Solving PDEs in computational mechanics using machine learning

Junbin Huang

Peking University

jbh@bicmr.pku.edu.cn

December 16, 2020

Contents

1. Traditional methods for numerical solution of PDEs
2. Machine learning approaches: Data-driven & Model-driven
   - Data-driven approaches
   - Model-driven approaches
3. A continuous formulation
4. What traditional methods can offer for NNs
   - Reduced basis methods
   - Adaptive finite element analysis using machine learning
   - Phase shift NNs & Multiscale NNs
   - Extended PINNs

Features of different methods

- Finite element methods: rigorous error estimates, fair accuracy, flexible for complex geometry, rich industrial applications. In engineering, quadrilateral or hexahedral elements are preferred, and the meshing procedure remains an issue because of the sensitivity to mesh distortions;
- Finite difference methods: easy to use on regular mesh, tricky on imposing boundary conditions, stability issue;
- Finite volume methods: suitable for conservation laws, naturally leads to conserved quantities, hard to formulate high-order methods on unstructured meshes;
- Spectral methods: highly accurate, but only for problems with sufficient regularity and regular geometry;
- Spectral elements: very high-order finite elements using spectral bases in each element;
- ...

Some new methods I

- Discontinuous Galerkin methods: combines aspects from finite elements and finite volume methods. The continuity of interpolation is weakly imposed via flux conditions. Artificial stabilization is usually required;
- Domain decomposition: converts the original problem to a set of coupled problems on different subdomains, iteratively exchanges information between shared interfaces, can simplify meshing and increase local accuracy;
- Meshless methods: uses global (rational) interpolation through scattered data points. The numerical integration and stability condition remain challenging;
- Isogeometric analysis: constructs interpolation using spline bases in geometry representation, requires a set of industrial blocks to work together;

Some new methods II

- AMORE/Overlapping paradigm: allows fast, automatic meshing by overlapping some regular sub-meshes. The accuracy and efficiency depend on the interpolation formulation and the implementation;
- ...

Neural networks can be used as a pure meshless method for general PDEs on complex geometries. The numerical integration is effectively replaced by Monte Carlo integration and mini-batch GD. Due to the universal approximation power, all difficulties are left for the optimization procedure.

Realtime fluid simulation[1]

Setting: Running time matters more than the physical exactness, e.g. in computer games or interactive design.

Features are designed from traditional SPH (smoothed particle hydrodynamics) formulations. An example:

- In SPH, the viscosity term for the *i*-th particle is

$$\mathbf{a}_i^{\text{visc}} = \frac{\mu}{\rho_0} \sum_j (\mathbf{v}_j - \mathbf{v}_i) \nabla^2 W(\mathbf{x}_j - \mathbf{x}_i).$$

- Corresponding, the viscosity feature can be given by

$$\Phi_{\mathbf{X},R}^{\text{visc}}(\mathbf{x}_i) = \frac{\mu}{\rho_0} \sum_{j \in \mathbf{X}} (\mathbf{v}_j - \mathbf{v}_i) \Omega_R(\mathbf{x}_j - \mathbf{x}_i).$$

Training data were obtained using some existing algorithm evaluated on many randomly generated scenes.

[1] L Ladický et al. "Data-driven fluid simulations using regression forests". *ACM Transactions on Graphics* 34.6 (2015).

Other examples

- In multiscale analysis, traditional approaches may be used in micro scale to estimate the average material response[2];
- PointNet for learning fluid flow near irregular objects[3];
- Prediction for history-dependent material response using RNNs[4].

Issues of data-driven approaches: generalization, and expensive offline cost.

[2] S Saha et al. "Hierarchical deep learning neural network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering". *Computer Methods in Applied Mechanics and Engineering* 373 (2021), 113452.

[3] A Kashefi, D Rempe, and LJ Guibas. *A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries*. 2020. arXiv: 2010.09469 [cs.LG].

[4] C Wang, LY Xu, and JS Fan. "A general deep learning framework for history-dependent response prediction based on UA-Seq2Seq model". *Computer Methods in Applied Mechanics and Engineering* 372 (2020), 113357.

Vanilla approach[6]

$$\mathcal{L}u = f, \quad x \in \Omega$$

$$\mathcal{B}u = g, \quad x \in \partial\Omega$$

$$u = u(x, \beta_i)$$

$$\text{Loss} = \int_{\Omega} \|\mathcal{L}u - f\|^2 \mathrm{d}V + \int_{\partial\Omega} \|\mathcal{B}u - g\|^2 \mathrm{d}S$$

This was trained by a quasi-Newton method. The idea of deep Ritz method[5] was also mentioned briefly.

