Numerical Solutions of Variational Inequalities by Neural Network

KHAN SADMAN SAMIN
SID: 56318572
DEPARTMENT OF MATHEMATICS
CITY UNIVERSITY OF HONG KONG
SUPERVISOR: PROF. ZHANG SHUN

Contents

1. Introduction:

The project focuses on the exploration and implementation of two innovative techniques, namely the Physics-Informed Neural Network (PINN) and Deep-Ritz method, for solving variational inequalities. Variational inequalities arise in numerous fields, including optimization, game theory, and mechanics, and are characterized by finding the equilibrium points of certain constrained optimization problems.

Traditional numerical methods for solving variational inequalities often face challenges related to convergence, computational efficiency, and handling complex constraints. The PINN and Deep-Ritz methods offer promising alternatives by leveraging deep learning and function approximation techniques.

Through this project, we aim to explore and implement both the PINN and Deep-Ritz methods for solving variational inequalities. By comparing their performance, computational efficiency, and robustness, we seek to gain insights into the strengths and limitations of each approach.

The outcomes of this project will contribute to the understanding and advancement of deep learning techniques in the context of variational inequalities. Additionally, the findings will have implications for various fields where variational inequalities play a crucial role, including optimization, economics, and engineering.

2. Theoretical Foundations

2.1 Variational Inequalities

Variational inequalities are mathematical problems that involve finding a solution within a given set, such that certain inequalities are satisfied. They arise in various fields, including optimization, economics, engineering, and physics, where the goal is to determine an equilibrium state or an optimal solution subject to certain constraints. Variational inequalities provide a framework for modeling and analyzing a wide range of problems involving inequalities.

Mathematical Formulation:

The general mathematical formulation of a variational inequality can be stated as follows:

Find x^* in a set X such that for all x in X, the following inequality holds:

$$F(x^*)^T(x - x^*) \geq 0$$

Here, $F: X \rightarrow \mathbb{R}^n$ is a mapping, X is a nonempty closed convex set, x^* is the unknown solution, and x is any point in X. The inequality $F(x^*)^T(x - x^*) \geq 0$ represents the variational inequality condition.

Variational inequalities possess several important properties:

1. Existence: Under certain conditions on the mapping F and the set X it can be guaranteed that a solution x^* exists for the variational inequality problem.

2. Uniqueness: The solution to a variational inequality may not be unique in general. However, under certain assumptions on F and X, uniqueness of the solution can be established.

3. Continuity: Variational inequalities exhibit continuity properties with respect to the mapping F and the set X. This property is important for the stability and convergence analysis of numerical methods.

4. Relation to optimization: Variational inequalities can be seen as a generalization of optimization problems. In fact, optimization problems can be formulated as a special case of variational inequalities by considering a specific set and mapping.

2.2. Obstacle Problems

Obstacle problems are mathematical problems that involve finding the solution to a partial differential equation (PDE) subject to inequality constraints known as obstacles. These problems arise in various fields, including physics, engineering, and finance, where the presence of obstacles or constraints influences the behavior of the system under consideration.

The general mathematical formulation of an obstacle problem can be stated as follows:

Find u in a domain D such that it satisfies the following conditions:

1. PDE Constraint: The solution u satisfies a given PDE equation, such as an elliptic, parabolic, or hyperbolic equation, subject to certain boundary and initial conditions.

2. Obstacle Constraint: The solution u must fulfill an inequality constraint imposed by one or more obstacles in the domain. The obstacles can be represented as functions or sets, and the constraint ensures that u remains above or below the obstacles.

Let $\Omega \subset \mathbb{R}^d$ be a bounded Lipschitz domain with the boundary $\partial\Omega$. The obstacle problem is to find the equilibrium position u of an elastic membrane under the action of the vertical force f. The membrane is fixed on the boundary $\partial\Omega$ with the function h and must lie over the obstacle g with $g \leq h$ on $\partial\Omega$. The differential form of the obstacle problem is

$$\begin{gathered} -\Delta u \geq f \text{ in } \Omega \\ u \geq g \text{ in } \Omega \\ (-\Delta u - f)(u - g) = 0 \text{ in } \Omega \\ u = h \text{ on } \partial\Omega \end{gathered} \tag{1}$$

Let's consider $f = 0, h = 0$ and φ be the function of the obstacle. Thus, it becomes a problem of minimization of the following energy functional

$$F(u) = \int_{\Omega} |\nabla u|^2 \, dx$$

among all functions u satisfying $u \geq \varphi$ in Ω, for a given obstacle φ. The Euler-Lagrange equation of such a minimization problem is

$$\begin{gathered} -\Delta u \geq 0 \text{ in } \Omega \\ u \geq \varphi \text{ in } \Omega \\ \Delta u = 0 \text{ in } \{u \geq \varphi\} \end{gathered} \tag{2}$$

In other words, the solution u is above the obstacle φ, is harmonic whenever it does not touch the obstacle, and it is superharmonic everywhere. The domain Ω will be split into two regions: one in which the solution u is harmonic, and one in which the solution equals the obstacle. The latter region is known as the contact set $\{u \geq \varphi\} \subset \Omega$. The interface that separates these two regions is the free boundary.

