Deep Learning Method for Plasma Simulation

An Adaptive Deep Nitsche Method for Elliptic Interface Problem

Qimu Luo, Hanzhe Xue
Supervisor: Dr. Ren Zhao, Dr. Yiru Ye

Department of Foundational Mathematics, School of Mathematics and Physics

SURF-2024-0016
SURF
Summer Undergraduate Research Fellowship

Abstract

In this project, we propose a gradient-based adaptive sampling strategy for the deep unfitted Nitsche method to solve the elliptic interface problem. The deep unfitted Nitsche method involves training two weakly coupled neural networks by minimizing an energy formualtion to address the discontinuities caused by the interface. The gradient-based adaptive sampling strategy is then designed to capture the singularity at the interface for more challenging problems. The algorithm's performance is presented through a numerical example to demonstrate its computational efficiency and accuracy.

Model Problem

The algorithm in this project is designed to solve the following elliptic interface problem utilizing the universal approximation property of neural networks.

$$\begin{aligned} -\nabla\cdot(\beta(x)\nabla u(x)) &= f, && \text{in } \Omega_1\cup\Omega_2 \\ u &= g, && \text{on } \partial\Omega \\ [\![u]\!] &= p, && \text{on } \Gamma \\ [\![\partial_n u]\!] &= q, && \text{on } \Gamma \end{aligned}$$

β as the diffusion coefficient is a piecewise function β_i for $\mathbf{x}\in\Omega_i$, the jump $[\![w]\!]$ on Γ is defined as $[\![w]\!] = w_2|_\Gamma - w_1|_\Gamma$ with $w_i = w|_{\Omega_i}$ being the restriction of w on Ω_i, and $\partial_n u = (\nabla u)\cdot n$ refers to the unit outward normal vector of the interface Γ.

Unfitted Nitsche Energy Functional

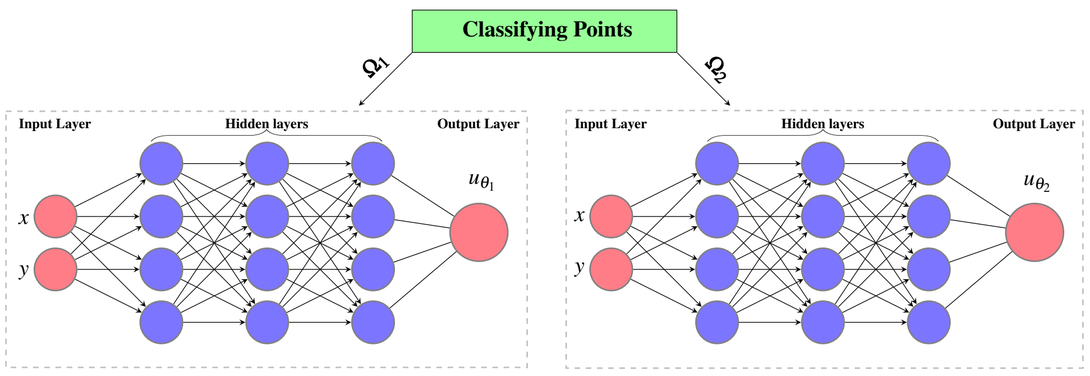
The unfitted Nitsche method is weakly formulated first to find $u\in H^1(\Omega_1\cup\Omega_2)$ such that $a(u,v) = \ell(v), \forall v\in H_0^1(\Omega_1\cup\Omega_2)$. The variational problem is reformulated into a minimization problem of the Nitsche energy functional $L_n(v)$ for the neural networks [1]:

$$\begin{aligned} L_n(v) :=& \frac{1}{2}a(v,v) - \ell(v) \\ =& \frac{1}{2}\sum_{i=1}^{2}\int_{\Omega_i}\beta_i\nabla v_i\cdot\nabla v_i dx + \frac{\gamma_f}{2}\int_\Gamma([\![v]\!]-p)^2 ds + \int_\Gamma([\![v]\!]-p)\{\!\{\beta\partial_n v\}\!\}ds - \int_\Omega fv dx + \int_\Gamma q\{\!\{v\}\!\}^* ds - \frac{\gamma_f}{2}\int_\Gamma p^2 ds. \end{aligned}$$

