Physics-informed neural networks in complex geometries

Drawn from the works of Berg and Nyström[1], Sukumar and Strivastava[2] & Miao and Li[3]

Alexis Lucas Clément Mazzocchi

École nationale des ponts et chaussées
Institut Polytechnique de Paris

10 mars 2025

[1] Berg and Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries".

[2] Sukumar and Srivastava, "Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks".

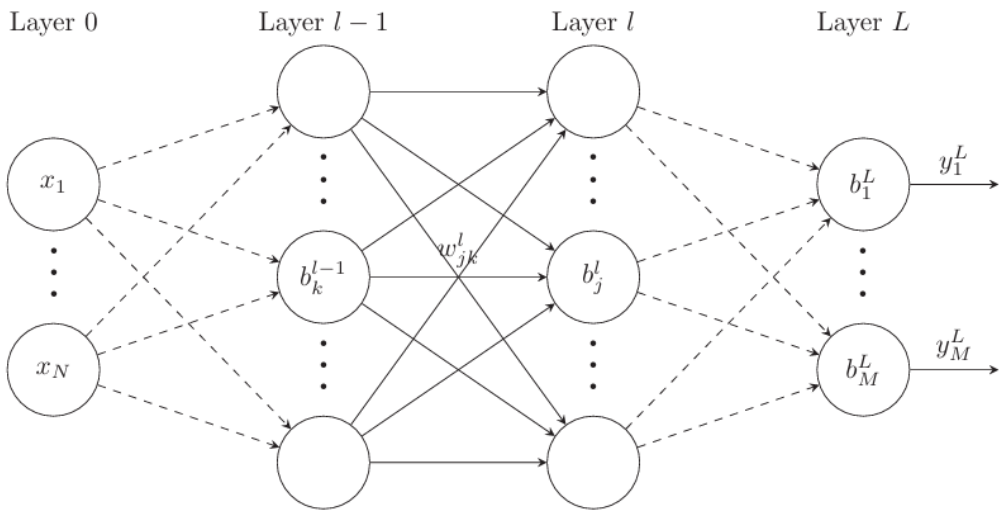
[3] Miao and Li, *GPINN: Physics-informed Neural Network with Graph Embedding*.

Contents

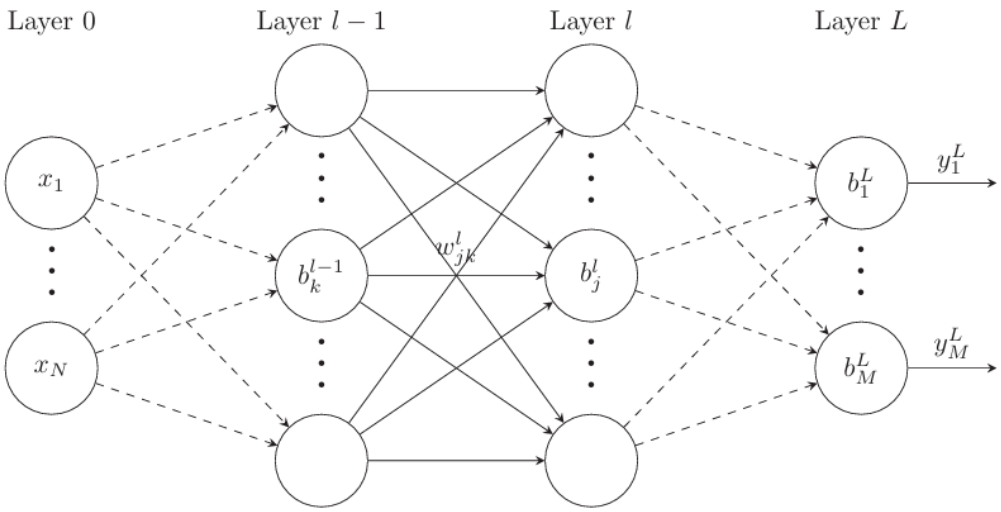
Deep neural networks

We consider a deep fully connected feedforward artifical neural network (ANN) with $L+1$ layers :

- layer $l = 0$: input layer;
- layers $0 < l < L$: hidden layers;
- layer $l = L$: output layer.