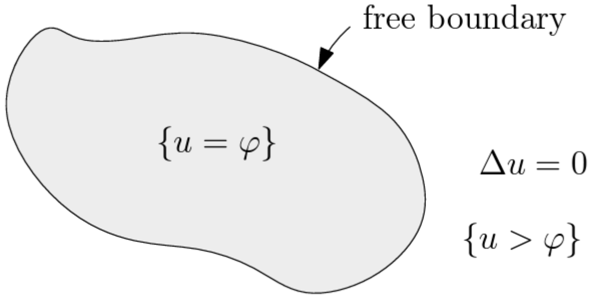


Figure 1: The contact set and the free boundary in the classical obstacle problem

Let $V = H^1(\Omega)$ and $U = \{v \in V : v = h \; on \; \partial\Omega\}$. We denote by $K = \{v \in V : v \geq g \; in \; \Omega, v = h \; on \; \partial\Omega\}$ as the set of admissible displacements. Assume that $f \in L_\infty(\Omega), g \in W_\infty^2(\Omega), h \in W_\infty^2(\partial\Omega)$. Problem (1) can be rewritten in the form of energy minimization

$$\text{Find } u \in K: \quad J(u) \leq J(v), \quad \forall v \in K, \tag{3}$$

Where $J(v) = a(v, v) - (f, v)$ with the quadratic form $a(v, v) = \frac{1}{2}\int_\Omega |\nabla v|^2 dx$.

The solution of (3) is characterized by the elliptic variational inequality

$$u \in K: \quad \int_\Omega \nabla u \cdot \nabla(v - u) dx \geq \int_\Omega \mathrm{f}(v - u) dx, \qquad \forall v \in K \tag{4}$$

The existence of such a minimizer u in (3) is assured by considerations of Hilbert space theory. Various numerical methods have been proposed for solving obstacle problems, the vast majority of which focus on approximation solutions to the weak variational inequality, such as Galerkin least squares finite element method [1], multigrid algorithm [2], piecewise linear iterative algorithm [3], the first-order least-squares method [4], the level set method [5], and dynamical functional particle method [6].

2.3 Deep Neural Networks

Neural networks are powerful mathematical models that can approximate complex functions by learning from data. They consist of interconnected artificial neurons, also known as nodes or units, organized in layers. The most common type of neural network used for function approximation is the feedforward neural network, characterized by the absence of feedback connections between the nodes.

Feedforward neural networks consist of an input layer, one or more hidden layers, and an output layer. Neurons in each layer perform computations on the input data. The network operates in two steps: forward propagation and activation. Forward propagation calculates neuron activations by passing input data layer by layer. Activation functions introduce non-linearity to model complex relationships. The output layer uses activations to make predictions, with activation function choice based on the problem type (regression or classification).

For $j = 1, \dots \dots, l-1$, let $N^{(j)}: \mathbb{R}^{n_{j-1}} \to \mathbb{R}^{n_j}$ be the vector-valued ridge function of the form

$$N^{(j)}\left(x^{(j-1)}\right) = \tau(\omega^{(j)} x^{(j-1)} - b^{(j)}) \qquad \text{for } x^{(j-1)} \in \mathbb{R}^{n_{j-1}}, \tag{5}$$

Where $\omega^{(j)} \in \mathbb{R}^{n_{j-1} \times n_j}$ and $b^{(j)} \in \mathbb{R}^{n_j}$ are the respective weights and bias to be determined; $x^{(0)} = x$; and $\tau(t)$ is a non-linear activation function. There are many activation functions such as ReLU, sigmoid, and hyperbolic tangent.

Let $\omega^{(l)} \in \mathbb{R}^{d \times n_{l-1}}$ and $b^{(j)} \in \mathbb{R}^{d}$ be the output weights and bias. Then a l-layer neural network generates the following set of vector fields in $\mathbb{R}^d$

$$\boldsymbol{\mathcal{M}}_N(l) = \{\omega^l \left(N^{(l-1)} \circ \cdots \circ N^{(l)}(x)\right) - b^l : \omega^{(j)} \in \mathbb{R}^{n_j \times n_{j-1}}, b^{(j)} \in \mathbb{R}^{n_j} \text{ for all } j\}, \tag{6}$$

where ° denotes the composition of functions.

Training a neural network involves adjusting weights to minimize the difference between predicted and desired outputs. Backpropagation is the algorithm for this. Steps: (1) Forward Propagation: Compute activations through the network. (2) Error Calculation: Compare network output to desired output using a loss function. (3) Backward Propagation: Propagate error backward, adjusting weights using gradients. (4) Weight Update: Use optimization algorithm (e.g., gradient descent) to update weights. (5) Iterative Training: Repeat steps until convergence, presenting data in mini batches. Neural networks learn to approximate functions and generalize by iteratively adjusting weights based on training data.

3. Physics-Informed Neural Networks (PINN) and Deep-Ritz Method

In the previous literature, two notable works have been discussed that focus on solving obstacle problems using deep neural networks.

3.1. Physics-Informed Neural Networks (PINN)

The first work by El Bahja et al. [10] extends the physics-informed neural network (PINN) framework to address obstacle-related partial differential equations (PDEs). The proposed PINNs leverage sparse and noisy data to solve these obstacle-related PDEs and demonstrate their performance in various scenarios, considering both linear and nonlinear PDEs subject to regular and irregular obstacles.

