# Application of Neural Networks in Solving the Heston PDE

Linus Wunderlich  Ivelina Mladenova

Queen Mary University of London

## Introduction

### Problem Statement

Options trading volumes have more than doubled in the last decade, highlighting the need for more timely and adaptable options pricing methods. Traditional methods for solving PDEs, and hence pricing options, struggle in high-dimensional settings. In contrast, we have seen neural networks (NNs) excel across various domains and are not affected by the curse of dimensionality, making them ideal for approximating **high-dimensional parametric PDEs**. However, neural networks are faced with the generalisation puzzle — performing well on new, unseen data after training on a limited dataset. The theory of **Neural Tangent Kernels (NTKs)**, developed in 2018 by [2], can help to analyse the training dynamics in the Deep Parametric Differential Equation (DPDE) method proposed by [1].

### Fully-Connected Neural Networks (FCNN)

The forward pass of a fully-connected network with parameters $\boldsymbol{\theta}$, is given by the function $f(\cdot;\boldsymbol{\theta}) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ with $L$ hidden layers, each containing $n_0, n_1, \cdots n_L$ neurons. The total parameters are $P = \sum_{h=0}^{L-1}(n_h+1)n_{h+1}$, hence $\boldsymbol{\theta} \in \mathbb{R}^P$.

Using the normalisation from NTK theory in infinitely wide neural networks, a FCNN is defined recursively as:

$$\begin{aligned}
&\alpha^{(0)} = \mathbf{x} \\
&\tilde{\alpha}^{(h+1)}(\mathbf{x}) = \frac{1}{\sqrt{n_h}}\mathbf{W}^{(h)}\alpha^{(h)} + \beta\mathbf{b}^{(h)} \quad \in \mathbb{R}^{n_{h+1}} \quad \text{; pre-activations} \\
&\alpha^{(h+1)}(\mathbf{x}) = \sigma\left(\tilde{\alpha}^{(h+1)}(\mathbf{x})\right) \quad \in \mathbb{R}^{n_{h+1}} \quad \text{; post-activations}
\end{aligned}$$

where $\mathbf{W}^{(h)} \in \mathbb{R}^{n_{h+1}} \times \mathbb{R}^{n_h}$ and $\mathbf{b}^{(h)} \in \mathbb{R}^{n_{h+1}}$ are the weights and biases and $\theta = \{\mathbf{W}^{(h)}, \mathbf{b}^{(h)}\}$.

### Well Posed Parametric Partial Differential Equation (PDE)

Consider a well-posed parametric PDE defined on a bounded domain $(0,T)\times\mathbb{R}^d$, with the state $\Omega$ and parameter $\Lambda$ domains respectively defined as subsets of $\mathbb{R}^d, \mathbb{R}^p$, with $d, p \in \mathbb{N}$, given by

$$\begin{aligned}
&\mathcal{D}^{\boldsymbol{\mu}}_{\tau,x}[u](\tau,\boldsymbol{x},\boldsymbol{\mu}) = f(\tau,\boldsymbol{x},\boldsymbol{\mu}), \quad (\tau,x) \in (0,T)\times\Omega\times\Lambda \\
&u(0,\boldsymbol{x},\boldsymbol{\mu}) = g(\boldsymbol{x},\boldsymbol{\mu}), \quad x \in \Omega\times\Lambda \\
&u(\tau,\boldsymbol{x},\boldsymbol{\mu}) = h(\tau,\boldsymbol{x},\boldsymbol{\mu}), \quad (\tau,x) \in (0,T)\times\partial\Omega\times\Lambda.
\end{aligned}$$

The PDE has a continuous nonlinear differential operator $\tau, x$, $\mathcal{D}^{\boldsymbol{\mu}}_{\tau,x} = \mathcal{D}$.