[5] W E and B Yu. "The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems". *Communications in Mathematics and Statistics* 6.1 (2018), 1–12.

[6] MWMG Dissanayake and N Phan-Thien. "Neural-network-based approximations for solving partial differential equations". *Communications in Numerical Methods in Engineering* 10.3 (1994), 195–201.

Stochastic formulation[7] I

Consider a semilinear parabolic PDE

$$\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\operatorname{Tr}\left(\sigma\sigma^{\top}(t,x)(\operatorname{Hess}_x u)(t,x)\right) + \nabla u(t,x)\cdot\mu(t,x) \\ + f\left(t,x,u(t,x),\sigma^{\top}(t,x)\nabla u(t,x)\right) = 0$$

with a terminal condition $u(T,x) = g(x)$. It can be reformulated as the following BSDE

$$u(t,X_t) - u(0,X_0) = \\ -\int_0^t f(s,X_s,u(s,X_s),\sigma^{\top}(s,X_s)\nabla u(s,X_s))\mathrm{d}s \\ + \int_0^t [\nabla u(s,X_s)]^{\top}\sigma(s,X_s)\mathrm{d}W_s$$

[7] J Han, A Jentzen, and W E. "Solving high-dimensional partial differential equations using deep learning". *Proceedings of the National Academy of Sciences* 115.34 (2018), 8505–8510.

Stochastic formulation II

where

$$X_t = \xi + \int_0^t \mu(s, X_s)\mathrm{d}s + \int_0^t \sigma(s, X_s)\mathrm{d}W_s$$

and W_s is a d-dimensional Brownian motion. The value of u at the initial position ξ and its derivatives are parameters to be optimized, along with the parameters for approximating the mapping $x \mapsto \sigma^\top(t,x)\nabla u(t,x)$ at each $t = t_n$. The loss is given by Loss $= \mathbb{E}\left[|g(X_{t_N}) - u\left(\{X_{t_n}\}, \{W_{t_n}\}\right)|^2\right.$

Here the Brownian motion naturally provides randomness in the SGD.

Shallow neural network from a continuous viewpoint[8]

Consider the integral-transform based representation

$$f = \int_{\mathbb{R}^{n+2}} a\sigma(\mathbf{w}^\top \tilde{\mathbf{x}})\pi(\mathrm{d}a, \mathrm{d}\mathbf{w})$$

where π is a probability distribution and the $\mathbf{w}$ can be understood as the parameters in a 2-layer neural network. If the probability is given by a particle discretization

$$\hat{\pi}(a, \mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \delta((a, \mathbf{w}) - (a_i, \mathbf{w}_i)),$$

the gradient flow leads to the continuous time GD dynamics of a 2-layer neural network model.

[8] W E, C Ma, and L Wu. "Machine learning from a continuous viewpoint, I". *Science China Mathematics* 63.11 (2020), 2233–2266.

What we may see from the continuous formulation

In low dimensions, the particle approximation may not be efficient. Assume sufficient regularity for f. Traditional approximations may be used, e.g. spectral basis, and piecewise polynomials.

This integral transform is similar to the Ridgelet transform[9]. However, the Ridgelet transform is adapted for high-dimensional intermittency along hyperplanes. If singularity exists along a general manifold, or if the singularity has low dimensions, the approximation should be modified.

Wavelet transform? Fourier transform?

[9] EJ Candès and DL Donoho. "Ridgelets: a key to higher-dimensional intermittency?" *Philosophical Transactions of the Royal Society A* 357.1760 (1999), 2495–2509.

When we have affine dependence I

Consider the weak statement

$$a^{\mu}(u^{\mu}, v) = f(v), \quad \forall v \in H$$

where $\mu \in \mathcal{P}$ is the parameter vector, and $u^{\mu} \in H$ is the numerical solution we are looking for. For each $\mu_i \in \mathcal{P}$, $i \in [N]$, we can use finite element methods to solve for a high fidelity solution u^{μ_i}, which may be called a snapshot. If

$$a^{\mu}(\cdot, \cdot) = \sum_{q} \Theta_q(\mu) a^{(q)}(\cdot, \cdot),$$

we say the bilinear form is affine in parameters.
After generating enough snapshots, greedy algorithms or POD can be used to select a reduced (orthogonal) basis $\{v_1, v_2, \dots, v_M\}$, $M \ll N$.