γ_f represents the stability parameter, $\{\!\{\cdot\}\!\}$ and $\{\!\{\cdot\}\!\}^*$ denote the weighted and dual weighted averaging of a function respectively *i.e.*, $\{\!\{w\}\!\} = \kappa_1 w_1|_\Gamma + \kappa_2 w_2|_\Gamma$ with κ refers to the weights of β.

Deep Neural Networks

The feedforward neural networks are selected as the ansatz function for approximation:



The l-th layer of the DNN is defined as:

$$\mathscr{N}^l(x^l) = \sigma(\boldsymbol{T}^l(\mathbf{x}^l)) = \sigma(\mathbf{W}^l\mathbf{x}^l + \mathbf{b}^l)$$

A L-layer DNN is defined as the composition for all $\mathscr{N}^l$:

$$\mathscr{NN}(\mathbf{x};\Theta) = \boldsymbol{T}^l\circ\mathscr{N}^{l-1}\circ\cdots\circ\mathscr{N}^2\circ\mathscr{N}^1(\mathbf{x})$$

where Θ stands for the sets of parameters, $\mathbf{W}\in\mathbb{R}^{m\times m}$ and $\mathbf{b}\in\mathbb{R}^m$ refers to the weights and biases respectively. Adding the boundary term and discretizing $L_n(v)$ by the Monte-Carlo method finalizes the preparation of the loss function for the neural networks, where $|\cdot|$ denotes the measure.

$$\begin{aligned} \hat{L}(u_\Theta) =& \frac{|\Omega_1|}{N_1}\sum_{k=1}^{N_1}\left(\frac{\beta_1}{2}\nabla u_{\theta_1}(x_k^1)\cdot\nabla u_{\theta_1}(x_k^1) - f(x_k^1)u_{\theta_1}(x_k^1)\right) + \frac{|\Omega_2|}{N_2}\sum_{k=1}^{N_2}\left(\frac{\beta_2}{2}\nabla u_{\theta_2}(x_k^2)\cdot\nabla u_{\theta_2}(x_k^2) - f(x_k^2)u_{\theta_2}(x_k^2)\right) \\ &+\frac{|\Gamma|}{N_f}\sum_{k=1}^{N_f}\left(\frac{\gamma_f}{2}([\![u_\Theta(x_k^f)]\!]-p(x_k^f))^2 - \frac{\gamma_f}{2}p(x_k^f)^2\right) + \frac{|\Gamma|}{N_f}\sum_{k=1}^{N_f}\left(([\![u_\Theta(x_k^f)]\!]-p(x_k^f))\{\!\{\beta\partial_n u_\Theta(x_k^f)\}\!\}\right) \\ &+\frac{|\Gamma|}{N_f}\sum_{k=1}^{N_f}\left(q(x_k^f)\{\!\{u_\Theta(x_k^f)\}\!\}^*\right) + \frac{|\partial\Omega|}{N_b}\sum_{k=1}^{N_b}\gamma_b\left(u_{\theta_2}(x_k^b) - g(x_k^b)\right)^2 \end{aligned}$$

Adaptive Sampling Algorithm

An algorithm is designed to distribute sampling points for $\{x_k^r\}_{k=1}^{N_r}$ in subdomains, $\{x_k^f\}_{k=1}^{N_f}$ on interfaces, and $\{x_k^b\}_{k=1}^{N_b}$ on boundaries during the training process [2, 3].

