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

Roshan Kumar & Kshitij Singhal

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI, INDIA.

May 1, 2023

Table of Contents

Recap: Phase-1

- Solved Black-Scholes equation with PINN. We used Deep Galerkin Method (DGM).
- A loss function is created consisting of three parts:
  1. A measure of how well the approximation satisfies the differential operator.
  2. A measure of how well the approximation satisfies the boundary condition.
  3. A measure of how well the approximation satisfies the initial condition.
- We Minimize the loss function using stochastic gradient descent.

Recap: PINN (Physics Informed Neural Network)

- A Deep neural network (DNN) 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 $u_{NN}(x) \in \mathbb{R}$
- 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

We consider the following problem:

$$\begin{aligned} &\mathcal{L}^q u(x,t) = f(x,t), && (x,t) \in \Omega \times (0,T] \\ &u(x,t) = h(x,t), && (x,t) \in \partial\Omega \times (0,T] \\ &u(x,0) = g(x), && x \in \Omega \end{aligned}$$

We define the strong form residual, the boundary residual, and initial residual as:

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

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

$$L^s = L_r^s + L_b + L_0,$$

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

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

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

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

Introduction to VPINN

Why a new method? Is PINN not good enough?

Introduction to VPINN

Why a new method? Is PINN not good enough?

1. In PINN, we require a large number of penalizing points. We will see in VPINN this requirement is replaced by small number of quadrature points.
2. Reduces the computational cost.
3. More accurate result

Introduction to VPINN

1. In VPINN, we incorporate the variational (weak) formulation of the problem and consider the variational loss function.
2. We choose v_j as a test funnction. Multiply our underlying equation by this test function and integrate it over the whole domain to get the variational form.

We define the variational loss function to be:

$$L^v = L_r^v + L_u + L_0,$$

$$L_r^v = \frac{1}{K}\sum_{j=1}^{K} |\mathcal{R}_j|^2,$$

$$where \quad \mathcal{R}_j(\tilde{u}) = \int_{\Omega\times(0,T]} r(\tilde{u}) v_j dxdt = 0$$

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

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

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.

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

In the strong form, we evaluate the residual at some penalizing points, while there are no such points in the variational form, and instead, we test the residual with a set of test functions.

Challenges

- A major challenge in VPINN, unlike PINN, is to compute the integrals in the loss function.
- The compositional structure of neural networks makes it almost impossible to obtain analytic expression of these integrals. Their computation is also not fully practically feasible as even in the case of network with two hidden layer.
- We employ a numerical integration technique such as quadrature rules

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 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

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

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 defined in VPINN.

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 $u(x) : \Omega \rightarrow \mathbb{R}$ and $\Omega = [-1, 1]$, that satisfies Poisson's equation:

$$\begin{aligned} -\frac{d^2u(x)}{dx^2} &= f(x), \\ u(-1) &= g \\ u(1) &= h. \end{aligned}$$

One-dimensional Poisson's equation

- We approximate solution be $u(x) \approx \tilde{u}(x) = u_{NN}$, then the strong-form residual becomes

$$r(x) = -\frac{d^2 u_{NN}(x)}{dx^2} - f(x), \quad x \in (-1,1),$$
$$r_b(x) = u_{NN}(x) - u(x), \quad x = \pm 1.$$

One-dimensional Poisson's equation

- 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 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.

Variational Residual

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

Loss Function

- hp-variational loss function for each case takes the form

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

Example

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

Code Implementation

- 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.

Results

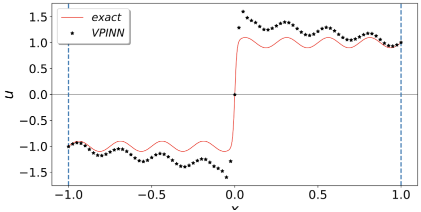


Figure: VPINN approximation to One-dimensional Poisson's equation

Results

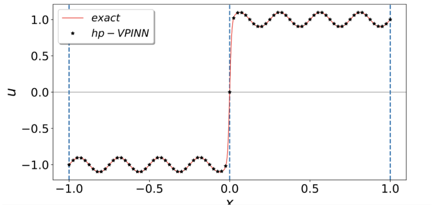


Figure: hp-VPINN approximation to One-dimensional Poisson's equation with 3 elements