When we have affine dependence II

For any new parameter vector μ, we solve the original weak problem

$$a^{\mu}(u^{\mu}, v) = f(v), \quad \forall v \in H^*$$

in a new space $H^* = \text{span}\{v_1, v_2, \ldots, v_M\}$ with much reduced dimensions. This leads to a linear system with much fewer DOFs. The calculation of most matrices can be performed offline and stored.

Learning the projection

When we have affine dependence, the solution in reduced space is determined by a Galerkin projection. However, the Galerkin projection no longer saves computational effort for general nonlinear problems.[10].

Procedures were proposed to approximate differential operators using affine expansion in general non-affine cases.

Machine learning here comes to help. We simply need to learn a projection from high-fidelity solutions to the reduced space H^*.

[10] M Guo and JS Hesthaven. "Reduced order modeling for nonlinear structural analysis using Gaussian process regression". *Computer Methods in Applied Mechanics and Engineering* 341 (2018), 807–826.

GPR numerical results I

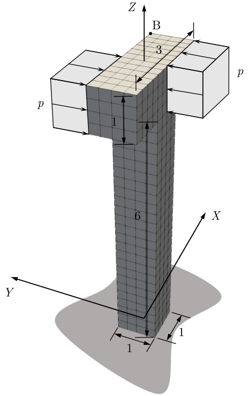


Figure: A twisting column. Reprinted from [10].

GPR numerical results II

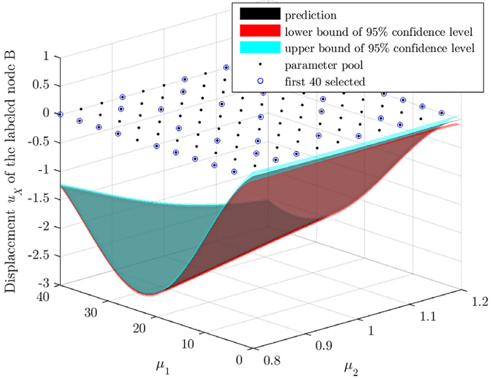


Figure: Regression results for the surface of displacement u_X of the labeled node B. Reprinted from [10]. (RB space of rank 6 generated from 25 snapshots; Regressor trained by 40 data.)

Adaptive finite element analysis

- p-adaptivity: increasing the order of finite elements locally;
- h-adaptivity: refining the mesh locally;
- r-adaptivity: optimizing the positions of finite element nodes.

Representing FEs using NNs[11] I

Motivation: ReLU can be used to represent the piecewise linear basis of linear elements.

1D example:

$$N_I(x) = \begin{cases} \frac{x - x_{I-1}}{x_I - x_{I-1}}, & x_{I-1} \leq x \leq x_I, \\ \frac{x_{I+1} - x}{x_{I+1} - x_I}, & x_I \leq x \leq x_{I+1}, \\ 0, & \text{elsewhere}, \end{cases}$$

$$u^h = \sum_I N_I(x) u_I.$$

[11] L Zhang et al. "Hierarchical deep-learning neural networks: finite elements and beyond". *Computational Mechanics* (2020).

Representing FEs using NNs II

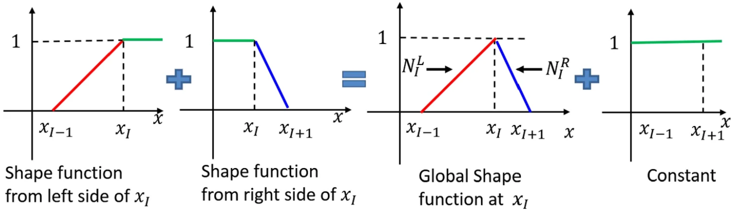
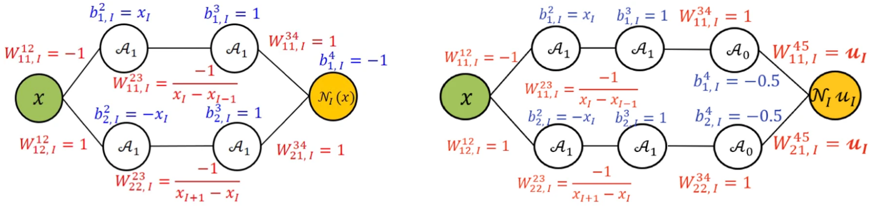

1
x_{I-1}
x_I
x
Shape function from left side of x_I
1
x_I
x_{I+1}
x
Shape function from right side of x_I
1
N_I^L
N_I^R
x_{I-1}
x_I
x_{I+1}
x
Global Shape function at x_I
1
x_{I-1}
x_{I+1}
x
Constant

Figure: 1D linear shape function at x_I. Reprinted from [11].