Deep neural networks



Parameters:

- Neuron j in layer l is provided by a bias b_j^l;
- Transition between neuron k in layer $l-1$ and neuron j in layer l is performed applying a weight w_{jk}^l.

The activation function of layer l is denoted σ_l.
Notations:

- $x_1, \ldots, x_N$ are the inputs;
- z_j^l is the output of neuron j in layer l before the activation function;
- $y_j^l = \sigma_l(z_j^l)$ so that $y_1^L, \ldots, y_M^L$ are the outputs.

$$\boxed{z_j^l = \sum_k w_{jk}^l \sigma_{l-1}(z_k^{l-1}) + b_j^l = \sum_k w_{jk}^l y_k^{l-1} + b_j^l}$$

Backpropagation

Given a cost function $C(y, y^L)$, our goal is to compute the parameters of the ANN which minimize C:

$$w^*, b^* = \arg\min_{w,b} C(y, y^L)$$

Therefore we need to compute

- $\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} y_k^{l-1}$
- $\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}$

Now, our problem is to determine $\delta_j^l := \frac{\partial C}{\partial z_j^l}$. We can use the backward recursion:

Backpropagation

- $\delta_j^L = \frac{\partial C}{\partial y_j^L}\frac{\partial y_j^L}{\partial z_j^L} = \frac{\partial C}{\partial u_j^L}\sigma_L'(z_j^L)$
- $\delta_j^l = \sum_k \frac{\partial C}{\partial z_k^{l+1}}\frac{\partial z_k l+1}{\partial z_j^l} = \sum_k \delta_k^{l+1} w_{kj}^{l+1}\sigma_l'(z_j^l)$

Contents

Partial differential equations resolution

We are interested in solving the following problem:

$$\begin{cases} Lu = f, & \forall x \in \Omega \\ Bu = g, & \forall x \in \Gamma \subset \partial\Omega \end{cases}$$

where L is a differential operator, f a forcing function, B a boundary operator and g the boundary data.
We design an appropriate cost function:

$$C = \frac{1}{2}\|L\hat{u} - f\|_{L^2}^2 = \frac{1}{2}\int_\Omega \|L\hat{u} - f\|_2^2 \mathrm{d}x$$

with $\hat{u}$ the ansatz computed by the ANN.
After discretizing Ω and Γ into sets of collocation points Ω_d and Γ_d, we fall back to solve the optimization problem:

$$w^*, b^* = \arg\min_{w,b} \frac{1}{2}\frac{1}{|\Omega_d|}\sum_{x_i \in \Omega_d} \|L\hat{u}(x_i) - f(x_i)\|_2^2$$

under the constraints

$$B\hat{u}(x_i) = g(x_i), \quad \forall x_i \in \Gamma_d$$

Integrating the boundary conditions into the cost function

A first way to solve the previous optimization problem is to integrate the boundary constraints into a new term in the cost function, which rewrites:

Cost function of a PINN

$$C = \frac{1}{2}\frac{1}{|\Omega_d|}\sum_{x_i \in \Omega_d} \|L\hat{u}(x_i) - f(x_i)\|_2^2 + \frac{1}{2}\frac{1}{|\Gamma_d|}\sum_{x_i \in \Gamma_d} \|B\hat{u}(x_i) - g(x_i)\|_2^2$$

This is the approach of Raissi et al[4] who first talked about "physics-informed neural networks".
Adding a new term to the cost function comes with instabilities and convergence issues, especially in complex geometries or high-dimensional settings.

[4] Raissi, Perdikaris, and Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations".

Contents

Integrating the boundary conditions into the ansatz

Following the steps of Berg and Nyström, we define our ansatz $\hat{u}$ as

$$\hat{u}(x) = G(x) + D(x)y^L(x)$$

where G is a smooth extension of the boundary data g and D is a smooth distance function that gives the distance of Γ from $x \in \Omega$.
Given that $\forall x \in \Gamma, D(x) = 0$, the boundary conditions are automatically satisfied (we consider Dirichlet conditions here).

