A High-Efficient Hybrid Physics-Informed Neural Networks Based on Convolutional Neural Network

Zhiwei Fang

IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS 2021

PINN

$$\begin{aligned} \mathcal{N}[u](\mathbf{x}) &= 0 \quad \text{in } \Omega, \\ \mathcal{B}[u](\mathbf{x}) &= 0 \quad \text{on } \partial\Omega \end{aligned}$$

$$\text{MSE} = \text{MSE}_u + \text{MSE}_b.$$

$$\text{MSE}_u = \frac{1}{N_u} \sum_{i=1}^{N_u} \left| \mathcal{N}[u]\left(\mathbf{x}_u^i\right) \right|^2.$$

$$\text{MSE}_b = \frac{1}{N_b} \sum_{i=1}^{N_b} \left| \mathcal{B}[u]\left(\mathbf{x}_b^i\right) \right|^2$$

Example 1:

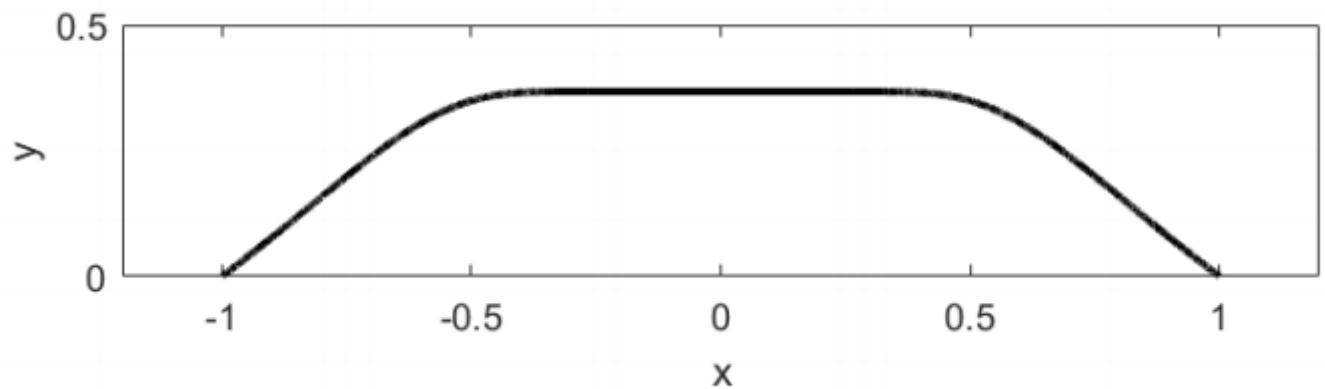
$$u''(x) = 0 \quad \text{in } (-1, 1),$$
$$u(-1) = u(1) = 0.$$

Exact solution: $u(x) = 0$

If we choose the point set {0,+1,-1} to train PINN, the loss function will be

$$\text{MSE} = |u''(0)|^2 + \frac{1}{2}\left(|u(-1)|^2 + |u(1)|^2\right).$$

The prediction of PINN may look like this



Example 2:

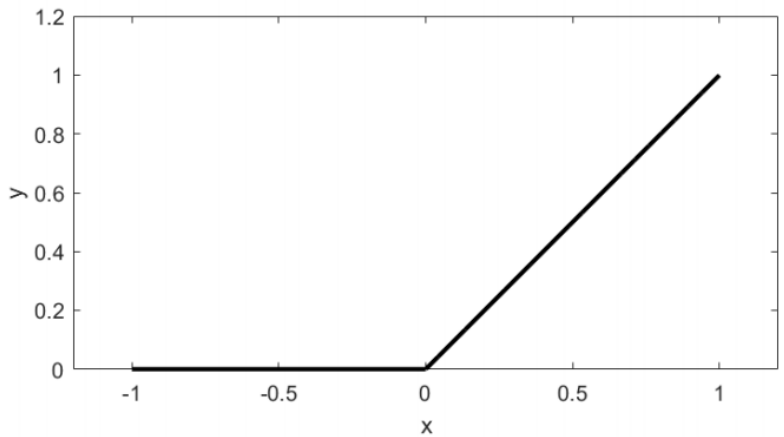
$$u''(x) = 0 \quad \text{in } (-1, 1),$$
$$u(-1) = 0, \quad u(1) = 1.$$

