Pricing under rough volatility". In: *Quantitative Finance* 16.6 (2016), pp. 887–904.

[8] Christian Bayer, Antonis Papapantoleon, and Tempone Raul. *Computational finance*. 2018.

[9] Lawrence C Evans. *Partial differential equations*. Vol. 19. American Mathematical Soc., 2010.

[10] Fang Fang and Cornelis W Oosterlee. "A novel pricing method for European options based on Fourier-cosine series expansions". In: *SIAM Journal on Scientific Computing* 31.2 (2009), pp. 826–848.

[11] Damir Filipovic and Eberhard Mayerhofer. "Affine diffusion processes: theory and applications". In: *Advanced financial modelling* 8 (2009), pp. 1–40.

[12] Crispin W Gardiner et al. *Handbook of stochastic methods*. Vol. 3. springer Berlin, 1985.

[13] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. "Volatility is rough". In: *Quantitative finance* 18.6 (2018), pp. 933–949.

[14] Emmanuil H. Georgoulis, Michail Loulakis, and Asterios Tsiourvas. *Discrete Gradient Flow Approximations of High Dimensional Evolution Partial Differential Equations via Deep Neural Networks*. 2022. DOI: 10.48550/ARXIV.2206.00290. URL: https://arxiv.org/abs/2206.00290.

[15] Maximilien Germain, Huyên Pham, and Xavier Warin. "Neural networks-based algorithms for stochastic control and PDEs in finance". In: *arXiv preprint arXiv:2101.08068* (2021).

[16] Alessandro Gnoatto, Christoph Reisinger, and Athena Picarelli. “Deep xVA Solver–A neural network based counterparty credit risk management framework”. In: *Available at SSRN 3594076* (2020).

[17] Lukas Gonon and Christoph Schwab. “Deep ReLU Network Expression Rates for Option Prices in high-dimensional, exponential Lévy models”. In: *arXiv preprint arXiv:2101.11897* (2021).

[18] Dan Hendrycks and Kevin Gimpel. “Gaussian Error Linear Units (GELUs)”. In: *arXiv preprint arXiv:1606.08415* (2016).

[19] Steven L Heston. “A closed-form solution for options with stochastic volatility with applications to bond and currency options”. In: *The review of financial studies* 6.2 (1993), pp. 327–343.

[20] Hui-Hsiung Kuo. *Introduction to Stochastic Integration*. Springer-Verlag New York, 2006. DOI: https://doi.org/10.1007/0-387-31057-6.

[21] Shuaiqiang Liu, Cornelis W. Oosterlee, and Sander M. Bohte. “Pricing Options and Computing Implied Volatilities using Neural Networks”. In: *Risks* 7.1 (2019). ISSN: 2227-9091. DOI: 10.3390/risks7010016. URL: https://www.mdpi.com/2227-9091/7/1/16.

[22] Marek Musiela and Marek Rutkowski. “American Options”. In: *Martingale Methods in Financial Modelling*. Springer, 2005, pp. 205–228.

[23] Ariel Neufeld and Julian Sester. “A deep learning approach to data-driven model-free pricing and to martingale optimal transport”. In: *arXiv preprint arXiv:2103.11435* (2021).

[24] Cornelis W Oosterlee and Lech A Grzelak. *Mathematical Modeling and Computation in Finance: With Exercises and Python and Matlab Computer Codes*. World Scientific, 2019.

[25] Myron Scholes and Fischer Black. “The pricing of options and corporate liabilities”. In: *Journal of Political Economy* 81.3 (1973), pp. 637–654.

[26] Justin Sirignano and Konstantinos Spiliopoulos. “Asymptotics of reinforcement learning with neural networks”. In: *arXiv preprint arXiv:1911.07304* (2021).

[27] Justin Sirignano and Konstantinos Spiliopoulos. “DGM: A deep learning algorithm for solving partial differential equations”. In: *Journal of computational physics* 375 (2018), pp. 1339–1364.

[28] *What is a neural network?* URL: https://www.tibco.com/reference-center/what-is-a-neural-network.