VPINN & *hp*-VPINN APPROACH TO 1-D BLACK SCHOLES EQUATION

A Project Report Submitted

in Partial Fulfilment of the Requirements

for the Degree of

BACHELOR OF TECHNOLOGY

in

Mathematics and Computing

by

Roshan Kumar (190123050) and

Kshitij Singhal (190123032)

to

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

April 2023

CERTIFICATE

This is to certify that the work contained in this project report entitled "VPINN & *hp*-VPINN APPROACH TO 1-D BLACK SCHOLES EQUATION" submitted by Roshan Kumar (Roll No.: 190123050) & Kshitij Singhal (Roll No.: 190123032) to the Department of Mathematics, Indian Institute of Technology Guwahati towards partial requirement of Bachelor of Technology in Mathematics and Computing has been carried out by him/her under my supervision.

It is also certified that this report is a survey work based on the references in the bibliography.

OR

It is also certified that, along with literature survey, a few new results are established/computational implementations have been carried out/simulation studies have been carried out/empirical analysis has been done by the student under the project.

Turnitin Similarity: 36 %

Guwahati - 781 039 (Prof. Rajen Kumar Sinha)

November 2016 Project Supervisor

ABSTRACT

The focus of this project is to solve the one-dimensional Black-Scholes equation using innovative methods such as Variational physics-informed neural networks and *hp*-Variational physics-informed neural networks.

To begin with, we discuss our Phase-1 work, where we solved the Black-Scholes equation using the Physics Informed Neural Network (PINN). We then introduce VPINN and *hp*-VPINN, highlighting their unique features and advantages over traditional neural networks.

We then delve into the mathematical formulation of VPINN and *hp*-VPINN, providing a detailed overview of the loss function used for both methods. Additionally, we discuss the implementation details, including the selection of appropriate test functions.

Moving on, we demonstrate the effectiveness of VPINN and *hp*-VPINN by solving Poisson's equation, before extending the approach to solve the Black-Scholes equation. Finally, we compare the results of *hp*-VPINN, VPINN, and PINN, analyzing their performance in terms of accuracy and efficiency.

Overall, this project highlights the potential of Variational physics-informed neural networks and *hp*-Variational physics-informed neural networks in solving complex PDEs such as the Black-Scholes equation. The findings have significant implications for the field of finance and provide a solid foundation for future research in this area

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Phase-1

In Phase-1, we solved the Black Scholes 1-D equation with the help of the Deep Galerkin Method(DGM). The DGM uses a deep neural network to solve the Black Scholes. A loss function is created using this parameterization to penalize the fitted function's departures from the required differential operator and boundary conditions. The method uses computational graphs and the backpropagation algorithm covered in the previous chapter to effectively compute the differential operator while simplifying evaluating the boundary conditions. The optimization is carried out via stochastic gradient descent, and the network uses points randomly picked from the region where the function is defined for the training data. The key finding of this method is that the training data is made up of randomly selected points inside the domain of the function. Without the computational barrier associated with grid-based approaches, the neural network "learns" the function by selecting mini-batches from various domain areas and successfully processing these small batches. This avoids the problem with the later method, which is the curse of dimensionality.

1.2 PINN & VPINN

As we know, a neural network, particularly a deep neural network, can be constructed to establish a relationship $u_{NN}(x) : \Omega \to \mathbb{R}$ between a high-dimensional input $x \in \Omega \subset \mathbb{R}^d$, where d is a positive integer, and the output $u_{NN}(x) \in \mathbb{R}$ through algebraic operations and nonlinear mapping. One of the key advantages of deep neural networks is that they can represent a vast array of functions using a relatively less number of parameters.

The fundamental challenge with neural networks (NNs) is that they lack knowledge of any underlying physical laws. To overcome this issue, a new type of NN called a physics-informed neural network (PINN), which incorporates the underlying mathematical model into the network architecture. In order to make sure the network output complies with the mathematical model, the loss function includes additional terms that act as constraints. However, this method necessitates a considerable amount of training data. The physics-informed approach addresses this problem by replacing the need for vast amounts of data with infusing information from the underlying physics, whereby the network output satisfies the corresponding mathematical model at specific points with a small penalty cost. Therefore, the network only requires a minimal number of costly training data.