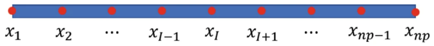
Representing FEs using NNs III



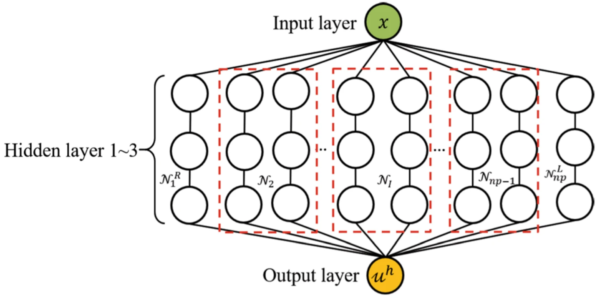
(a) DNN-based 1D shape function **(b)** DNN-based 1D interpolation function

Figure: Neural network representation of the 1D linear shape function and interpolation function. Reprinted from [11].

Representing FEs using NNs IV



(a) 1D mesh of the initial nodal configuration.



(b) Assembly of the DNNs for the 1D mesh of *np* nodes

Figure: Neural network representation of the global interpolation. Reprinted from [11].

Adaptive finite element analysis

- By adding quadratic activation function and inverse activation function, any rational interpolation can be represented using structured DNNs; $(f_1 f_2 = \frac{1}{2}\left((f_1 + f_2)^2 - f_1^2 - f_2\right)...)$
- Solving PDEs using FEM is equivalent to the training of the weights at the last hidden layer;
- Optimizing the loss function (energy) leads to optimization of the nodal positions (r-adaptivity).

Numerical results I

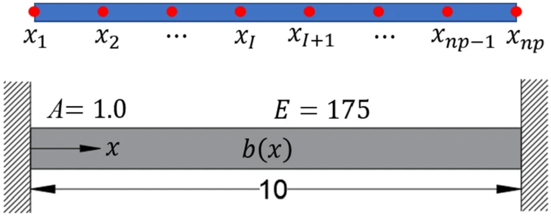


Figure: A 1D example. Reprinted from [11].

Numerical results II

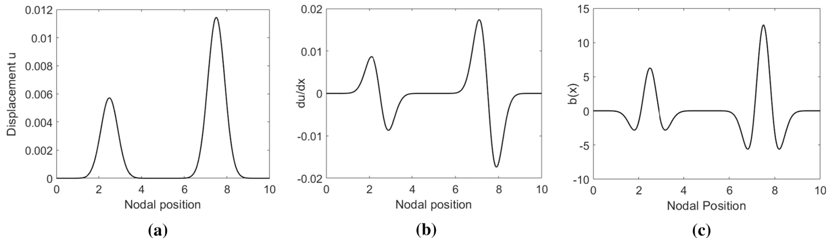


Figure: The exact solution. Reprinted from [11].

Numerical results III

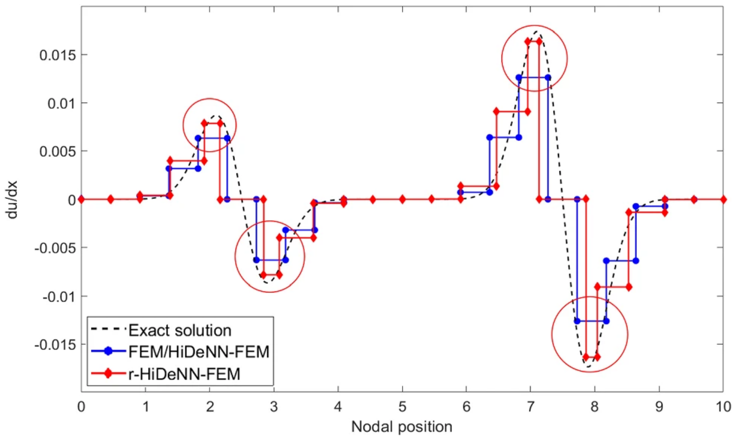


Figure: The numerical solution. Reprinted from [11].