## Loss Function in Unsupervised Learning

The deep Parametric PDE method (DPPDE) method is a generalisation of the Physics Informed Neural Networks **(PINNs)** Method and suggests that we approximate the solution of the PDE using a neural network function $\tilde{u}(\tau,\boldsymbol{x},\boldsymbol{\mu};\boldsymbol{\theta}) \approx u(\tau,\boldsymbol{x},\boldsymbol{\mu})$.

The parameters $\boldsymbol{\theta}$ of the network are trained with an unsupervised learning method by minimising the following composite loss function

$$\mathcal{L}(\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\sum_{i=1}^{N_b}\left|\tilde{u}\left(\boldsymbol{x}_b^i,\boldsymbol{\mu}_b^i;\boldsymbol{\theta}\right) - g\left(\boldsymbol{x}_b^i,\boldsymbol{\mu}_b^i\right)\right|^2}_{\mathcal{L}_b(\boldsymbol{\theta})} + \mathcal{L}_r(\boldsymbol{\theta}), \tag{1}$$

where

$$\mathcal{L}_r(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N_r}\left|\mathcal{D}\tilde{u}(\tau_r^i,\boldsymbol{x}_r^i,\boldsymbol{\mu}_r^i;\boldsymbol{\theta}) - f(\tau_r^i,\boldsymbol{x}_r^i,\boldsymbol{\mu}_r^i)\right|^2. \tag{2}$$

where $\mathcal{L}_b(\boldsymbol{\theta})$ and $\mathcal{L}_r(\boldsymbol{\theta})$ are the boundary and residual losses found using batches of uniform randomly sampled training data $\{\boldsymbol{x}_b^i,\boldsymbol{\mu}_b^i\}_{i=1}^{N_b}$ and $\{\tau_r^i,\boldsymbol{x}_r^i,\boldsymbol{\mu}_r^i\}_{i=1}^{N_r}$ respectively.

## Option Pricing Models

### Black-Scholes-Merton (BSM) Model

The Black-Scholes-Merton model is the most widely used model for *pricing option contracts* due to its analytical solution. It assumes that the stock price follows a geometric Brownian motion with a constant drift and volatility. The PDE is given by $\mathcal{D}_{\tau,S}U = \frac{\partial U}{\partial\tau} + \frac{1}{2}\sigma^2S^2\frac{\partial^2U}{\partial S^2} + rS\frac{\partial U}{\partial S} - rU$, where $\boldsymbol{S}$ is the stock price and $\boldsymbol{\sigma}^2$ is the constant volatility, and the solution $\boldsymbol{U}(\tau,\boldsymbol{S})$ gives the price of the option. However, the model's assumptions for the market conditions are *unrealistic in practice*.

### Heston Model

The Heston model accounts for the dynamic nature of volatility observed in financial markets by modeling both the asset price and its volatility as stochastic processes. Under risk-neutral pricing, the Heston partial differential equation (PDE) is derived to price options, given by:

$$\mathcal{D}^{\mu}_{\tau,x}U = \frac{\partial U}{\partial\tau} - \frac{1}{2}v\frac{\partial^2U}{\partial x^2} - \rho\gamma v\frac{\partial^2U}{\partial x\partial v} - \frac{1}{2}\gamma^2v\frac{\partial^2U}{\partial v^2} - (r-\frac{1}{2}v)\frac{\partial U}{\partial x} - \kappa(\theta-v)\frac{\partial U}{\partial v} + rU \tag{3}$$

$$g(x,\mu) = \max\{0, e^x - K\}, \tag{4}$$

where $\boldsymbol{U}$ is the call option price, $\boldsymbol{\tau}$ is the time to maturity, $\boldsymbol{v}$ is the instantaneous variance, $\boldsymbol{\gamma}$ is the volatility of volatility, $\boldsymbol{\rho}$ is the correlation coefficient between the asset price and its volatility, $\boldsymbol{r}$ is the risk-free rate, $\boldsymbol{\kappa}$ is the mean reversion speed, and $\boldsymbol{\theta}$ is the long-term average variance. The boundary condition $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{\mu})$ reflects the payoff at expiry. The large number of model parameters makes it a high-dimensional problem which makes it harder to solve.