Let's consider the following obstacle problem:

$$\begin{gathered} -\Delta u \geq f \text{ in } \Omega \\ u \geq g \text{ in } \Omega \\ (-\Delta u - f)(u - g) = 0 \text{ in } \Omega \\ u = h \text{ on } \partial\Omega \end{gathered} \tag{7}$$

The loss function, when solving using PINN, is designed as:

$$Loss = \frac{1}{N_r}\sum_{i=1}^{N_r} |H(u(x_r^i;\theta) - g(x_r^i)) \cdot R(x_r^i;\theta) + ReLU(g(x_r^i) - u(x_r^i;\theta))|^2 + \frac{\alpha_b}{N_b}\sum_{i=1}^{N_b} |u(x_b^i;\theta) - h(x_b^i)|^2 \tag{8}$$

where H is the Heavyside step Function

$$H(x) = \begin{cases} 1 & if\ x \geq 0, \\ 0 & otherwise \end{cases}$$

and $R(x_r^i;\theta) = -\Delta u - f$

3.2. Deep-Ritz Method

Cheng et al. [7] presented a deep neural network-based method Deep-Ritz for solving obstacle problems. A deep neural network is employed to approximate the solution, and the convergence analysis is established by decomposing the error into approximation error, statistical error, and optimization error. The depth and width of the network affect the approximation error, the number of samples influences the statistical error, and the optimization error is captured by the empirical loss term. The proposed method is characterized by its unsupervised and meshless nature, which contributes to its wide applicability.

The algorithm for the Deep-Ritz Method is as follows:

The network depth D and width W, the penalty parameters α and β, the number of samples N and M, the initial guess of the parameter θ_1, the learning rate η, and the total number of iterations K.

```
for k=1, 2,...K do
    Randomly sample a batch $\{X_i\}_{i=1}^{N} \sim U(\Omega), \{Y_i\}_{i=1}^{M} \sim U(\partial\Omega)$
    Compute $loss_1 = \frac{1}{N}\sum_{i=1}^{N}[\frac{1}{2}\|\nabla u_\theta(X_i)\|_2^2 - f(X_i)u(X_i)]$,
        $loss_2 = \frac{1}{N}\sum_{i=1}^{N}[g(X_i) - u_\theta(X_i)]_+^2$,
        $loss_3 = \frac{1}{M}\sum_{j=1}^{M}[u_\theta(Y_i) - h(Y_i)]^2$.
    Compute $\mathcal{L}(\theta) = loss_1 + \alpha loss_2 + \beta loss_3$
    Update $\theta_{k+1} = \theta_k - \eta\nabla_\theta\mathcal{L}(\theta_k)$
end for
```

Both works emphasize the limitations of traditional numerical methods for obstacle problems, such as lack of convergence, non-differentiability, and dependence on domain discretization. The application of deep neural networks in these contexts offers promising solutions to overcome these limitations and provides a general framework for tackling a variety of obstacle-related PDEs. The use of PINNs and Deep-Ritz enables the incorporation of physics-based constraints while leveraging the expressive power of neural networks for function approximation. These methods showcase the potential of deep learning techniques in addressing obstacle problems and open up avenues for further research in this area.

4. Numerical Examples

4.1. Solving a simple 1D differential equation using Deep-Ritz

The problem: $-u''(x) = f(x)$ in $\Omega = [0, 1]$, $f(x) = 2$, with boundary conditions $u(x) = 0$ in $\partial\Omega$:

Network Architecture: For this problem, a neural network with an input dimension of 1 and three hidden layers of size 20 was proposed. The activation function used is tanh.

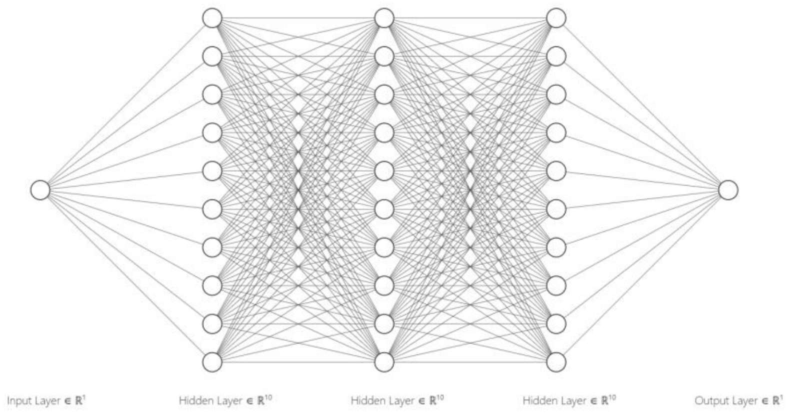
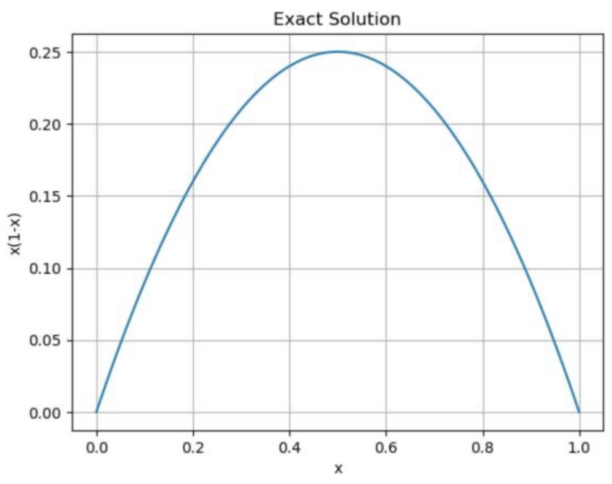