Exact solution: $u(x) = (x + 1/2)$

Suppose we choose the point set [-1,1] except 0 to train PINN. The loss function is:

$$\text{MSE} = \frac{1}{N_u} \sum_{i=1}^{N_u} \left| u''(x_u^i) \right|^2 + \frac{1}{2} \left(|u(-1)|^2 + |u(1) - 1|^2 \right)$$

The prediction of PINN may look like this



Inspiration

A feature of PINN and its variations is that they all rely on the AD technique(automatic differentiation)

The author found a heat equation online about solving PDEs and it is a finite-difference scheme. In that example, the Laplacian operator has been approximated by the convolution with kernel

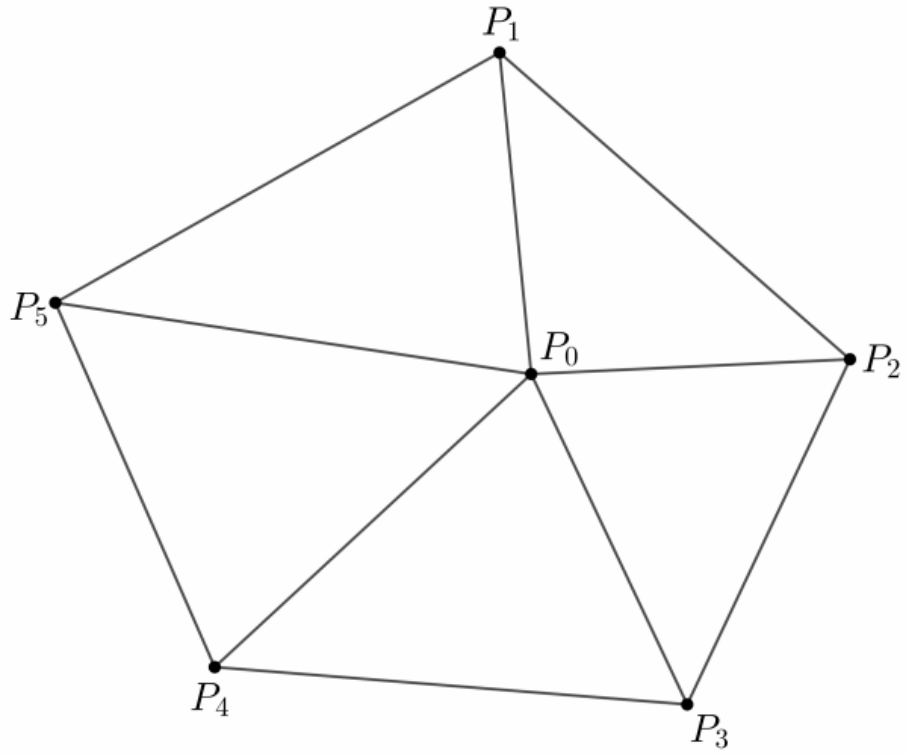
$$\begin{bmatrix} 0.5 & 1 & 0.5 \\ 1 & -6 & 1 \\ 0.5 & 1 & 0.5 \end{bmatrix}.$$

New method: add in traditional numerical method and CNN structure and approximate the differential operator by the numerical method instead of AD

Method:

In a finite volume method or generalized difference method, the Laplacian operator at P0 can be discretized by

$$\Delta u(P_0) = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \Big|_{P_0} \approx \sum_{i=0}^{5} w_i u(P_i).$$



$$\{1,\ x,\ y,\ x^2,\ xy,\ y^2\}$$

$$\Delta u(P_0) = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)\Big|_{P_0} \approx \sum_{i=0}^{5} w_i u(P_i).$$

$$\sum_{i=0}^{5} w_i = 0, \quad \sum_{i=0}^{5} w_i x_i = 0, \quad \sum_{i=0}^{5} w_i y_i = 0,$$

$$\sum_{i=0}^{5} w_i x_i^2 = 2, \quad \sum_{i=0}^{5} w_i x_i y_i = 0, \quad \sum_{i=0}^{5} w_i y_i^2 = 2$$

If the coefficient matrix of above is full-ranked, then this system is uniquely solvable and we get the approximation of at P0.

$\mathcal{P}_d^k$ the set of d-dimensional homogeneous polynomial with a degree no more than k

$\widetilde{\mathcal{P}}_d^k$ the set of d-dimensional homogeneous polynomial with degree k

$$\rho(d,k) = \binom{d+k}{k}$$

$$\widetilde{\rho}(d,k) = \binom{d+k-1}{k}$$

} the number of elements in above sets

P0 = (0, 0) (center), P1 = (0, 1) (up), P2 = (−1, 0) (left), P3 = (0, −1) (down), and P4 = (1, 0) (right), and then fit the coefficients by using x^2 and x^4

$$\Delta(x^2)\big|_{(0,0)} \approx w_2 + w_4 = 2 \qquad \Delta(x^4)\big|_{(0,0)} \approx w_2 + w_4 = 0$$

Reason: there is no k ∈ N such that ρ(2, k) = 5

A general case:

$$\Delta u(P_0) \approx \sum_{i=0}^{m} w_i u(P_i). \qquad P_0 \in \mathbb{R}^d$$

use the least-square method:

$$\min_{\substack{w_i \in \mathbb{R} \\ i=1,\ldots,m}} W_1 \sum_{f \in \mathcal{P}_d^{k_{\min}}} \left(\sum_{i=0}^{m} w_i f(P_i) - \Delta f(P_0) \right)^2 + W_2 \sum_{f \in \left(\mathcal{P}_d^{k_{\max}} \setminus \mathcal{P}_d^{k_{\min}} \right)} \left(\sum_{i=0}^{m} w_i f(P_i) - \Delta f(P_0) \right)^2$$

$$\begin{cases} \rho(d, k_{\min}) \leq n_{\min} \\ \rho(d, k_{\max}) \geq n_{\max}. \end{cases}$$

Theorem 1:

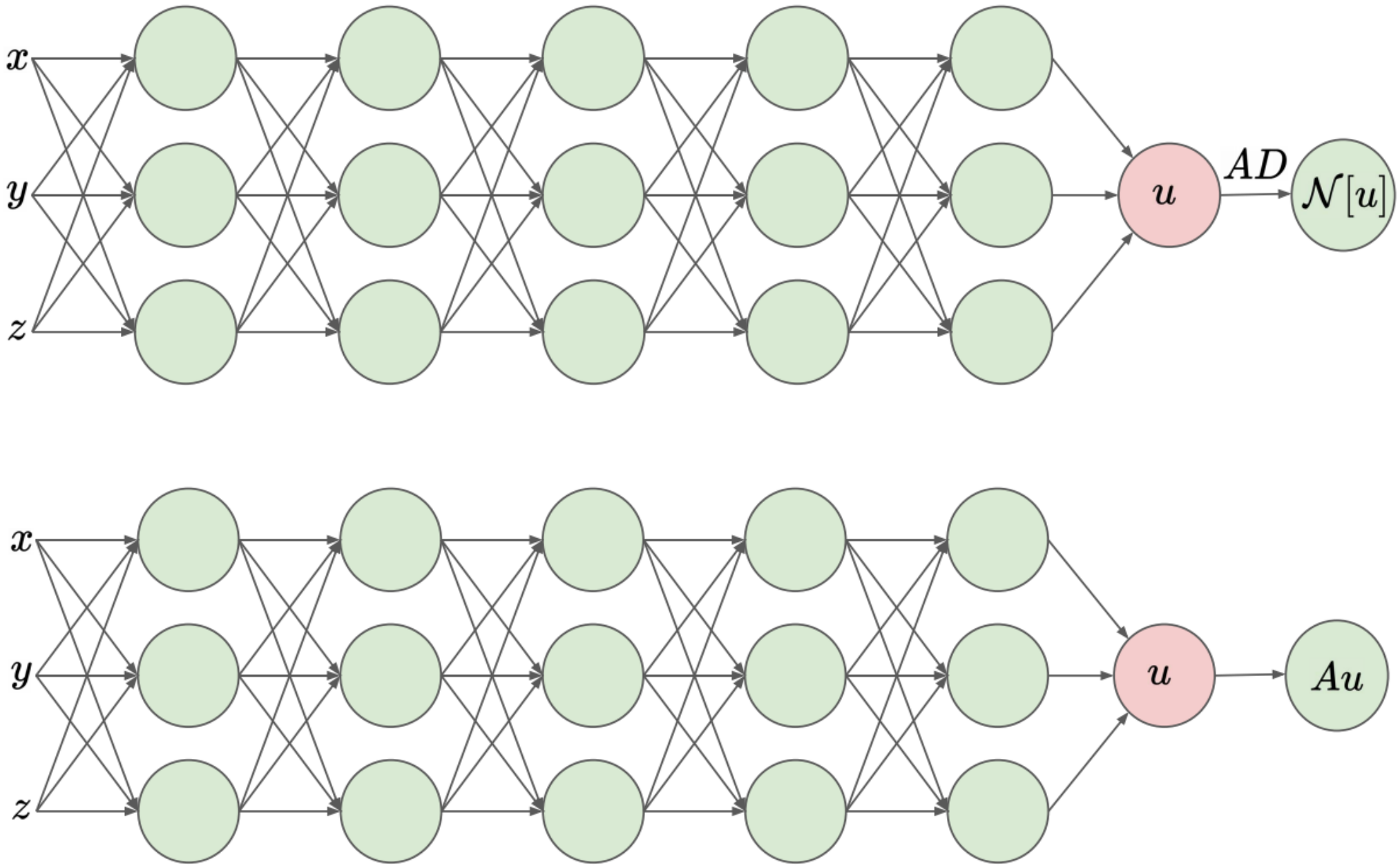
This method is correct for any nth order linear differential operator under the corresponding conditions (not just Laplacian operator)

$$\mathcal{L}[u](P_0) \approx \sum_{i=0}^{m} w_i u(P_i)$$

$$\mathcal{L} = \sum_{j=1}^{l} c_j \partial^{\boldsymbol{\alpha}_j} \quad \partial_i^{\alpha_i} = \frac{\partial^{\alpha_i}}{\partial x_i^{\alpha_i}}.$$

$$\left| \mathcal{L}[u](P_0) - \sum_{i=0}^{m} w_i u(P_i) \right| \leq C h^{n+1} \left(\sum_{i=1}^{m} |w_i| \right)$$

$$\mathcal{L}[f](P_0) = \sum_{i=0}^{m} w_i f(P_i) \quad \forall f \in \mathcal{P}_d^n$$


x
y
z
u
AD
$\mathcal{N}[u]$
x
y
z
u
Au

$$\mathcal{L}[u](\mathbf{x}) = f(\mathbf{x}) \text{ in } \Omega,$$
$$\mathcal{B}[u](\mathbf{x}) = 0 \text{ on } \partial\Omega.$$

Algorithm 1 Framework of Hybrid PINN

Input:

The set of mesh points $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=1}^{N_u}$;

The set of boundary points $\mathcal{D}_b = \{\mathbf{x}_i\}_{i=1}^{N_b}$;

The set of polynomials $\mathcal{D}_p = \{p_i(\mathbf{x})\}_{i=1}^{k}$ who satisfies theorem 1;

$$\mathcal{L}[f](P_0) = \sum_{i=0}^{m} w_i f(P_i) \quad \forall f \in \mathcal{P}_d^n$$

Establish a neural network u_h to approximate the solution. The input of this neural network is $\mathbf{x}$, and the output is the solution value at $\mathbf{x}$.