**The direct application of DPDE fails to yield satisfactory results**

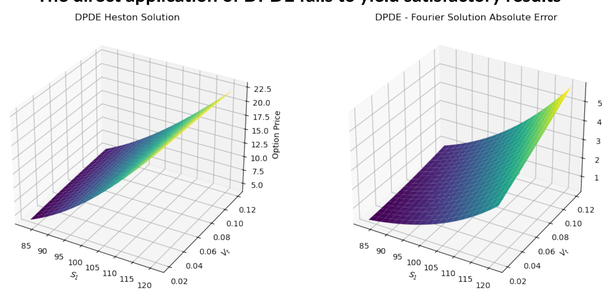


Figure 1. DPDE solution for the Heston model.

**Using the known Black-Scholes solution helps improve the accuracy**

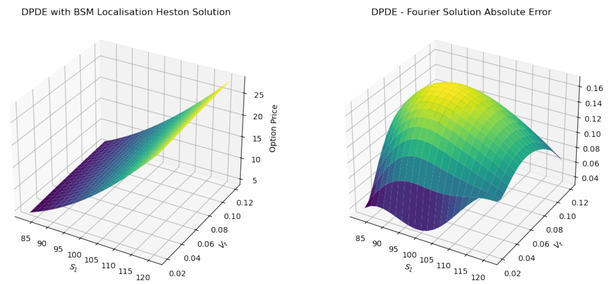


Figure 2. DPDE with Black-Scholes localisation solution for the Heston model.

A naive application of the **DPDE** method does not give satisfactory results. For better results we *transformed the PDE*, taking into account the known BSM formula, which we *subtract* from the unknown solution of the Heston PDE. The transformation of the PDEs simplifies training and yields far better results with **errors below $0.2$**.

## Neural Tangent Theory of DPPDE

To learn the parameters $\boldsymbol{\theta}$ of the network approximation $\tilde{u}(\cdot;\boldsymbol{\theta})$, we would like to minimise the empirical loss function $\mathcal{L}(\boldsymbol{\theta})$ in (1) by gradient descent (GD). Using an infinitesimally small learning rate (step size) in GD, yields the continuous-time gradient flow (GF) training dynamics $d\boldsymbol{\theta}/dt = -\nabla_\theta\mathcal{L}(\boldsymbol{\theta})$. The training dynamics of a neural network approximation of a parametric PDE $\tilde{u}(0,\boldsymbol{x},\boldsymbol{\mu};\boldsymbol{\theta}(t)) = \tilde{u}_b(\boldsymbol{\theta}(t))$ and $\mathcal{D}\tilde{u}(\tau,\boldsymbol{x},\boldsymbol{\mu};\boldsymbol{\theta}(t)) = \mathcal{D}\tilde{u}_r(\boldsymbol{\theta}(t))$ during GF training evolve in the following way

$$\begin{bmatrix} \frac{d\tilde{u}_b(\boldsymbol{\theta}(t))}{dt} \\ \frac{d\mathcal{D}\tilde{u}_r(\boldsymbol{\theta}(t))}{dt} \end{bmatrix} = -\underbrace{\begin{bmatrix} \boldsymbol{K}_{bb}(t) & \boldsymbol{K}_{br}(t) \\ \boldsymbol{K}_{rb}(t) & \boldsymbol{K}_{rr}(t) \end{bmatrix}}_{\boldsymbol{K}(t)} \cdot \begin{bmatrix} \tilde{u}\left(\boldsymbol{x}_b,\boldsymbol{\mu}_b;\boldsymbol{\theta}(t)\right) - g\left(\boldsymbol{x}_b,\boldsymbol{\mu}_b\right) \\ \mathcal{D}\tilde{u}\left(\boldsymbol{\tau}_r,\boldsymbol{x}_r,\boldsymbol{\mu}_r;\boldsymbol{\theta}(t)\right) - f\left(\tau,\boldsymbol{x}_r,\boldsymbol{\mu}_r\right) \end{bmatrix}.$$