Input Layer ∈ $\mathbb{R}^1$
Hidden Layer ∈ $\mathbb{R}^{10}$
Hidden Layer ∈ $\mathbb{R}^{10}$
Hidden Layer ∈ $\mathbb{R}^{10}$
Output Layer ∈ $\mathbb{R}^1$

Figure 2: Neural Network architecture used for problem in 4.1

Loss Function: The loss function is designed based on the energy functional. It aims to minimize the energy functional $E(x;\theta) = \frac{1}{2}|\nabla_x u(x;\theta)|^2 - f(x)u(x;\ \theta)$. Additional boundary conditions are incorporated using a penalty term: $10^5 * (u(0;\ \theta)^2 + u(1;\ \theta)^2)$.



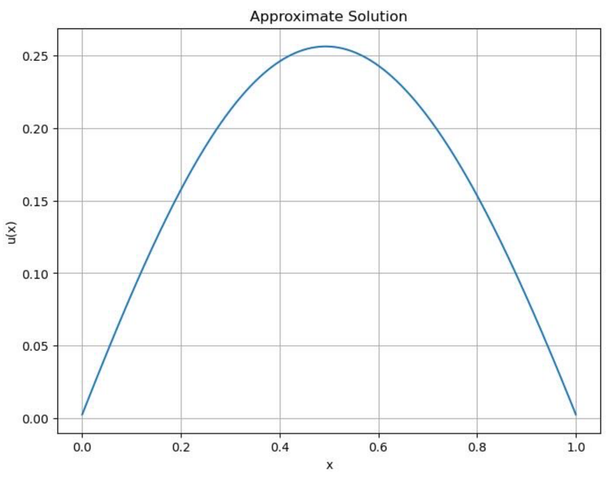


Figure 3: Solution for problem in 4.1. Exact solution (left), Approximate solution (right)

4.2. Solving $-u'' \geq 0$ with obstacle condition $u - \varphi \geq 0$ using Deep-Ritz

$\varphi = 1 - 4x^2$, $(u - \varphi)(u'') = 0, \Omega = [-1, 1]$

Network Architecture: Similar to the problem in 2, a neural network with an input dimension of 1 and three hidden layers of size 20 is employed. The activation function used is tanh. The architecture is the same as shown in Figure

Loss Function: The loss function is derived from the energy functional and includes terms to enforce the obstacle condition and the inequality constraint. The loss function is defined as follows:

$$Loss = \frac{1}{2}|\nabla_x u(x;\theta)|^2 - f(x)u(x;\theta) + 10^3 * (\max(0, \varphi - u)^2 + \max(0, u'')^2).$$

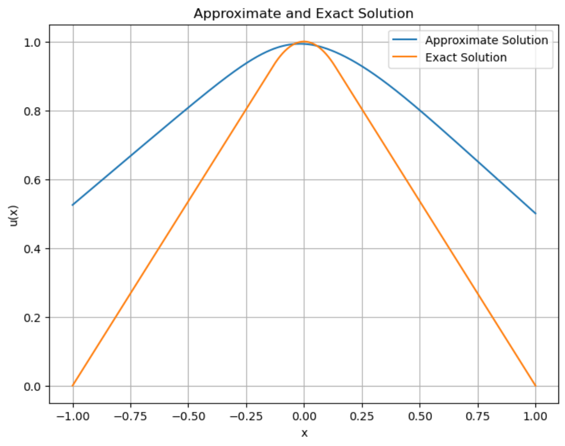


Figure 4: Solution for problem in 4.2. Exact solution (Orange), Approximate solution (Blue)

4.3. Solving a 1D Obstacle Problem using Deep-Ritz

The problem was taken from [7]:

$$-u'' \geq 0 \text{ in } \Omega$$

$$u \geq g \text{ in } \Omega$$

$$u''(u-g) = 0 \text{ in } \Omega$$

$$u = 0 \text{ on } \partial\Omega$$

where $\Omega = [0, 1]$ and

$$g(x) = \begin{cases} 100x^2 & for \quad 0 \leq x \leq 0.25, \\ 100x(1-x) - 12.5 & for \quad 0.25 \leq x \leq 0.5, \\ g(1-x) & for \quad 0.5 \leq x \leq 1.0 \end{cases}$$

The exact solution is

$$u_{exact}(x) = \begin{cases} 100 - 50\sqrt{2} & for \quad 0 \leq x \leq \frac{1}{2\sqrt{2}}, \\ 100x(1-x) - 12.5 & for \quad \frac{1}{2\sqrt{2}} \leq x \leq 0.5, \\ u_{exact}(1-x) & for \quad 0.5 \leq x \leq 1.0 \end{cases}$$

Network Architecture: The neural network architecture, used in this problem, comprises multiple fully connected layers interleaved with layer normalization and $ReLU$ activation functions. The input dimension, hidden layer size, and output dimension can be specified during instantiation (for our problem, $Input\ size = Output\ size = 1$ and $Number\ of\ hidden\ layers = 5$. Each hidden layer is set to have a size of 50). The network takes an input tensor and sequentially applies linear transformations followed by layer normalization to propagate the input through the network. The output of each layer is then passed through a cubic power ReLU activation function. The final layer maps the hidden representation to the output dimension. This architecture has been shown to effectively approximate the desired solutions for the problem at hand, demonstrating its suitability for solving obstacle problems using the Deep-Ritz method.