Output:

1: Get the approximation matrix $\mathbf{A}$ by solving (16) with the polynomials set $\mathcal{D}_p$.

2: Train the neural network u_h by the loss function:

$$MSE = \frac{1}{N_u} \sum_{\mathbf{x}_i \in \mathcal{D}_u} |\mathbf{A}\mathbf{u}_h - f(\mathbf{x}_i)|^2 + \frac{1}{N_b} \sum_{\mathbf{x}_i \in \mathcal{D}_b} |\mathcal{B}[u_h](\mathbf{x}_i)|^2,$$

where $\mathbf{u}_h$ is a column vector of u_h at $\mathbf{x}_i$, $i = 1, \ldots N_u$.

3: **return** u_h, who can predict the solution of (23)-(24) with any point in Ω.

Experiments:

$$\|u(\mathbf{x})\|_{l^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}|u(\mathbf{x}_i)|^2}$$

$$\|u(\mathbf{x}) - u_h(\mathbf{x})\|_{l^2}$$

$$-\Delta u(\mathbf{x}) = f(\mathbf{x}) \text{ in } \Omega = (0,1)^2,$$
$$u(\mathbf{x}) = u_0(\mathbf{x}) \text{ on } \partial\Omega$$

$$u_0(\mathbf{x}) = e^{x^2}\sin(y).$$

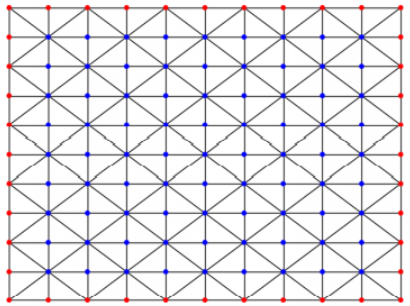


Fig. 6. Sample of regular mesh.

Mesh size	Discretized $L^2(\Omega)$ error	Time for $\mathbf{A}$ (s)	Time to train DNN (s)
0.2	$2.887440e-01$	0.1745	15.2004
0.1	$2.338030e-01$	0.5266	13.2835
0.05	$8.151677e-03$	5.5681	42.6032
0.025	$4.989049e-05$	94.7228	46.0270

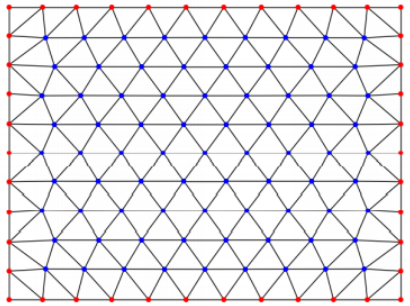


Fig. 7. Sample of irregular mesh.

Mesh size	Discretized $L^2(\Omega)$ error	Time for $\mathbf{A}$ (s)	Time to train DNN (s)
0.2	$3.885836e-01$	4.6685	14.6000
0.1	$2.338030e-01$	23.0893	13.2835
0.05	$7.950568e-04$	390.7417	45.6711
0.025	$1.066508e-04$	415.1484	38.5859

$$-(c_1(\mathbf{x})\partial_{xx}u(\mathbf{x})+c_2(\mathbf{x})\partial_{yy}u(\mathbf{x}))=f(\mathbf{x}) \text{ in } \Omega=(0,1)^2$$
$$u(\mathbf{x})=u_0(\mathbf{x}) \text{ on } \partial\Omega$$

$$c_1(\mathbf{x}) = 2+\cos(x+y),\ c_2(\mathbf{x}) = 2+\sin(x+y)$$
$$u_0(\mathbf{x}) = e^{x^2}\sin(y)$$