Numerical results IV

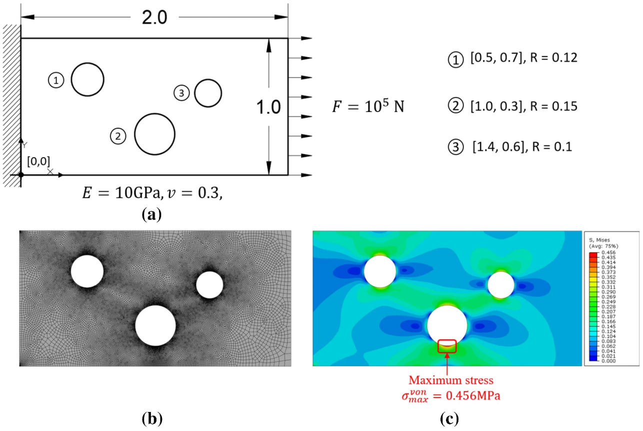


Figure: A 2D example. Reprinted from [11].

Numerical results V

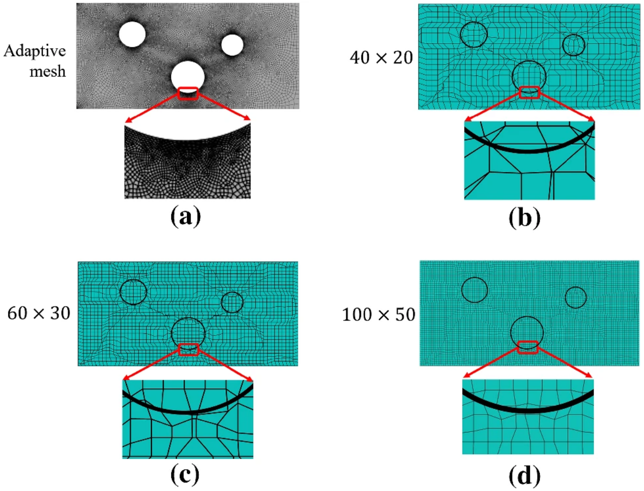


Figure: The adaptive meshes. Reprinted from [11].

Frequency principle[12]

Theorem

Consider a DNN of one hidden layer with a tanh activation function. For any frequencies k_1 and k_2 such that $|\hat{f}(k_1)| > 0$, $|\hat{f}(k_2)| > 0$, and $|k_2| > |k_1| > 0$, there exist positive constants c and C such that for sufficiently small δ, we have

$$\frac{\mu\left(\left\{\mathbf{W}^0 : \left|\frac{\partial L(k_1)}{\partial \theta_j}\right| > \left|\frac{\partial L(k_2)}{\partial \theta_j}\right| \ \forall j\right\} \cap B_\delta\right)}{\mu(B_\delta)} > 1 - C\exp(-c/\delta)$$

where B_δ is a ball with radius δ centered at the origin of the $\mathbf{W}^0$-parameter space and $\mu(\cdot)$ is the Lebesgue measure.

[12] ZQJ Xu et al. "Frequency principle: Fourier analysis sheds light on deep neural networks". *Communications in Computational Physics* 28.5 (2020), 1746–1767.

Decomposition in the frequency domain

Assume $\operatorname{supp} \hat{f} \subset [-M\Delta k, M\Delta k]$. We write

$$\hat{f}(k) = \sum_j \hat{f}_j(k)$$

where $\operatorname{supp} \hat{f}_j \subset \{k : (j-1/2)\Delta k \le k \le (j+1/2)\Delta k\}$ for phase shift NNs[13], or $\operatorname{supp} \hat{f}_j \subset \{k : (j-1)\Delta k \le |k| \le j\Delta k\}$ for multiscale NNs[14].

[13] W Cai, X Li, and L Liu. "A phase shift deep neural network for high frequency approximation and wave problems". *SIAM Journal on Scientific Computing* 42.5 (2020), A3285–A3312.

[14] Z Liu, W Cai, and ZQJ Xu. "Multi-scale deep neural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains". *Communications in Computational Physics* 28.5 (2020), 1970–2001.

Phase shift NNs

Correspondingly, we have

$$f(x) = \sum_j f_j(x)$$

where $f_j(x) = \mathcal{F}^{-1}[\hat{f}_j](x)$. Due to the frequency principle, $f_j^{\text{shift}}(x) = e^{-ij\Delta kx} f_j(x)$ can be learned quickly.