Computation of G We require G to verify $\|G(x) - g(x)\|_2 < \epsilon, \quad \forall x \in \Gamma$.
We can compute G using an ANN with the cost function

$$C_G = \frac{1}{2}\frac{1}{|\Gamma_d|}\sum_{x_i \in \Gamma_d} \|G(x_i) - g(x_i)\|_2^2$$

Computation of D We require G to verify $\|D(x)\|_2 < \epsilon, \quad \forall x \in \Gamma$ and $D(x) \approx d(x) := \min_{y \in \Gamma} \|x - y\|_2, \quad \forall x \in \Omega$.
We can compute D using an ANN with the cost function

$$C_D = \frac{1}{2}\frac{1}{|\omega_d| + |\Gamma_d|}\sum_{x_i \in \omega_d \cup \Gamma_d} \|D(x_i) - d(x_i)\|_2^2$$

with $\omega_d \subset \Omega_d$.

We can ensure that $|\Gamma_d| \ll |\Omega_d|$ and $|\omega_d| + |\Gamma_d| \ll |\Omega_d|$ so that the computational cost of G and D is negligible compared to the resolution of the problem.

Contents

Exactly imposing the boundary conditions

In the continuity of the previous approach, we may want to analytically calculate the distance function D to increase the robustness of the method. We present a formalism developed by Sukumar and Strivastava based on R-functions.
Our objective is to approximate the distance function $d(x) = \min_{y \in \Gamma} \|x - y\|_2$.

We will use an approximate distance function (ADF) ϕ which verifies:

Properties of ADFs

- $\phi(x) = 0, \quad \forall x \in \Gamma$
- $\phi(x) > 0, \quad \forall x \in \Omega$
- $\nabla\phi(x) \neq 0, \quad \forall x \in \Gamma$
- $\frac{\partial \phi}{\partial v} = 1, \quad \frac{\partial^k \phi}{\partial v^k} = 0, \quad \forall 2 \leq k \leq m$ with v the unitary normal vector to Γ and m the order of normalization

We will then search a solution of the form $\hat{u}(x) = G(x) + \phi(x) y^L(x)$.
The calculation of ADFs relies on the use of R-functions $F(\omega_1, \ldots, \omega_q)$ whose sign only depends on the signs of its arguments (which are real-valued functions).

R-functions

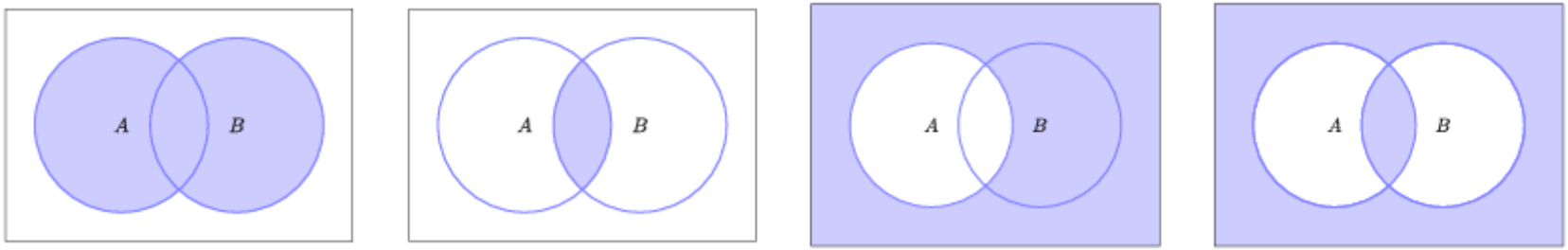
R-functions can represent the geometry of a domain. An elementary R-function ω associated to a set A is non-negative inside A and non-positive outside A.
R-functions can be combined using boolean operations:

Negation (complementary) $\neg\omega = -\omega$

Disjunction (union) $\omega_1 \vee \omega_2 = \max(\omega_1, \omega_2)$

Conjunction (intersection) $\omega_1 \wedge \omega_2 = \min(\omega_1, \omega_2)$

There operations correspond to complementary, union and intersection in a set theory setting. They make possible the description of a wide variety of geometries.