To solve partial differential equations (PDEs), physics-informed neural networks (PINNs) use automatic differentiation to penalize the PDE in the loss function at a random set of points within the domain of interest. In this research, we introduce a Petrov-Galerkin version of PINNs by selecting the trial space as the space of neural networks and the test space as the space of

Legendre polynomials based on the nonlinear approximation of deep neural networks. We build a variational physics-informed neural network (VPINN) that formulates the variational residual of the PDE using the DNN approximation by adding the variational form of the problem into the network's loss function. Furthermore, we integrate by parts the integrand in the variational form to reduce the order of the differential operators represented by the neural networks, thereby reducing the training cost while increasing accuracy compared to PINNs that use delta test functions. Our goal is to solve the Black-Scholes Equation using VPINN.

1.3 *hp*-VPINN

We propose a comprehensive framework for *hp*-variational physics-informed neural networks (*hp*-VPINNs), which utilize both shallow and deep neural networks and *hp*-refinement via domain decomposition and projection onto high-order polynomials. In this framework, the *trial space* is defined globally over the entire computational domain as the space of the neural network, while the *test space* consists of piecewise polynomials. The *hp*-refinement technique involves a *global approximation with a local learning* algorithm that enables efficient localization of the network parameter optimization. We demonstrate the effectiveness of *hp*-VPINNs in terms of accuracy and training cost through various numerical examples, including function approximation and solving differential equations.
Domain decomposition offers the potential to use separate networks in each sub-domain and assign optimization to a specific computer node. The overall solution is then reconstructed by combining the solutions from individual networks in each sub-domain. This parallelization strategy can significantly

reduce the total computational cost, but it requires special treatment at the sub-domain boundaries to ensure proper communication between individual networks and optimize the total loss function. However, in the current hp-VPINN formulation, a single DNN is still used to approximate the solution over the entire computational domain, even though the domain is decomposed into several sub-domains. This approach avoids the challenge of interface treatment, but parallelization may not be straightforward as there is only one loss function associated with the DNN.

Chapter 2

Mathematical formulation

In this chapter, we will see the mathematical details for VPINN and *hp*-VPINN. We first start with PINN then we develop VPINN from PINN and lastly we will present *hp*-VPINN method. We start with the following problem:

$$\mathcal{L}^q u(x,t) = f(x,t), \qquad (x,t) \in \Omega \times (0,T] \tag{2.1}$$

$$u(x,t) = h(x,t), \qquad (x,t) \in \partial\Omega \times (0,T] \tag{2.2}$$

$$u(x,0) = g(x), \qquad x \in \Omega$$

where the bounded domain $\Omega \subset \mathbb{R}^d$ with boundaries $\partial\Omega, T > 0$, and $u(x,t) : \Omega \times [0,T] \to \mathbb{R}$ describes the underlying physical phenomena modeled by the above equation. The forcing $f(x,t)$ is some known external excitation, and $\mathcal{L}^q$ contains differential (or integro-differential) operators with a parameter q. A proper numerical method obtains the approximate solution $\tilde{u}(x,t) \approx u(x,t)$ to the above set of equations while satisfying the boundary/initial conditions is a critical key in the stability and convergence of the method.

2.1 Mathematical formulation of PINNs

To get the solution of equations (**2.1**) and (**2.2**), we can use a deep neural network as an approximated solution, i.e., $u(x,t) \approx \tilde{u}(x,t) = u_{NN}(x,t;w,b)$ where u_{NN} is a output we get from neural network. {w,b} is the weights and biases , respectively. This is known as physics-informed neural network (PINN). The PINN technique forces the network output to meet its associated mathematical model in the interior domain and at boundaries, infusing the governing equation into the network. We will define the strong form residual, the boundary residual, and the initial residual as follow:

$$\begin{aligned}
r(\tilde{u}) &= \mathcal{L}^q u(\tilde{u}) - f, && \forall(x,t) \in \Omega \times (0,T], \\
r_b(\tilde{u}) &= \tilde{u} - h, && \forall(x,t) \in \partial\Omega \times [0,T], \\
r_0(\tilde{u}) &= \tilde{u} - g, && \forall(x,t) \in \Omega \times \{t=0\}.
\end{aligned} \tag{2.3}$$