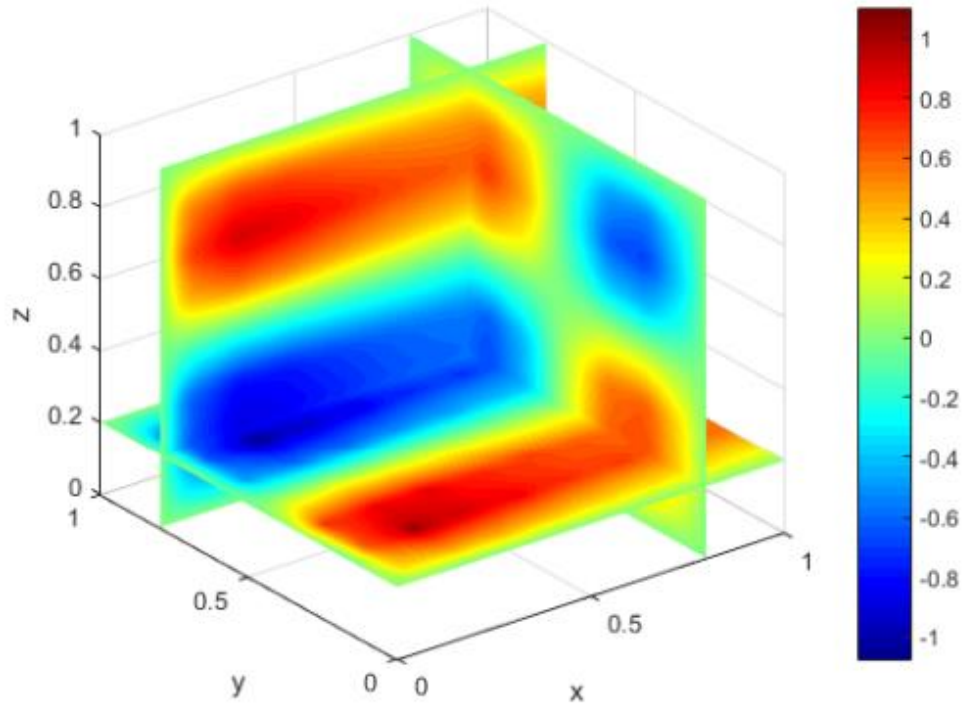


Fig. 9. Solution for test 3.

$$u_0(\mathbf{x}) = \exp\left(\left(\frac{y}{1000}\right)^2\right)\sin\left(\frac{x}{1000}\right)$$