R-functions

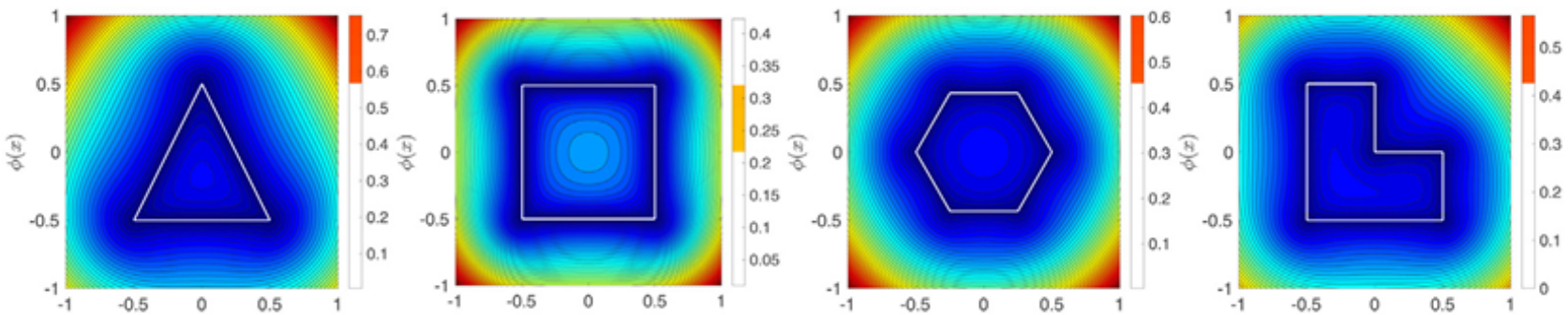
Thanks to R-functions equivalence, one can construct an ADF ϕ for a boundary composed of two elements whose ADFs are ϕ_1 and ϕ_2 respectively:

$$\phi(\phi_1, \phi_2) = \frac{\phi_1 \phi_2}{\sqrt[m]{\phi_1^m + \phi_2^m}}$$

This formula can be easily generalized:

R-functions equivalence

$$\phi(\phi_1, \ldots, \phi_n) = \frac{1}{\sqrt[m]{\frac{1}{\phi_1^m} + \ldots + \frac{1}{\phi_n^m}}}$$



ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES
IP PARIS

Examples

We consider the line segment $[\mathbf{x}_1, \mathbf{x}_2]$.

We denote $L = \|\mathbf{x}_2 - \mathbf{x}_1\|_2$ and $\mathbf{x}_c = \dfrac{\mathbf{x}_1 + \mathbf{x}_2}{2}$.

Construction of an approximate distance function for a line segment

The distance to the line going through $\mathbf{x}_1$ and $\mathbf{x}_2$ from a point $\mathbf{x}$ is

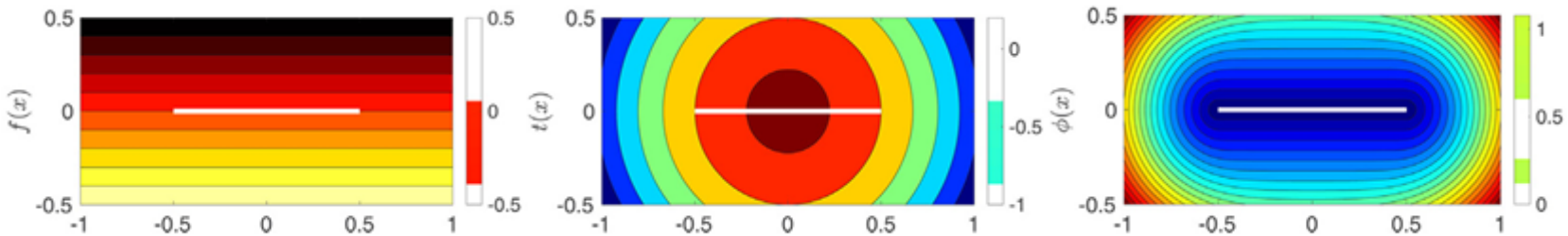
$$f(\mathbf{x}) = \frac{(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1)}{L}$$