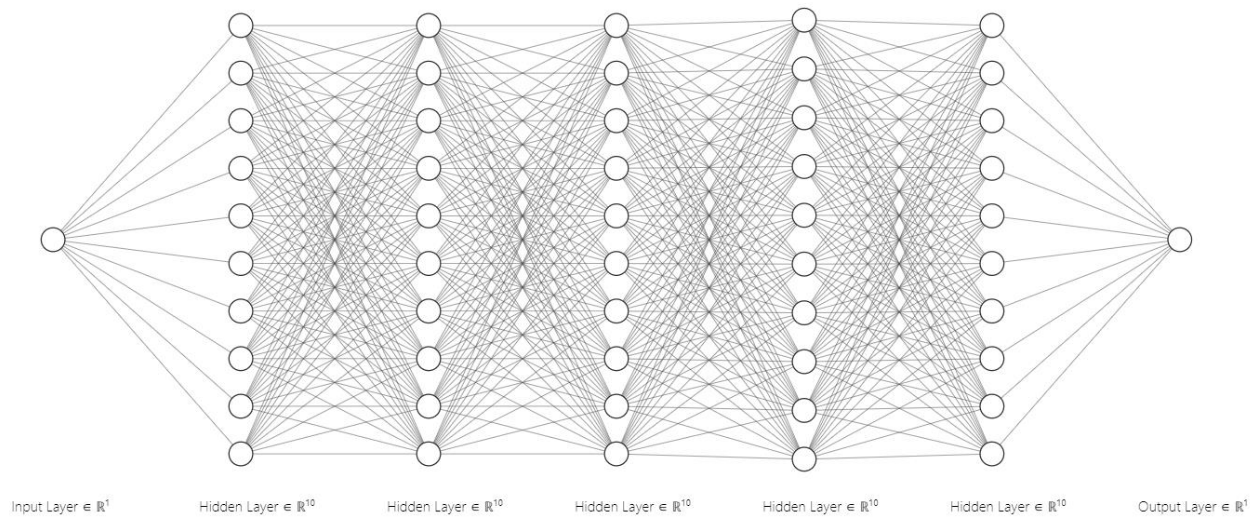


Figure 5: Neural Network architecture used for problem in 4.3.

The Adam version of SGD (Stochastic Gradient Descent) is used for optimization.

Loss Function: The loss function is based on the energy functional and incorporates the obstacle condition and the obstacle-interface compatibility equation. The loss function is defined as follows:

$$Loss = \frac{1}{2}|\nabla_x u(x;\theta)|^2 - f(x)u(x;\ \theta) + 10^3 * (\max(0,\varphi - \ u)^2) + \ 10^6 * [u(0;\ \theta)^2 + u(1;\ \theta)^2].$$

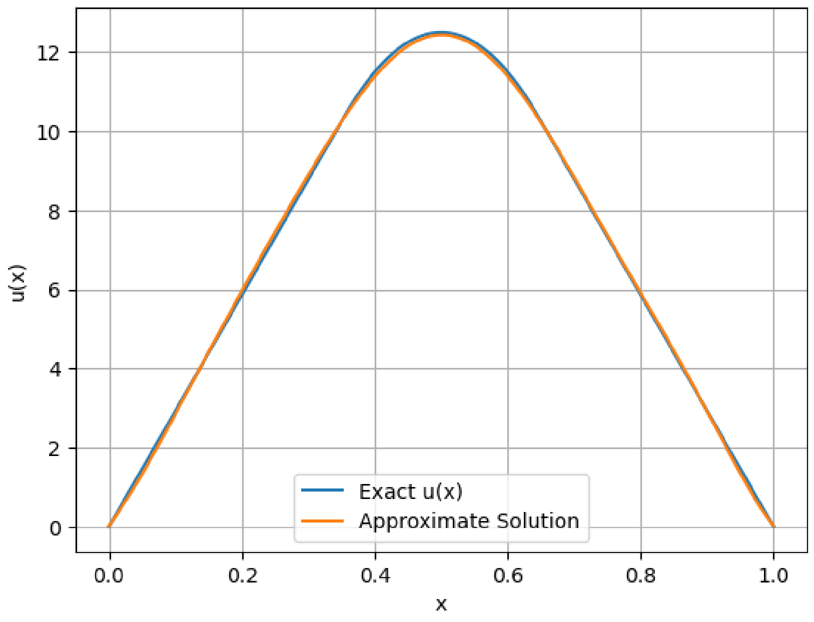


Figure 6: Solution for problem in 4.3. Exact solution (Blue), Approximate solution (Orange)

4.4. Solving a 1D Obstacle Problem using PINN

The previous 1D problem from [7] is solved again using PINN.

The loss function is:

$$Loss = \frac{1}{N_r}\sum_{i=1}^{N_r} |H(u(x_r^i;\theta) - g(x_r^i)) \cdot R(x_r^i;\theta) + ReLU(g(x_r^i) - u(x_r^i;\theta))|^2 + \alpha_b |u(0;\theta)|^2 + \alpha_b |u(1;\theta)|^2$$

where H is the Heavyside step Function

$$H(x) = \begin{cases} 1 & if\ x \geq 0, \\ 0 & otherwise \end{cases}$$

and $R(x_r^i;\theta) = u''$

For my problem, I set $\alpha_b = 10^2$ and $N_r = 500$, i.e. I have generated 500 points within the domain to train the model.