Subsequently, we define the *strong-form loss function* as

$$L^s = L_r^s + L_b + L_0, \tag{2.4}$$

$$L_r^s = \frac{1}{N_r}\sum_{i=1}^{N_r} |r(x_r^i, t_r^i)|^2,$$

$$L_b = \tau_b \frac{1}{N_b}\sum_{i=1}^{N_b} |r_b(x_b^i, t_b^i)|^2,$$

$$L_0 = \tau_0 \frac{1}{N_0}\sum_{i=1}^{N_0} |r_0(x_0^i)|^2$$

τ_b, τ_0 denote the weight coefficients in the loss function. we use superscript s to denote the loss function associated with strong form of residual. In this setting, we pose the problem of solving (**2.1**) and (**2.2**) as:

$$find\ \ \tilde{u}(\mathbf{x}) = u_{NN}(\mathbf{x};\mathbf{w}^*,\mathbf{b}^*)\ such\,that\ \ \{\mathbf{w}^*,\mathbf{b}^*\} = argmin(L^s(\mathbf{w},\mathbf{b})).$$

Three parts comprise the strong-form loss function: the first penalizes the strong-form residual at some interior (penalizing) points x_r, the second penalizes the solution at the boundary points x_b, and the third penalizes the solution at initial points x_0, resulting in a penalty-like method.

2.2 Mathematical formulation of VPINNs

Here, we propose a novel approach for the variational physics-informed neural network inspired by the mathematical problem's variational form. We represent the approximation as $\tilde{u}(\mathbf{x},\mathbf{t}) = u_{NN}(\mathbf{x},\mathbf{t};\mathbf{w},\mathbf{b})$ with weights and biases $\{\mathbf{w},\mathbf{b}\}$, respectively. Furthermore, we select a suitable test function v_j $(1 \leq j \leq K)$ and multiply equation (**2.1**) by v_j before integrating over the entire domain to obtain the variational form.

$$(\mathcal{L}^q u_{NN}(\mathbf{x},\mathbf{t}), v_j)_\Omega = (f(\mathbf{x}), v_j)_\Omega \tag{2.5}$$

$$\begin{aligned} u(x,t) &= h(x,t), && (x,t) \in \partial\Omega \times (0,T] \\ u(x,0) &= g(x), && x \in \Omega \end{aligned} \tag{2.6}$$

To ensure that the approximation $\tilde{u}$ satisfies Eqs. (2.1)-(2.2) and recovers the exact solution, we use residuals as a measure of its accuracy. The residuals are obtained by subtracting the approximate solution from the exact one, and ideally, they should be identically zero. To obtain the weighted integrals of the residuals, we project them onto a suitable space of test functions V

and set them to zero. This approach leads to the variational form of the problem. To implement it, we select a set of test functions v_j from V and use them to construct the weighted integrals

We define the variational loss function to be:

$$
\begin{aligned}
L^v &= L_r^v + L_u + L_0, \qquad (2.7)\\
L_r^v &= \frac{1}{K}\sum_{j=1}^{K} |\mathcal{R}_j|^2,\\
\textit{where} \quad \mathcal{R}_j(\tilde{u}) &= \int_{\Omega\times(0,T]} r(\tilde{u}) v_j dxdt = 0\\
L_b &= \tau_b \frac{1}{N_b}\sum_{i=1}^{N_b} |r_b(x_b^i, t_b^i)|^2,\\
L_0 &= \tau_0 \frac{1}{N_0}\sum_{i=1}^{N_0} |r_0(x_0^i)|^2
\end{aligned}
$$

τ_b, τ_0 are the weight coefficients in the loss function and superscript v is used to denote to the loss function associated with the variational form of residual. In this setting, we pose the problem of solving (2.6) and (2.7) as:

$$
\textit{find} \;\; \tilde{u}(\mathbf{x}) = u_{NN}(\mathbf{x}; \mathbf{w}^*, \mathbf{b}^*) \;\textit{such that}\; \{\mathbf{w}^*, \mathbf{b}^*\} = \textit{argmin}(L^v(\mathbf{w},\mathbf{b})). \qquad (2.8)
$$