Next, we define the trimming function

$$t(\mathbf{x}) = \frac{1}{L}\left(\left(\frac{L}{2}\right)^2 - \|\mathbf{x} - \mathbf{x}_c\|_2^2\right)$$

Finally, we can write the approximate distance function

$$\phi(\mathbf{x}) = \sqrt{f(\mathbf{x})^2 + \left(\frac{\sqrt{t(\mathbf{x})^2 + f(\mathbf{x})^4} - t(\mathbf{x})}{2}\right)^2}$$



Examples

We consider the circular arc of center $\mathbf{x}_c$ and radius R :
$\mathbf{x}(\theta) = \mathbf{x}_c + R(\cos(\theta), \sin(\theta)), \quad \forall \theta \in [\theta_1, \theta_2]$.
We denote $\mathbf{x}_1 = \mathbf{x}_c + R(\cos(\theta_1), \sin(\theta_1))$ and $\mathbf{x}_2 = \mathbf{x}_c + R(\cos(\theta_2), \sin(\theta_2))$.

Construction of an approximate distance function for a circular arc

The distance to the circle superposed on the circular arc from a point $\mathbf{x}$ is

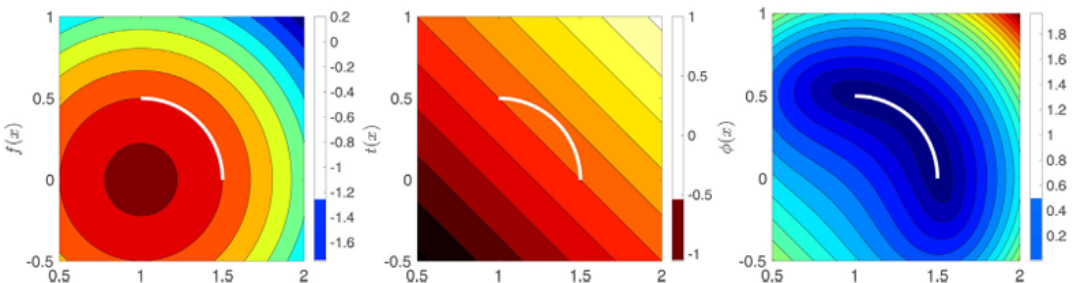
$$f(\mathbf{x}) = \frac{R^2 - \|\mathbf{x} - \mathbf{x}_c\|_2^2}{2R}$$

Next, we define the trimming function

$$t(\mathbf{x}) = \frac{(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1)}{\|\mathbf{x}_2 - \mathbf{x}_1\|_2}$$

Finally, we can write the approximate distance function

$$\phi(\mathbf{x}) = \sqrt{f(\mathbf{x})^2 + \left(\frac{\sqrt{t(\mathbf{x})^2 + f(\mathbf{x})^4} - t(\mathbf{x})}{2} \right)^2}$$



ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES
IP PARIS

Examples

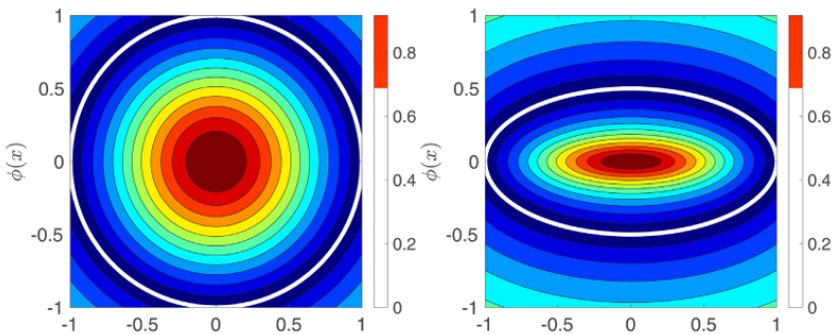
We consider the circle of center $\mathbf{x}_c$ and radius R : $\|\mathbf{x} - \mathbf{x}_c\|_2 = R$.