A parallel procedure: 1) conduct frequency decomposition; 2) generate training data for $f_j^{\text{shift}}(x)$; 3) train a NN $T_j(x)$ approximate $f_j^{\text{shift}}(x)$; 4) $f(x) \approx \sum_j e^{ij\Delta kx} T_j(x)$.
A coupled procedure: directly consider the NN given by $T(x) = \sum_j e^{i\omega_j x} T_j(x)$.

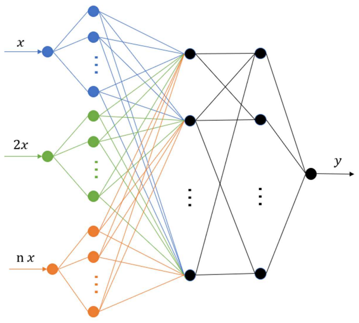
It can be shown that if the weights in input layer are small, the loss functions of the two procedures are approximately equivalent[13].

Multiscale NNs I

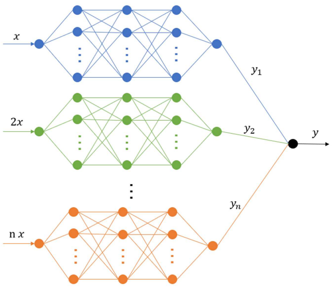
We consider scaling each $\hat{f}_j$ by $\hat{f}_j^{(\text{scale})} = \hat{f}_j(\alpha_j k)$, where $\alpha_j > 1$. Consequently, $f_j^{(\text{scale})}(x) \propto f_j\left(\frac{1}{\alpha_j}x\right)$, which can be learned quickly if α_j is large. This motivates the NN structure $f(x) \approx \sum_j T_j(\alpha_j x)$.

It is also found that activation functions with compact support perform better for MscaleDNNs.

Multiscale NNs II



(a) MscaleDNN-1



(b) MscaleDNN-2

Figure: Two MscaleDNN structures. Reprinted from [14].

Multiscale NNs III

Consider the following Poisson-Boltzmann equation

$$-\nabla(\epsilon(\mathbf{x})\nabla u(\mathbf{x})) + \kappa(\mathbf{x})u(\mathbf{x}) = f(\mathbf{x}),\ \mathbf{x} \in \Omega = [-1,1]^3,$$

and

$$f(\mathbf{x}) = (\mu_1^2 + \mu_2^2 + \mu_3^2 + x_1^2 + 2x_2^2 + 3x_3^2)\sin(\mu_1 x_1)\sin(\mu_2 x_2)\sin(\mu_3 x_3),$$

$$\kappa(\mathbf{x}) = (x_1^2 + 2x_2^2 + 3x_3^2),$$

where $\mu_1 = 15$, $\mu_2 = 20$, and $\mu_3 = 25$.
Two networks are used: 1) a fully-connected DNN with size 1-900-900-900-1 (normal); 2) a MscaleDNN-2 with six subnetworks with size 1-150-150-150-1 and scale coefficients 1,2,4,8,16,32 (Mscale).

Multiscale NNs IV

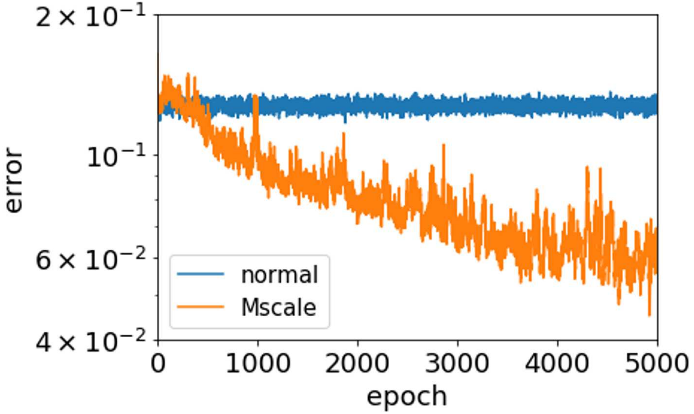


Figure: Training errors for the variable coefficient PB equation. Reprinted from [14].

Domain decomposition

XPINN concepts[15]:

- Subdomains: $\Omega = \bigcup_i \Omega_i$ (non-overlapping);
- Interfaces: common boundary of two subdomains;
- Sub-net: Individual PINN T_i on each subdomain Ω_i with its own hyperparameters;
- **Interface conditions**: problem dependent.