In contrast to the physics-informed neural network (PINN), the computation of integrals in the loss function presents a significant challenge for the variational physics-informed neural network (VPINN). Due to the compositional structure of neural networks, it is almost impossible to obtain an analytical expression for these integrals. Even for networks with two hidden layers, computing these integrals is not entirely realistic, as there is no explanation for their quadrature rules. To overcome this challenge, we first employ a shallow network with only one hidden layer, enabling us to perform the derivations analytically, albeit only in a few specific cases. We then propose deep VPINNs by combining multiple hidden layers, taking into account deep networks. However, in such cases, we must use numerical integration techniques to compute the integrals in the variational residuals.

2.3 Mathematical formulation of *hp*-VPINN

The hp-VPINN formulation is based on the following localized test functions, defined over non-overlapping subdomains V_k, $k = 1, 2, ..., N_{sd}$, of a partition of the set V ($\Omega \times (0, T]$ or Ω in this work). The test function defined on a subset $V_k \subset V$ reads:

$$v_k = \begin{cases} v \neq 0 & over \quad V_k \\ 0 & over \quad V_k^c \end{cases} \qquad V_k \cup V_k^c = V$$

We can use the approach described above to derive a sub-domain method, which involves choosing a non-vanishing test function $\overline{v}$ that is a polynomial of a certain order, as determined in practice.

We define the elemental variational residual as

$$\mathcal{R}^{(e)} = (\mathcal{L}^q u_{NN} - f, v)$$

which is enforced for the admissible local test function within element e.

Subsequently, we define the variational loss function as

$$L^v = \sum_{e=1}^{N_{el}} \frac{1}{K^{(e)}} \sum_{k=1}^{K^{(e)}} |\mathcal{R}_k^{(e)}|^2 + \tau_b \frac{1}{N_b} \sum_{i=1}^{N_b} |r_b(x_b^i, t_b^i)|^2 + \tau_0 \frac{1}{N_0} \sum_{i=1}^{N_0} |r_0(x_0^i)|^2, \quad (2.9)$$

where $K^{(e)}$ is the total number of test functions in element e, the term $\mathcal{R}_k^{(e)}$ is the kth entry of the corresponding tensor associated with element e, and r_b and r_0 have the same form as in (2.3)
When strong-form residuals are projected onto test functions, two major truncation and numerical integration errors are introduced into the existing approximation and generalization errors of DNNs. Increasing the number of test functions to eliminate truncation errors can lead to complications in the loss function and raise the likelihood of optimization failure. Shallow networks can obtain the variational residual analytically, effectively eliminating the numerical integration error. However, the compositional structure of hidden layers in DNNs makes it almost impossible to compute these integrals analytically. To address this, numerical integration techniques such as Gauss quadrature rules can be used, but this introduces new challenges in developing and analyzing numerical integration methods for functions represented by DNNs. To deal with high-dimensional problems, numerical approaches such as quasi-Monte Carlo integration or sparse grid quadratures can be used to avoid the curse of dimensionality.

Chapter 3

Implementation

3.1 One-dimensional Poisson's equation

We will provide a detailed discussion on the derivation of our proposed formulation, *hp*-VPINN, for the one-dimensional problem. The problem can be defined as finding the function $u(x)$, where $u(x) : \Omega \to \mathbb{R}$ and $\Omega = [-1, 1]$, that satisfies Poisson's equation:

$$-\frac{d^2u(x)}{dx^2} = f(x), \tag{3.1}$$

with boundary conditions $u(-1) = g$ and $u(1) = h$.
We approximate solution be $u(x) \approx \tilde{u}(x) = u_{NN}$, then the strong-form residual becomes

$$r(x) = -\frac{d^2u_{NN}(x)}{dx^2} - f(x), \quad x \in (-1, 1), \tag{3.2}$$

$$r_b(x) = u_{NN}(x) - u(x), \quad x = \pm 1. \tag{3.3}$$

To apply hp-VPINN method to the Poisson's equation, we first divide the domain $\Omega = [-1, 1]$ into non-overlapping elements $\Omega_e = [x_{e-1}, x_e]$ using a domain decomposition grid $-1 = x_0, x_1, ..., x_{N_{el}} = 1$. Next, we choose a set of localized non-overlapping test functions $v_k(x)$, which are high-order polynomials defined as $P_{k+1}(x) - P_{k-1}(x)$, where $P_k(x)$ is the Legendre polynomial of order k. The variational residual then becomes