Approximate distance function for a circle

$$\phi(\mathbf{x}) = \frac{R^2 - \|\mathbf{x} - \mathbf{x}_c\|_2^2}{2R}$$

This formula can be extended to an ellipse whose closure is given by the R-function $\omega(\mathbf{x}) \geq 0$:

$$\phi(\mathbf{x}) = \frac{\omega(\mathbf{x})}{\sqrt{\omega^2(\mathbf{x}) + \|\nabla\omega(\mathbf{x})\|_2^2}}$$



Contents

Limitations of euclidean space and introduction of GPINN

Standard PINNs operate in Euclidean space, which is not in accordance with physical constraints since euclidean distance does not always correspond to the physical distance.

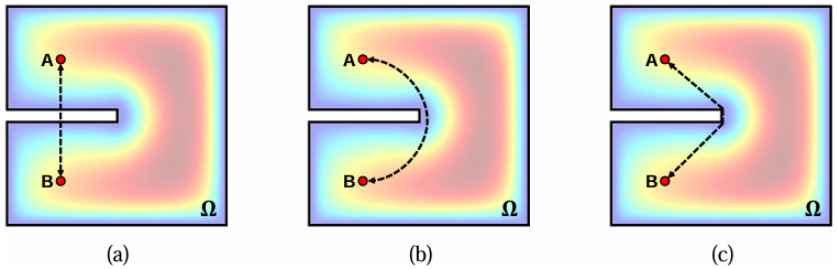


Figure 1: Distances of heat propagation (a) in the input space, (b) in a possible propagation path in physics and (c) in the shortest propagation path in physics.

GPINN Reformulation: Instead of working in an euclidean space (x, t), GPINN introduce an additional dimension z:

$$u_{NN}(x, t, z) : \Omega \to \mathbb{R}$$

The loss function in GPINN remains similar but incorporates topology-informed learning:

$$L = \omega_1 L_{\mathrm{PDE}} + \omega_2 L_{\mathrm{Data}} + \omega_3 L_{\mathrm{IC}} + \omega_4 L_{\mathrm{BC}}$$

Graph embedding and Fiedler Vector

Mathematical formulation of the graph structure and Fiedler vector extraction.

Graph representation

The computational domain is represented as a graph $G = (V, E)$ where:

- V represents mesh nodes, E represents edges.
- The adjacency matrix A encodes node connectivity.
- The degree matrix D is diagonal with $D_{ii} = \sum_j A_{ij}$.

Laplacian matrix and Fiedler vector

The Laplacian matrix is defined as:

$$L = D - A$$

The Fiedler vector v_2 is the second eigenvector of L, capturing the domain's topology:

$$Lv_2 = \lambda_2 v_2$$

The new input space is:

$$(x, t, z), \quad z = v_2$$

Heat propagation: FEM vs PINN vs GPINN

Evaluating GPINN on a heat conduction problem in a 2D house.

Heat Conduction

The steady-state heat equation is given by:

$$\Delta u(x) = f(x), \quad x \in \Omega$$

with boundary conditions:

$$u(x) = u_B(x), \quad x \in \partial\Omega_D$$

$$\nabla u(x) \cdot n = v_B(x), \quad x \in \partial\Omega_N$$

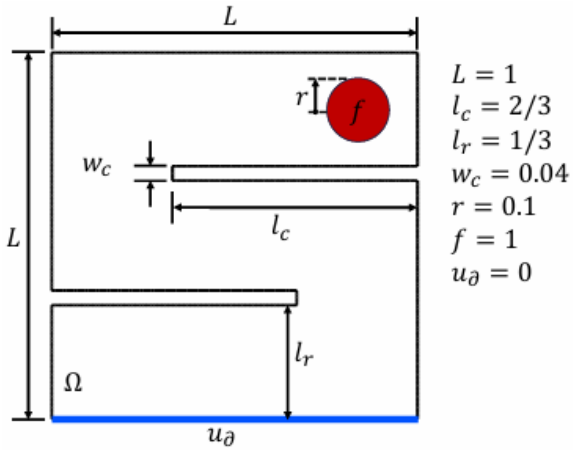


Figure 6: Schematic of the heat propagation problem. The domain of the 2D 'house' is defined as Ω; there is a heat source f in Ω ; the Dirichlet boundary conditoin is assigned on the boundary at the bottom of Ω that represents a 'window' whose temperature u_∂ is the same as 'outside'.