Algorithm 1: Gradient only-based adaptive deep Nitsche algorithm

1 Initialization: Set the initial training data size: $H\times H$ of a grid and N_f on the interface.
2 **while** *Number of refinements* $< N$ **do**
3 Train the neural networks for fixed epochs to solve the objective problem;
4 Global refinement: uniformly add more points based on the previous refinement level;
5 Compute the gradient magnitude $\mathbf{I}_t = |\nabla u|$ for each added point t of the current level;
6 Rank the gradients and keep the top-ranked 10% points to update $\{x_k^r\}$, $\{x_k^f\}$, and $\{x_k^b\}$;
7 **end**

Numerical Example

Consider the sunflower interface problem in the domain $\Omega = [-1,1]^2$, the interface is defined as:

$$\begin{cases} x(t) = r(\theta)\cos(\theta) + x_c \\ y(t) = r(\theta)\sin(\theta) + y_c \end{cases}$$

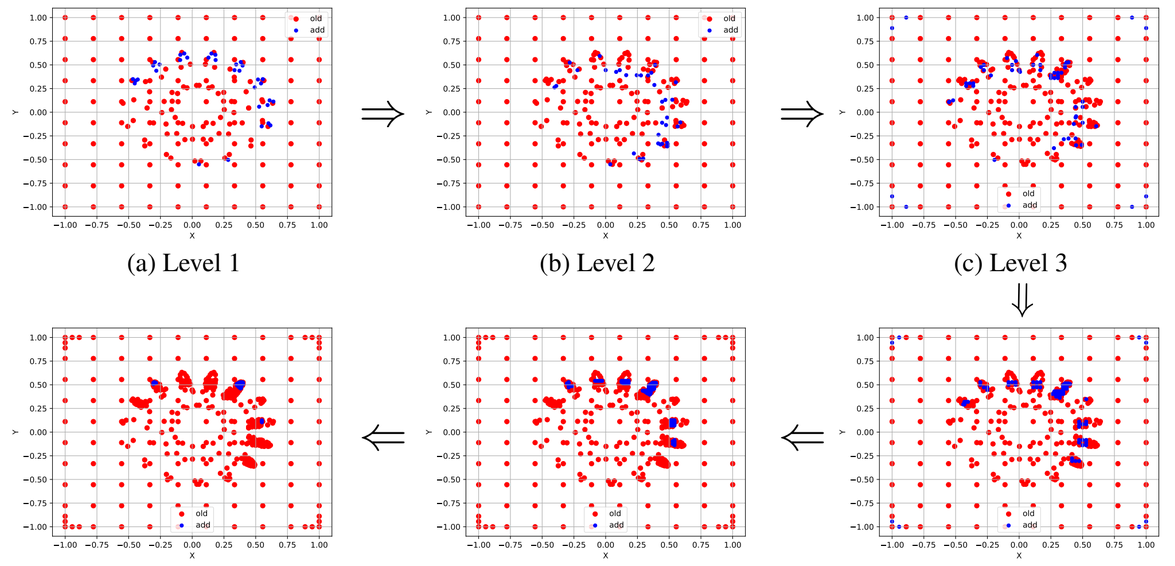
where $r(\theta) = r_0 + r_1\sin(\omega\theta)$, $\theta\in[0,2\pi]$, $r_0 = 0.4, r_1 = 0.2, \omega = 15$ and $x_c = y_c = 0.02\sqrt{5}$. The level set function is described as $(x-x_c)^2 + (y-y_c)^2 = r(\theta)^2$. The piecewise diffusion coefficients are $\beta_1 = 1$ and $\beta_2 = 10$ in the following exact solution:

$$u(\mathbf{x}) = \begin{cases} \frac{r^2}{\beta_1}, & \text{if } \mathbf{x}\in\Omega_1, \\ \frac{r^4 - 0.1ln(2r)}{\beta_2}, & \text{if } \mathbf{x}\in\Omega_2. \end{cases}$$

The parameters are as follows:

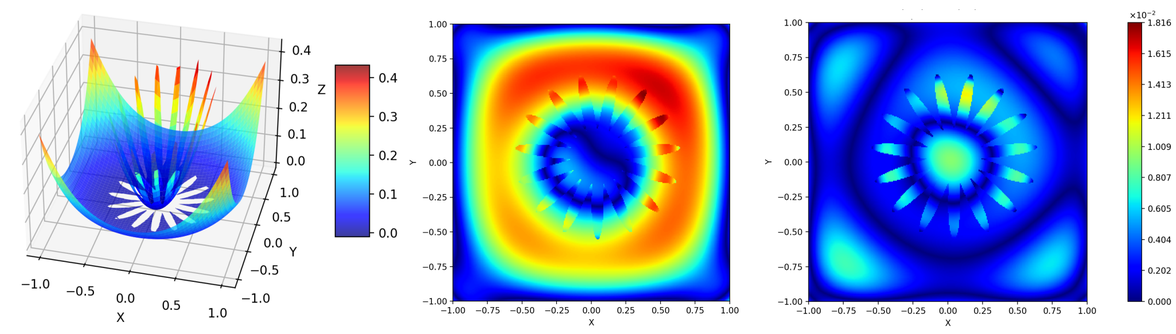
- Grid size: 10×10; Interface points: $N_f = 75$
- Architecture of the neural networks: 64 neurons and 5 layers
- Activation function: $\sigma = tanh$; Optimizer: ADAM with a learning rate $lr = 1e-3$
- Epochs for each level: 1500

The adaptive sampling process over 6 refinements and the cooresponding numerical solution are plotted in the following figures, which capture the singularity on the complex interface accurately.



(a) Level 1 (b) Level 2 (c) Level 3

(f) Level 6 (e) Level 5 (d) Level 4



Numerical Solution Non-adaptive Error Adaptive Error

The relative L^2-error $= \frac{\|u-u_\Theta\|_{0,\Omega_1\cup\Omega_2}}{\|u\|_{0,\Omega_1\cup\Omega_2}}$ is utilized to quantify the performance and make comparisons with random sampling strategy. The results are demonstrated in the following table:

Sampling Method	Point Distribution	Net Type	N_r	N_f	N_b	Training Time (s)	Relative L^2 Error (%)
Adaptive	Level 1	FcNet	66	107	36	6.78	130.85
	Level 2		76	133	36	7.06	37.58
	Level 3		108	150	44	7.72	8.78
	Level 4		182	153	52	9.17	7.50
	Level 5		315	153	52	12.72	3.82
	Level 6		452	153	52	21.29	3.54
Non-adaptive	Random	ResNet	1024	256	256	82.91	5.02
	Random	FcNet	1024	256	256	100.27	9.07

Notably, the adaptive sampling strategy is superior to random sampling methods over 9000 iterations both in accuracy and efficiency due to the efficient usage of resources and it indeed reduces error near the interface in the figure depicting the distribution of errors, matching the sampling process above.

Conclusion

In the project, an adaptive sampling algorithm is combined with the deep unfitted Nitsche method to solve the elliptic interface problem. The adaptive sampling strategy effectively captures the singularity along the interface and from the comparison with the traditional sampling method, it demonstrates superiority in both accuracy and efficiency. We aim to explore more hybrid deep learning methods for the numerical solution of different types of PDEs in the future.

Acknowledgements

This project is supported by XJTLU Summer Undergraduate Research Fellowship SURF-2024-0016, Research Development Fund RDF-22-01-063, and the Ministry of Education of the People's Republic of China Chunhui Programme (Grant No. 202201217).

References

[1] Hailong Guo and Xu Yang. Deep unfitted nitsche method for elliptic interface problems.*Communications in Computational Physics*, 31(4):11621179, June 2022.
[2] Cuiyu He, Xiaozhe Hu, and Lin Mu. A mesh-free method using piecewise deep neural network for elliptic interface problems.*Journal of Computational and Applied Mathematics*, 412:114358, 2022.
[3] Zhiping Mao and Xuhui Meng. Physics-informed neural networks with residual/gradient-based adaptive sampling methods for solving partial differential equations with sharp solutions. *Applied Mathematics and Mechanics*, 44(7):1069–1084, 2023.