$$\mathcal{R}_k = \sum_{e=1}^{N_{el}} \mathcal{R}_k^{(e)} = \sum_{e=1}^{N_{el}} \int_{x_{e-1}}^{x_e} \left(-\frac{d^2 u_{NN}(x)}{dx^2} - f(x) \right) v_k^{(e)}(x) dx \tag{3.4}$$

hp-variational loss function for each case takes the form

$$L = \sum_{e=1}^{N_{el}} \frac{1}{K^{(e)}} \sum_{k=1}^{K^{(e)}} \left| R_k^{(e)} \right|^2 + \frac{\tau_b}{2} \left| u_{NN}(-1) - g \right|^2 + \left| u_{NN}(1) - h \right|^2 \tag{3.5}$$

The number of test functions in each element is denoted by $K^{(e)}$.

3.1.1 Example

$$u_{\text{exact}}(x) = 0.1 \sin(8\pi x) + \tanh(80x) \tag{3.6}$$

To construct our model, we implemented a fully connected neural network with four layers (l=4) and 20 neurons in each layer (N=20), all of which were activated with the sine function. Additionally, we utilized up to order 60 Legendre polynomials and integrated them using 80 Gauss-Lobatto quadrature points and weights per element. The results are shown in Fig 3.2 and Fig 3.1.

3.2 Two-dimensional Poisson's equation

We will provide a detailed discussion on the derivation of our proposed formulation, hp-VPINN, for the two-dimensional problem. The problem can be defined as finding the function $u(x, y)$, where $u(x, y) : \Omega \to \mathbb{R}$ and

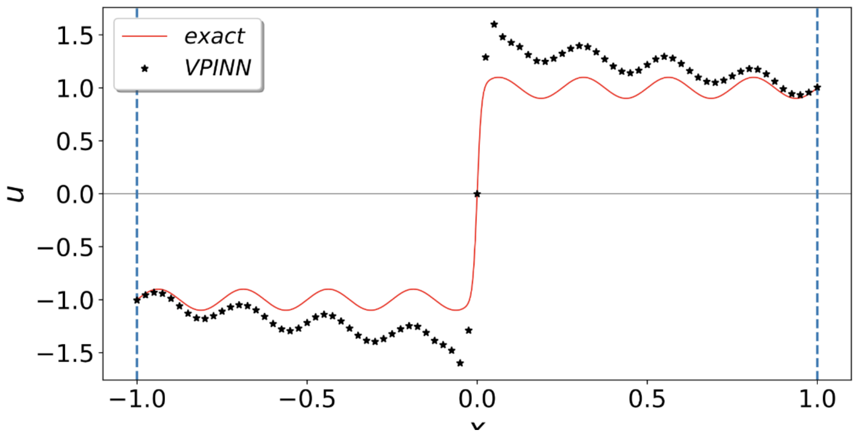


Figure 3.1: VPINN approximation to One-dimensional Poisson's equation

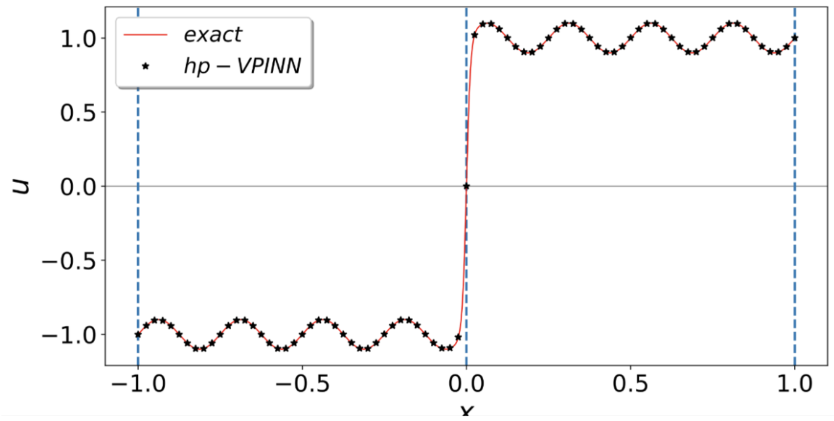


Figure 3.2: hp-VPINN approximation to One-dimensional Poisson's equation with 3 elements