Two-dimensional Poisson's equation

- The problem can be defined as finding the function $u(x, y)$, where $u(x, y) : \Omega \to \mathbb{R}$ and $\Omega = [-1, 1] \times [-1, 1]$, that satisfies the two-dimensional Poisson's equation:

$$\nabla^2 u(x, y) = f(x, y),$$

Two-dimensional Poisson's equation

- We approximate solution be $u(x) \approx \tilde{u}(x) = u_{NN}$, then the strong-form residual becomes

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

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

Two-dimensional Poisson's equation

- 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 $e_y = 1, 2, ..., N_{el_y}$.

Compact support

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

Variational Residual

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

Loss Function

- hp-variational loss function for each case takes the form

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

Example

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

Code Implementation

- 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.

Results

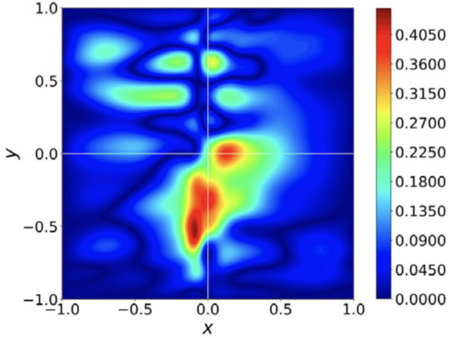


Figure: VPINN point-wise error for Poisson-2D

Results

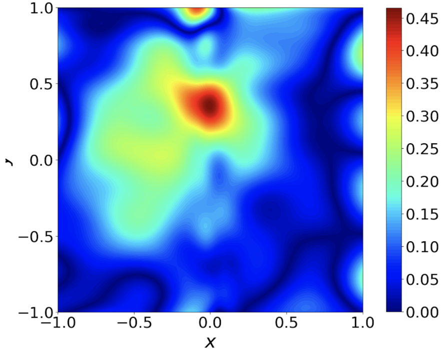


Figure: hp-VPINN point-wise error for Poisson-2D

Is hp-VPINN more accurate than VPINN?

- 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.

One-Dimensional Black-Scholes PDE

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

where $H(x) = max(0, x - K)$ for European Call Option

One-Dimensional Black-Scholes PDE

- 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.

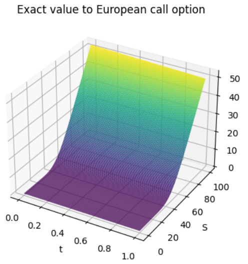
One-Dimensional Black-Scholes PDE

- 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 $e_x = 1, 2, ..., N_{el_x}$.

Code implementation

- To employ the hp-VPINN formulation, we construct a fully connected neural network with a tanh activation function and specify the following parameters: $l = 3$, $N = 5$ and $K_1 = K_2 = 5$. In addition, we use Legendre test functions in both space and time directions.
- Value of $\sigma = 0.25, K = 50, r = 0.05$.

Results



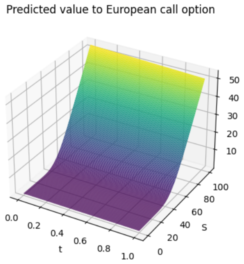


Figure: 1-D Black Scholes equation

Conclusion

- We presented a novel approach to solving the Black-Scholes equation using Variational Physics-Informed Neural Network (VPINN) and hp-VPINN.
- 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.
- This demonstrates the potential of VPINN and hp-VPINN in solving the Black-Scholes equation and other complex partial differential equations in finance and beyond.

Bibliography

1. Ali Al-Aradi. *Solving Nonlinear and High-Dimensional Partial Differential Equations via Deep Learning*. 2020
2. Ehsan Kharazmia, Zhongqiang Zhangb, and George E.M.Karniadakis. VPINNs: *Variational physics-informed neural networks for solving partial differential equations.* arXiv:1912.00873, 2021.
3. Ehsan Kharazmia, Zhongqiang Zhangb, and George E.M.Karniadakis. hp-VPINNs: *Variational physics-informed neural networks with domain decomposition*. 2021

Thank You