Heat propagation: FEM vs PINN vs GPINN

- FEM provides the reference solution.
- PINN fails to capture heat flow correctly due to domain discontinuities.
- GPINN aligns with physical constraints and improves accuracy.

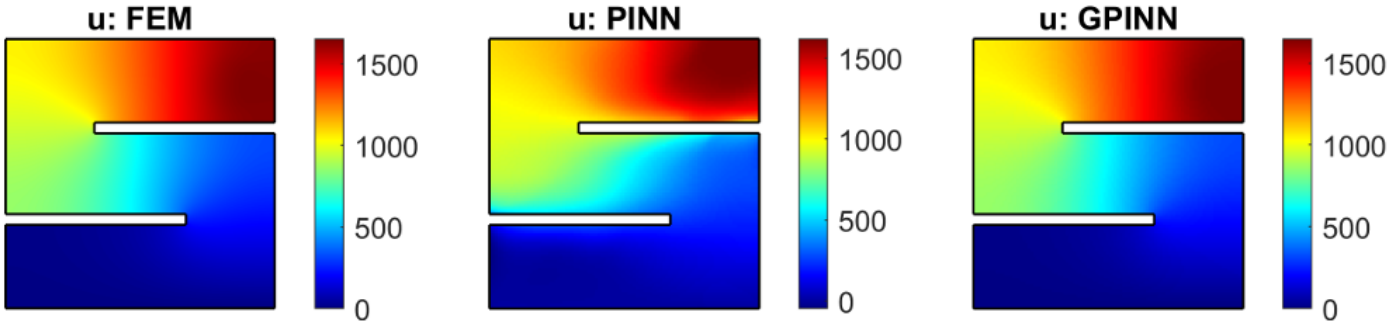


Figure 7: Reference(FEM) and Sample(NN) solutions of the steady temperature field.

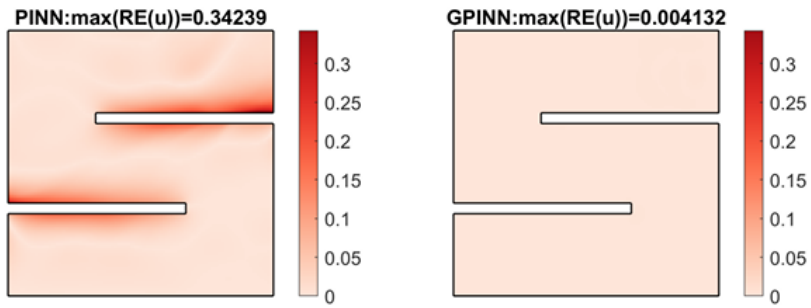


Figure 8: Relative errors (RE) of the NN solutions to the reference FEM solution: $\mathrm{RE}(u) = |u - u^*| / \max(|u^*|)$. The subfigure on the left side is the relative error of PINN while the right one represents the GPINN.

Crack modeling: FEM vs PINN vs GPINN

Evaluating GPINN on a single-side crack problem.

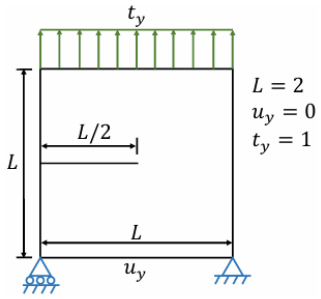


Figure 10: Schematic of the single-side crack tensile test. The Dirichlet and Neumman boundary conditions are assigned as indicated.