Network Architecture: The neural network used in this problem consists of fully connected layers and Tanh activation functions. The architecture allows for customization of input size, output size, and the number of hidden layers. For this specific problem, the input and output sizes are set to 1, and there are 5 hidden layers, each with a size of 60. The network applies linear transformations to the input, followed by Tanh activation for each layer. The final layer maps the hidden representation to the output dimension. This architecture has proven effective in approximating the desired solutions for obstacle problems using the PINN method.

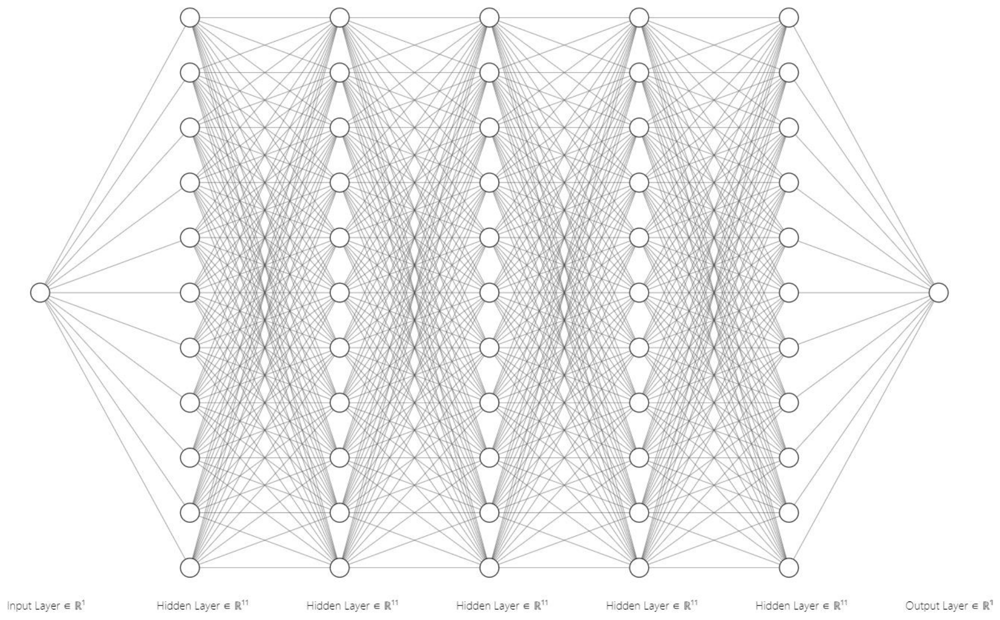


Figure 7: Neural Network architecture used for problem in 4.4.

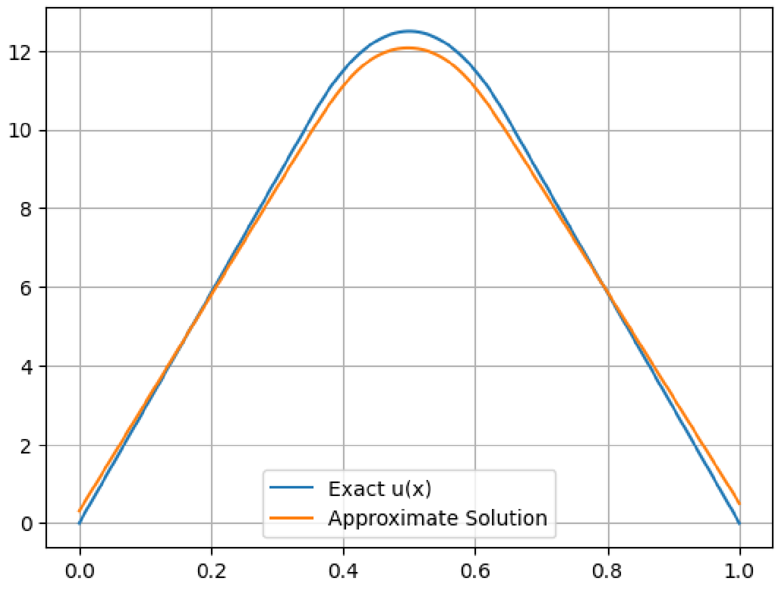


Figure 8: Solution for problem in 4.4. Exact solution (Blue), Approximate solution (Orange)

4.5. Solving 2D Obstacle Problem using PINN

$$-\Delta u \geq f \text{ in } \Omega$$

$$u \geq g \text{ in } \Omega$$

$$(-\Delta u - f)(u - g) = 0 \text{ in } \Omega$$

$$u = h \text{ on } \partial\Omega$$

where $\Omega = [-2,2]^2$, $f = 0$ and the obstacle function

$$g(x,y) = \begin{cases} \sqrt{1-r^2} & r = \sqrt{x^2+y^2} \leq 1, \\ -1 & elsewhere \end{cases}$$