$\Omega = [-1,1] \times [-1,1]$, that satisfies the two-dimensional Poisson's equation:

$$\nabla^2 u(x,y) = f(x,y), \tag{3.7}$$

Assuming the approximate solution is given by $u(x,y) \approx \tilde{u}(x,y) = u_{NN}(x,y)$, the corresponding strong-form residual can be expressed as:

$$r(x,y) = \nabla^2 u_{NN}(x,y) - f(x,y), \tag{3.8}$$

$$r_b(x,y) = u_{NN}(x,y) - h(x,y), \tag{3.9}$$

To construct our model, we define a grid in the x and y dimensions as follows: $-1 = x_0, x_1, ..., x_{N_{el_x}} = 1$ and $-1 = y_0, y_1, ..., y_{N_{el_y}} = 1$. Next, we divide the domain Ω into structured sub-domains by constructing non-overlapping elements $\Omega_{e_x,e_y} = [x_{e_x-1}, x_{e_x}] \times [y_{e_y-1}, y_{e_y}]$, where $e_x = 1,2,...,N_{el_x}$ and $e_y = 1,2,...,N_{el_y}$. By doing so, the resulting variational residual can be expressed as:

$$\mathcal{R}_{k_1k_2} = \sum_{e_x=1}^{N_{el_x}} \sum_{e_y=1}^{N_{el_y}} \mathcal{R}_{k_1k_2}^{(e_xe_y)} \tag{3.10}$$

$$\mathcal{R}_{k_1k_2}^{(e_xe_y)} = \sum_{e_x=1}^{Nel_x} \sum_{e_y=1}^{Nel_y} \int_{x_{e_x-1}}^{x_{e_x}} \int_{y_{e_y-1}}^{y_{e_y}} \left(\nabla^2 u_{NN}(x,y) - f(x,y)\right) \phi_{k_1}^{(e_x)}(x) \phi_{k_2}^{(e_y)}(y) dx dy \tag{3.11}$$

We can also see that the non-vanishing part of $\phi_{k_1}(x)$ and $\phi_{k_2}(y)$ has a similar structure as the earlier one-dimensional case. Therefore, the local test functions $v_{k_1k_2}^{(e)}(x,y)$ have compact support over Ω_e. Therefore, for all elements, we have

$$\phi_{k1}^{(e_x)}(x_{e_x-1}) = \phi_{k_1}^{(e_x)}(x_{e_x}) = 0, \qquad k_1 = 1, 2, \ldots, K_1, \ldots, K_2$$
$$\phi_{k1}^{(e_y)}(x_{e_y-1}) = \phi_{k_1}^{(e_y)}(y_{e_y}) = 0, \qquad k_2 = 1, 2, \ldots, K_2, \ldots, K_2$$

We define the variational loss function as

$$L_v = \sum_{e_x=1}^{N_{el_x}} \sum_{e_y=1}^{N_{el_y}} \frac{1}{K_1 K_2} \sum_{K_1} K_1 K_2 |\mathcal{R}_k^{(e_x e_y)}|^2 + \tau_b \frac{1}{N_b} \sum_{i=1}^{N_b} |r_b(x_b, t_b)|^2, \tag{3.12}$$

3.2.1 Example

$$u_{exact}(x, y) = (0.1 \sin(2\pi x) + \tanh(10x)) \times \sin(2\pi y) \tag{3.13}$$

For this particular case, we opt to utilize a wider neural network architecture, where we set the number of layers to l = 3, the number of neurons in each layer to N = 20, and employ the tanh activation function. Additionally, we utilize Legendre polynomials in both the x and y directions and perform the integral in each element by employing the appropriate number of Gauss quadrature points using the tensor product rule. The results are shown in Fig 3.3 and Fig 3.4.