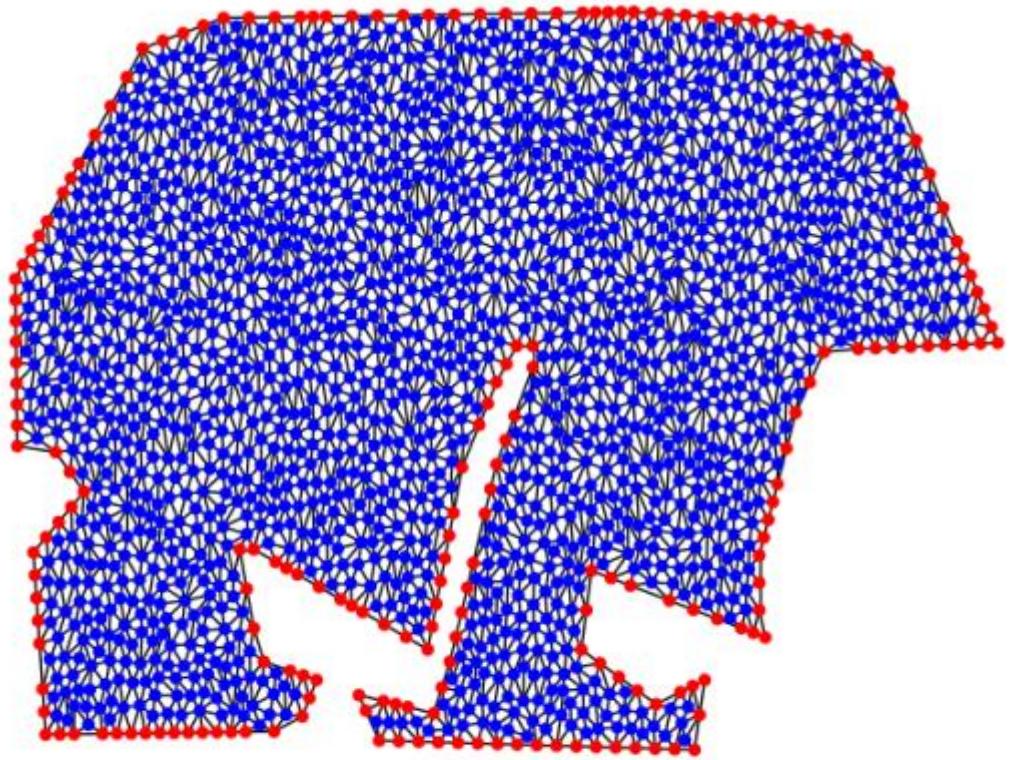


Fig. 10. Mesh for the car shape domain.

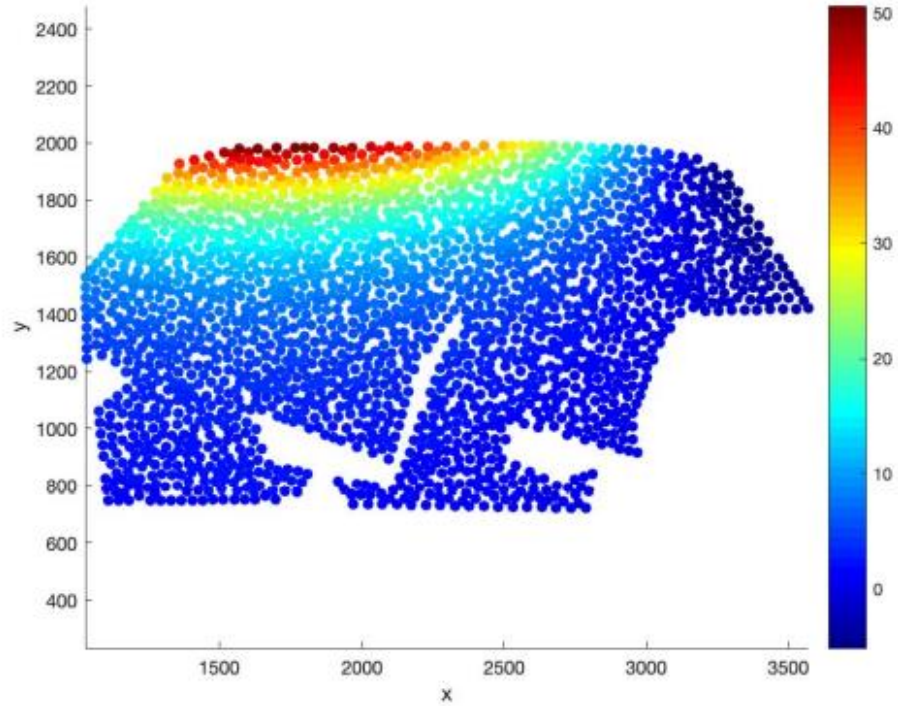


Fig. 11. Solution for test 4.