$\boldsymbol{K}_{rr}(t), \boldsymbol{K}_{bb}(t), \boldsymbol{K}_{rb}(t), \boldsymbol{K}_{br}(t)$ are all positive semi-definite matrices, $\boldsymbol{K}_{rb}(t) = \boldsymbol{K}_{br}^\top(t)$ and $\boldsymbol{K}(t)$ is the **Neural Tangent Kernel** *(NTK)* where

$$\left(\boldsymbol{K}_{bb}\right)_{ij}(t) = \left\langle \frac{d\tilde{u}_b(\cdot;\boldsymbol{\theta}(t))}{d\boldsymbol{\theta}}, \frac{d\tilde{u}_b(\cdot;\boldsymbol{\theta}(t))}{d\boldsymbol{\theta}} \right\rangle, \left(\boldsymbol{K}_{br}\right)_{ij}(t) = \left\langle \frac{d\tilde{u}_b(\cdot;\boldsymbol{\theta}(t))}{d\boldsymbol{\theta}}, \frac{d\mathcal{D}\tilde{u}_r(\cdot;\boldsymbol{\theta}(t))}{d\boldsymbol{\theta}} \right\rangle$$

$$\left(\boldsymbol{K}_{rr}\right)_{ij}(t) = \left\langle \frac{d\mathcal{D}\tilde{u}_r(\cdot;\boldsymbol{\theta}(t))}{d\boldsymbol{\theta}}, \frac{d\mathcal{D}\tilde{u}_r(\cdot;\boldsymbol{\theta}(t))}{d\boldsymbol{\theta}} \right\rangle.$$

## Theoretical Results of NTK of PINNs

*Note that the following results are yet to be shown for parametric PDEs.*
Theoretical results by [3] for an **infinitely wide** FCNN with one hidden layer regarding NTK for the PINNs method show that under the assumption of uniformly bounded parameters $\boldsymbol{\theta}(t)$ and loss function $\mathcal{L}(\boldsymbol{\theta})$ during training, as well as a smooth activation function $\sigma(\cdot)$ with all derivatives bounded up to the 4th order, the following holds:

1. $\boldsymbol{K}(0) \xrightarrow{P} \boldsymbol{\Theta}^*$, as $n_1 \to \infty$, where $\boldsymbol{\Theta}^*$ is deterministic.
2. The kernel $\boldsymbol{K}(t)$ also stays asymptotically constant during training, i.e.

$$\lim_{P\to\infty}\sup_{t\in[0,T]}\|\boldsymbol{K}(t) - \boldsymbol{K}(0)\|_2 = 0 \text{ as } n_1 \to \infty.$$

## NTK Eigenvalue Numerical Results

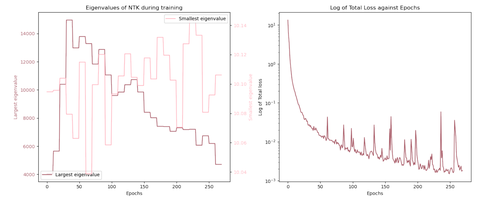


Figure 3. DPDE with BSM localisation NTK eigenvalues and total training loss.

The largest NTK eigenvalues seem to characterise how fast the training loss decreases during training. The components of the target function that correspond to the eigenvectors of the larger eigenvalues seem to be the main focus of the learning. The results are consistent with the theory and this implies theoretical results by [3] made for infinitely large NN work for finite NNs.

## References

[1] Kathrin Glau and Linus Wunderlich. The deep parametric pde method and applications to option pricing. *Applied Mathematics and Computation*, 432:127355, 2022.

[2] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[3] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.


ahw864@qmul.ac.uk