In this particular case, we find that domain decomposition does not result in any significant improvement in the approximation, as the point-wise error remains of the same order across all formulations. However, this technique can still be useful for parallel computation purposes. By dividing the domain into sub-domains, each one can be solved on a separate computer node, thereby reducing the total computational costs.

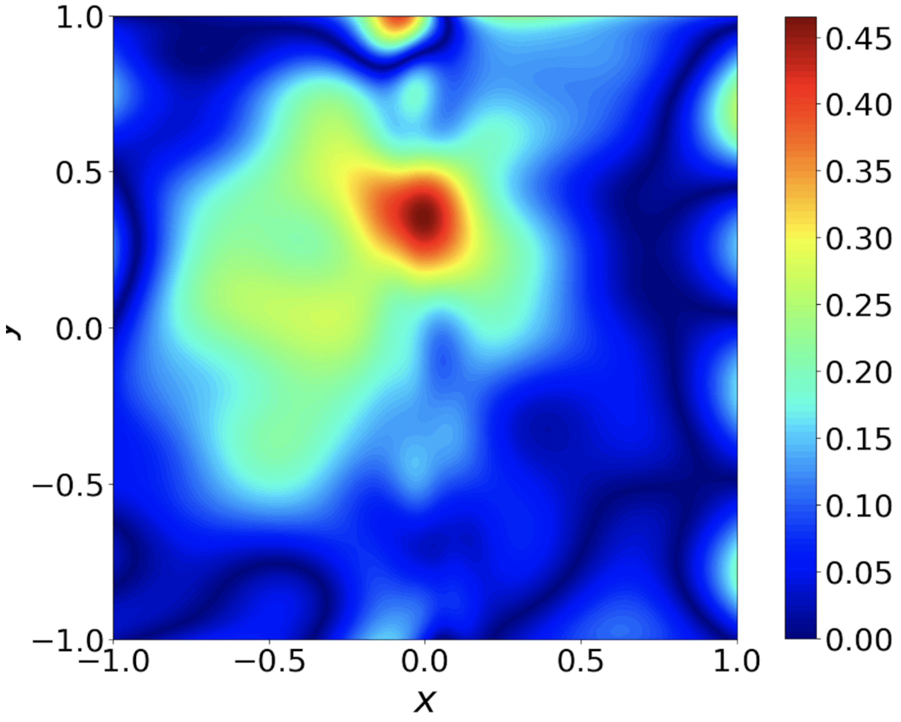


Figure 3.3: hp-VPINN point-wise error for Poisson-2D

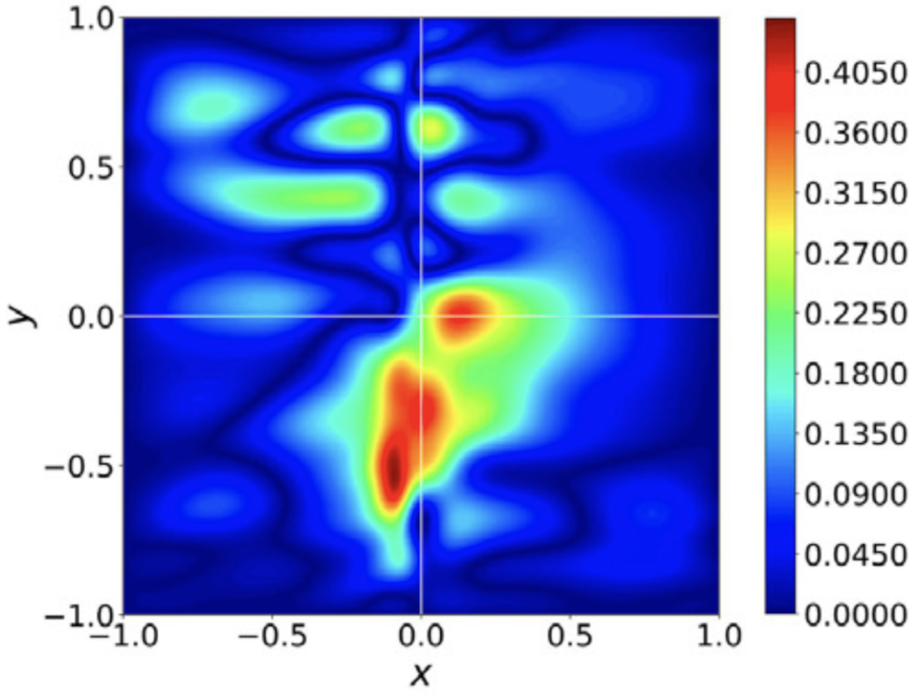


Figure 3.4: VPINN point-wise error for Poisson-2D

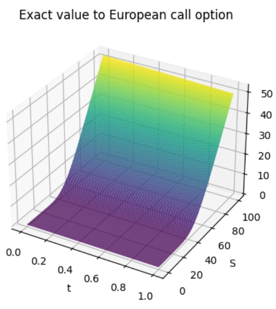
3.3 European Call Option

One-Dimensional Black-Scholes PDE

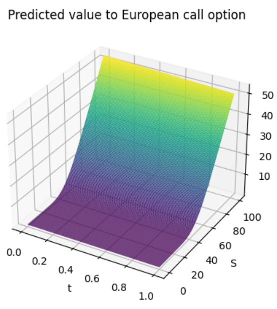
$$\begin{aligned}\partial_t u(t,x) + rx.\partial_x u(t,x) + \frac{1}{2}\sigma^2 x^2.\partial_{xx} u(t,x) = r.u(t,x)\\ u(T,x) = H(x)\end{aligned}$$

where $H(x) = max(0, x - K)$

We consider the function $u(t,x) : \Omega \to \mathbb{R}$, where $\Omega = [0,1] \times [0,100]$, and approximate the solution using $\widetilde{u}(t,x) = u_{NN}(t,x)$. As time is a variable in the transient problem, we treat it as another dimension, resulting in variational residuals that are similar to the previous examples. To partition the space-time domain Ω, we construct a temporal grid $0 = t_0, t_1, ..., t_{N_{el_t}} = 1$ and a spatial grid $0 = x_0, x_1, ..., x_{N_{el_x}} = 100$, and divide Ω into structured, non-overlapping sub-domains (elements) $\Omega_{e_t e_x} = [t_{e_t-1}, t_{e_t}] \times [x_{e_x-1}, x_{e_x}]$, with $e_t = 1, 2, ..., N_{el_t}$ and $e_x = 1, 2, ..., N_{el_x}$. To employ the hp-VPINN formulation, we construct a fully connected neural network with a tanh activation function and specify the following parameters: $l = 3$, $N = 5$, $K_1 = K_2 = 5$, and $Q = 10 \times 10$. In addition, we use Legendre test functions in both space and time directions. The results are shown in Fig 3.5.