Global representation: $u(x) = \sum_i T_i(x) I_{\Omega_i}(x)$ and averaged on interfaces.

Loss function for each subdomain...

[15] AD Jagtap and GE Karniadakis. "Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations". *Communications in Computational Physics* 28.5 (2020), 2002–2041.

Numerical results I

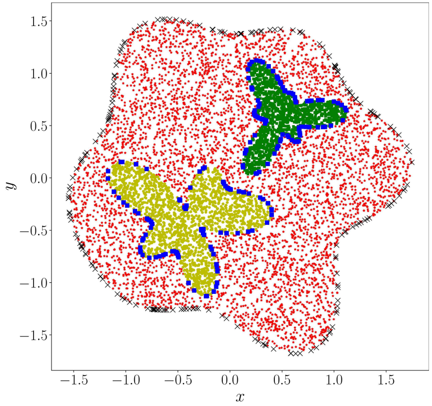


Figure: Solving 2D Poisson problem using 3 subdomains. Reprinted from [15].

Numerical results II

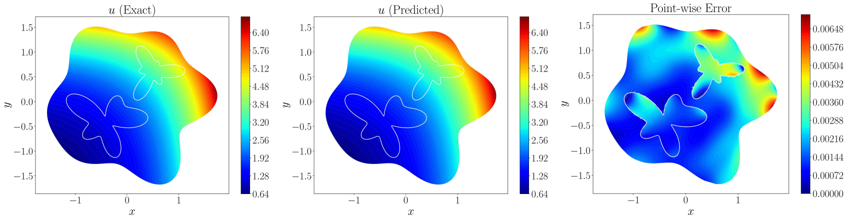


Figure: Exact and numerical solutions. Reprinted from [15].

Thank you.

References I

Cai, W, X Li, and L Liu. “A phase shift deep neural network for high frequency approximation and wave problems”. *SIAM Journal on Scientific Computing* 42.5 (2020), A3285–A3312.

Candès, EJ and DL Donoho. “Ridgelets: a key to higher-dimensional intermittency?” *Philosophical Transactions of the Royal Society A* 357.1760 (1999), 2495–2509.

Dissanayake, MWMG and N Phan-Thien. “Neural-network-based approximations for solving partial differential equations”. *Communications in Numerical Methods in Engineering* 10.3 (1994), 195–201.

E, W, C Ma, and L Wu. “Machine learning from a continuous viewpoint, I”. *Science China Mathematics* 63.11 (2020), 2233–2266.

E, W and B Yu. “The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems”. *Communications in Mathematics and Statistics* 6.1 (2018), 1–12.

References II

Guo, M and JS Hesthaven. "Reduced order modeling for nonlinear structural analysis using Gaussian process regression". *Computer Methods in Applied Mechanics and Engineering* 341 (2018), 807–826.

Han, J, A Jentzen, and W E. "Solving high-dimensional partial differential equations using deep learning". *Proceedings of the National Academy of Sciences* 115.34 (2018), 8505–8510.

Jagtap, AD and GE Karniadakis. "Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations". *Communications in Computational Physics* 28.5 (2020), 2002–2041.

Kashefi, A, D Rempe, and LJ Guibas. *A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries*. 2020. arXiv: 2010.09469 [cs.LG].

References III

- Ladický, L et al. “Data-driven fluid simulations using regression forests”. *ACM Transactions on Graphics* 34.6 (2015).
- Liu, Z, W Cai, and ZQJ Xu. “Multi-scale deep neural network (MscaleDNN for solving Poisson-Boltzmann equation in complex domains”. *Communications in Computational Physics* 28.5 (2020), 1970–2001.
- Saha, S et al. “Hierarchical deep learning neural network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering”. *Computer Methods in Applied Mechanics and Engineering* 373 (2021), 113452.
- Wang, C, LY Xu, and JS Fan. “A general deep learning framework for history-dependent response prediction based on UA-Seq2Seq model”. *Computer Methods in Applied Mechanics and Engineering* 372 (2020), 113357.

References IV

Xu, ZQJ et al. “Frequency principle: Fourier analysis sheds light on deep neural networks”. *Communications in Computational Physics* 28.5 (2020), 1746–1767.

Zhang, L et al. “Hierarchical deep-learning neural networks: finite elements and beyond”. *Computational Mechanics* (2020).