The loss function is

$$Loss = \frac{1}{N_r}\sum_{i=1}^{N_r} |H(u(x_r^i;\theta) - g(x_r^i)) \cdot R(x_r^i;\theta) + ReLU(g(x_r^i) - u(x_r^i;\theta))|^2$$
$$+ \frac{\alpha_b}{N_b}\sum_{i=1}^{N_b} |u(x_b^i;\theta) - h(x_b^i)|^2$$

where H is the Heavyside step Function

$$H(x) = \begin{cases} 1 & if\ x \geq 0, \\ 0 & otherwise \end{cases}$$

and $R(x_r^i;\theta) = -\Delta u - f$

For this simulation the following values were chosen: N_r= 2500, $N_b = 200$ and $\alpha_b = 10^2$

Architecture of the Neural Network Used

The architecture used is a PINN model with multiple fully connected layers and Tanh activation functions, which allows for the approximation of the solution to a PDE based on the input features (x). The input to the network is a tensor representing the input features (x). The input size is 2, indicating two-dimensional input data. The output of the network is a tensor representing the predicted solution (u) of the PDE, with size (batch_size, output_size). The output size is typically 1, indicating a scalar prediction. The number of layers and the size of the hidden layers are configurable parameters. The default values are set to 5 layers and 60 hidden units, respectively.

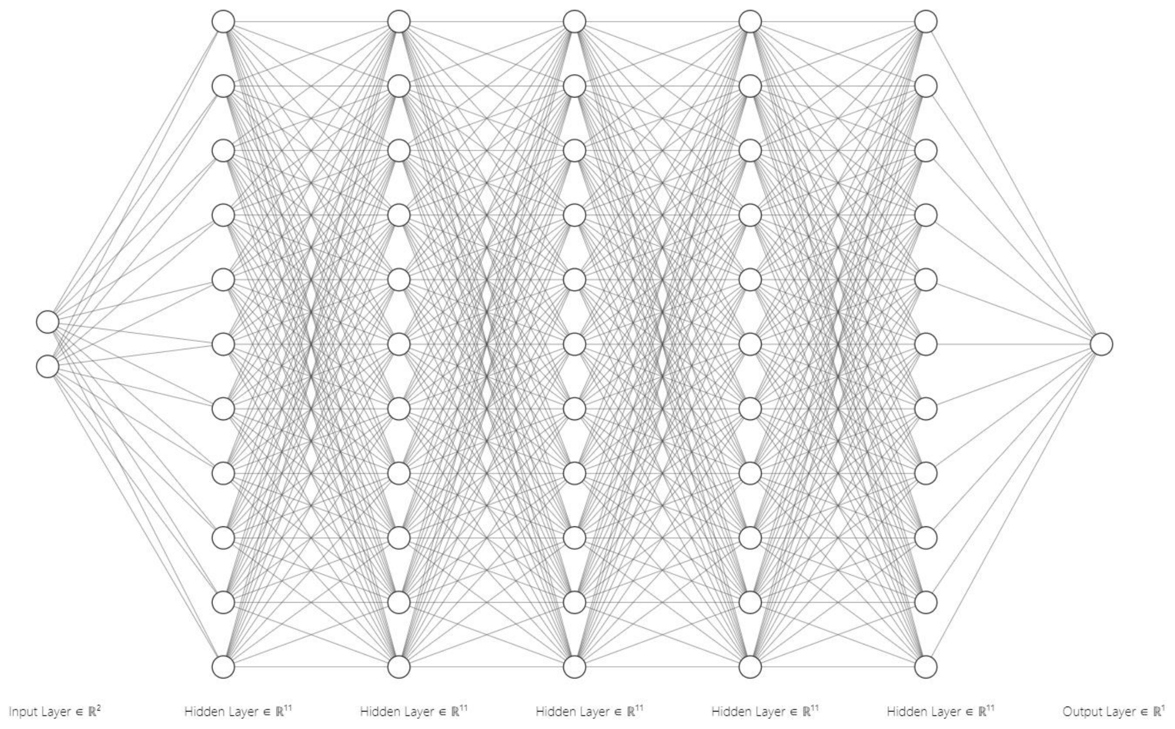


Figure 9: Neural Network architecture used for problem in 4.5.

The solution achieved from the PINN model is shown below:

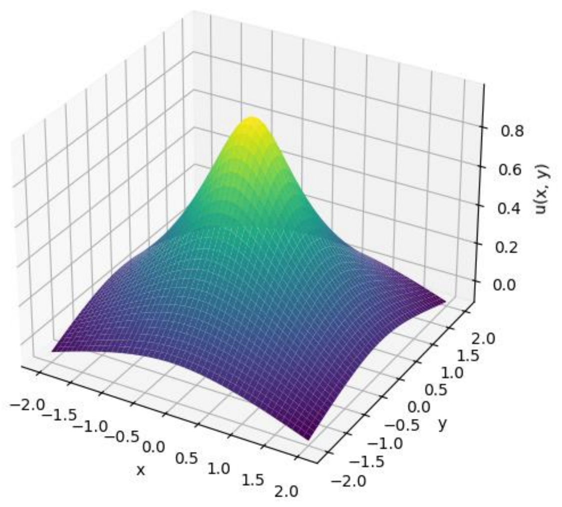


Figure 10: Solution for problem in 4.4 using PINN.