(a) Value calculated using exact solution of equation



(b) Value calculated using hp-VPINN

Figure 3.5: 1-D Black Scholes equation

Chapter 4

Conclusion

Our research paper presented a novel approach to solving the Black-Scholes equation using Variational Physics-Informed Neural Network (VPINN) and hp-VPINN. The Black-Scholes equation is a well-known partial differential equation that has played a significant role in financial mathematics, providing a theoretical foundation for pricing options and other financial derivatives. However, solving this equation can be challenging due to its complex nature and the high dimensionality of the input space. We placed our focus on the projection onto the space of higher-order polynomials. The optimal choice of test functions in the current developments is an important open question and requires further analysis. Overall, our study demonstrates the potential of VPINN and hp-VPINN in solving the Black-Scholes equation and other complex partial differential equations in finance and beyond. Our work provides a foundation for future research in this area and opens up new opportunities for deep learning to solve complex problems in various fields.

Bibliography

[1] Ali Al-Aradi. *Solving Nonlinear and High-Dimensional Partial Differential Equations via Deep Learning.* arXiv:1811.08782, 2020.

[2] Ehsan Kharazmia, Zhongqiang Zhangb, and George E.M. Karniadakis. *hp-VPINNs: Variational physics-informed neural networks with domain decomposition.* Computer Methods in Applied Mechanics and Engineering, 2021.

[3] Ehsan Kharazmia, Zhongqiang Zhangb, and George E.M. Karniadakis. *VPINNs: Variational physics-informed neural networks for solving partial differential equations.* arXiv:1912.00873, 2021.