Linear Elasticity

The equilibrium equation for linear elasticity is:

$$\nabla \cdot \sigma(x) = 0, \quad x \in \Omega$$

with boundary conditions:

$$\sigma(x) \cdot n = t(x), \quad x \in \partial\Omega_N$$

$$u(x) = u_B(x), \quad x \in \partial\Omega_D$$

where the stress-strain relation is:

$$\sigma(x) = C : \varepsilon(x), \quad \varepsilon(x) = \nabla u(x)$$

Crack Modeling: FEM vs PINN vs GPINN

- FEM captures stress discontinuities accurately.
- PINN struggles with fracture representation.
- GPINN incorporates topology, enabling better crack modeling.

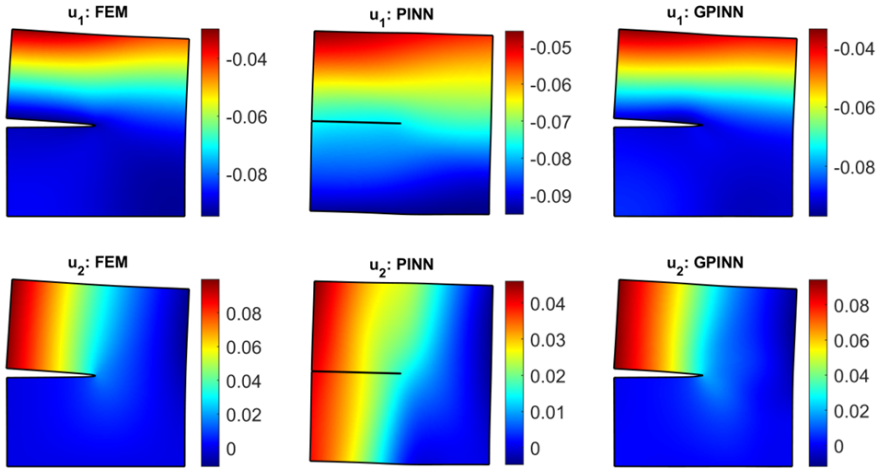


Figure 11: Reference (FEM) and Sample (NN) solutions of the steady temperature field.

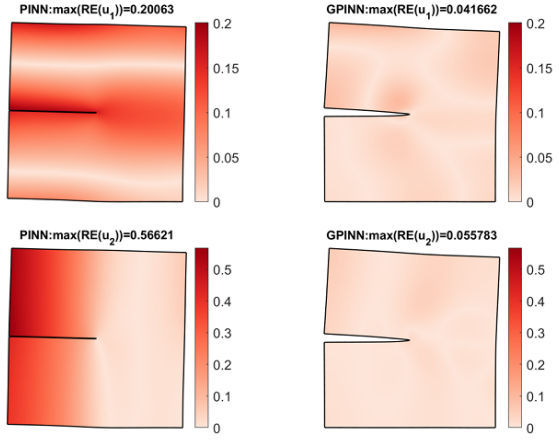


Figure 12: Relative errors (RE) of the NN solutions to the reference FEM solution: $\mathrm{RE}(u) = |u - u^*| / \max(|u^*|)$. The subfigures on the left side are the relative errors of PINN while the right ones represent the GPINN.

Questions?


Berg, Jens and Kaj Nyström. "A unified deep artificial neural network approach to partial differential equations in complex geometries". In: *Neurocomputing* 317 (2018), pp. 28–41. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.06.056. URL: https://www.sciencedirect.com/science/article/pii/S092523121830794X.


Miao, Yuyang and Haolin Li. *GPINN: Physics-informed Neural Network with Graph Embedding*. 2023. arXiv: 2306.09792 [cs.LG]. URL: https://arxiv.org/abs/2306.09792.


Raissi, M., P. Perdikaris, and G.E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.


Sukumar, N. and Ankit Srivastava. "Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks". In: *Computer Methods in Applied Mechanics and Engineering* 389 (2022), p. 114333. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114333. URL: https://www.sciencedirect.com/science/article/pii/S0045782521006186.