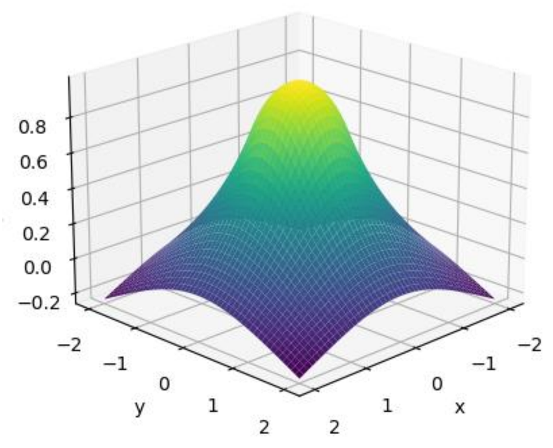


Figure 11: Exact solution for problem in 4.4.

4.6. Solving 2D Obstacle Problem using Deep-Ritz

The 2D problem in 4.5 has been further solved using the Deep-Ritz method.

The loss function is

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}[\frac{1}{2}\|\nabla u_\theta(X_i)\|_2^2 - f(X_i)u(X_i)]$$
$$+\frac{\alpha}{N}\sum_{i=1}^{N}[g(X_i) - u_\theta(X_i)]_+^2 + \frac{\beta}{M}\sum_{j=1}^{M}[u_\theta(Y_i) - h(Y_i)]^2.$$

In this simulation, the values chosen are as follows: $N = 2500$, $M = 200$, $\alpha = 10^2$ and $\beta = 10^4$.

Architecture of the Neural Network Used

The architecture used is a Deep-Ritz model with multiple fully connected layers and ReLU activation functions cubed. The input to the network is a tensor representing the input features (x). The input size is 2, indicating two-dimensional input data. The output of the network is a tensor representing the predicted solution (u) of the PDE, with size (batch_size, output_size). The output size is typically 1, indicating a scalar prediction. The number of layers and the size of the hidden layers are configurable parameters. The default values are set to 5 layers and 60 hidden units, respectively.

The solution achieved from the Deep-Ritz model is shown below:

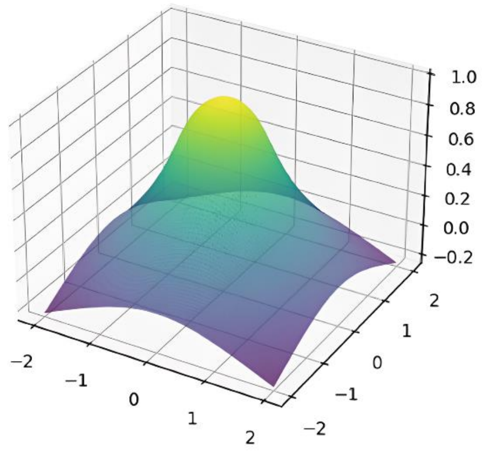


Figure 12: Solution for problem in 4.4 using Deep-Ritz method.

5. Observations and Conclusion:

The project demonstrates the effectiveness of the Deep-Ritz method and PINN for solving obstacle problems. The developed neural networks, along with the defined loss functions, have provided accurate approximations to the desired solutions. The models demonstrated promising performance in solving obstacle problems using the Deep-Ritz method and PINN. The approximate solutions obtained from the trained models exhibited high accuracy compared to the exact solutions. The neural networks and loss functions employed effectively captured the underlying dynamics and constraints of the problems.

Throughout the project, the architecture was carefully designed and implemented, considering configurable parameters such as the number of layers and hidden units. The model's ability to handle PDEs was showcased by passing input tensors through the network, resulting in predicted solution tensors.

By leveraging the power of deep learning techniques, the PINN model and Deep-Ritz method offered promising alternatives to traditional numerical methods for solving PDEs. Its ability to incorporate physics-based constraints while harnessing the expressive power of neural networks made it a valuable tool in tackling complex PDE problems.

The successful implementation of the PINN architecture opens possibilities for further research and exploration in the field of PDE solving using deep learning. Future work could involve investigating the model's performance on a wider range of PDEs, optimizing hyperparameters for improved accuracy, and exploring additional architectural variations to enhance its capabilities.

6. References:

[1] E. Burman, P. Hansbo, M.G. Larson, R. Stenberg, Galerkin least squares finite element method for the obstacle problem, Comput. Methods Appl. Mech. Engrg. 313 (2017) 362–374.
[2] R. Kornhuber, Monotone multigrid methods for elliptic variational inequalities II, Numer. Math. 72 (1996) 481–499.
[3] D.M. Yuan, X.L. Cheng, An iterative algorithm based on the piecewise linear system for solving bilateral obstacle problems, Int. J. Comput. Math. 89 (16–18) (2012) 2374–2384.
[4] T. Fuhrer, First-order least-squares method for the obstacle problem, Numer. Math. (4) (2020) 55–88.
[5] K. Majava, X.-C. Tai, A level set method for solving free boundary problems associated with obstacles, Int. J. Numer. Anal. Model. 1 (2) (2004) 157–171.
[6] Q. Ran, X. Cheng, S. Abide, A dynamical method for solving the obstacle problem, Numer. Math. Theory Methods Appl. 2 (13) (2020) 353–371.
[7] X. Cheng, X. Shen, X. Wang, K. Liang, A deep neural network-based method for solving obstacle problems, Nonlinear Analysis: Real World Applications, 72, 2023.
[8] Kinderlehrer, D., & Stampacchia, G. (1980). An Introduction to Variational Inequalities and Their Applications. Academic Press.
[9] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
[10] Bahja, Hamid El, et al. "A physics-informed neural network framework for modeling obstacle-related equations." *arXiv preprint arXiv:2304.03552* (2023).