$$-\Delta_{\mathbb{S}^2} u(\tilde{\mathbf{x}}) = f(\tilde{\mathbf{x}}) \text{ on } \mathbb{S}^2 \qquad u_0(\tilde{\mathbf{x}}) = \tilde{x}\sin(\tilde{y}) + \tilde{z}$$

$$\begin{cases} \tilde{x} = (x/r),\ \tilde{y} = (y/r), \text{ and } \tilde{z} = (z/r) \\ x = r\sin(\theta)\cos(\phi), \quad y = r\sin(\theta)\sin(\phi), \quad z = r\cos(\theta) \\ r \in [0, \infty),\ \theta \in [-(\pi/2), (\pi/2)], \text{ and } \phi \in [0, 2\pi). \end{cases}$$

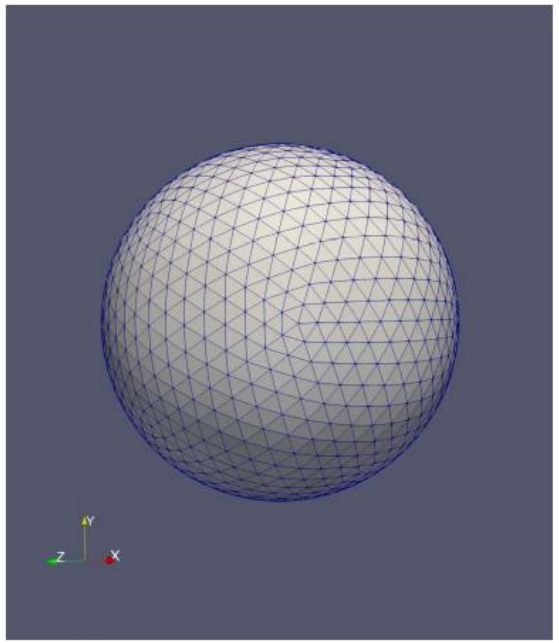


Fig. 12. Mesh for the surface PDEs.

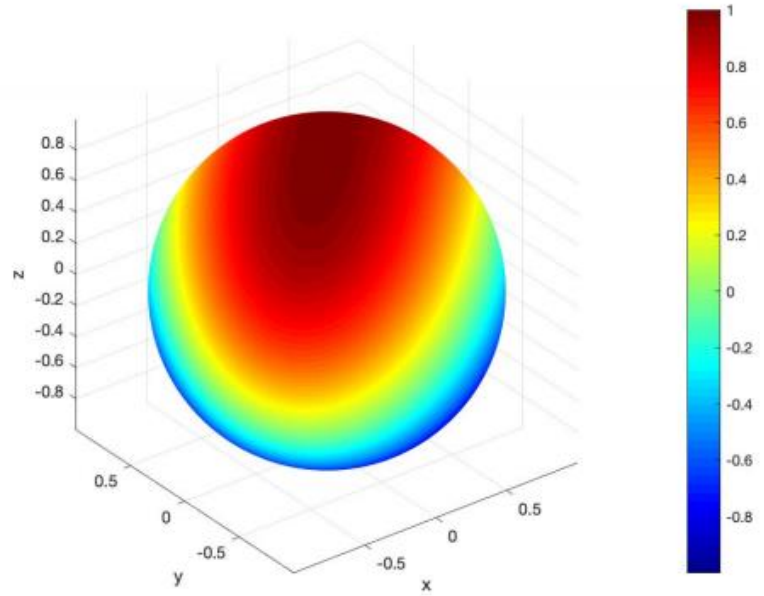


Fig. 13. Solution for test 5.

TABLE IV

RESULTS FOR (31) BY HYBRID PINN

Number of mesh points	Discretized $L^2(\Omega)$ error	Time for $\mathbf{A}$ (s)	Time to train DNN (s)
252	$3.116710e-03$	0.2748	7.4433
1002	$7.656996e-04$	1.6822	22.0047
4002	$1.918568e-04$	3.6810	49.1625
16002	$5.638292e-05$	68.4128	253.6164

TABLE V

COMPARISON TO PINN

#(points)	Discretized $L^2(\Omega)$ error	Time for PINN (s)
252	23.9306	326.8594
1002	11.7916	411.8043
4002	5.9045	443.2334
16002	2.9753	740.6775