Solving interface problems with deep learning

Zunding Huang

UCSD - ML Seminar

10/27/2023

Outline

1. Deep Ritz method
2. Interface problems
3. Some numerical results

Solving interface problems with deep learning

Introduction of some DNN-based PDE solver

- Deep Ritz method: Solve Possion problems and eigenvalue problems from variational principles.
- PINN & DGM: Train DNNs to approximate the solution by minimizing the residual of the PDEs and also of the initial and boundary conditions.

Deep Ritz method

Deep Ritz method: Deep NN + Ritz method, for solving variational problem. If we want to solve the following Possion's equation:

$$\begin{cases} -\Delta u = f \text{ in } \Omega, \\ u = g \text{ in } \partial\Omega. \end{cases}$$

It is equivalent to

$$\min_{u \in H} I(u)$$

where

$$I(u) = \int_{\Omega} \left(\frac{1}{2} |\nabla u(x)|^2 - f(x)u(x) \right) dx$$

and

$$H = \{u \in H^1(\Omega) : u = g \text{ on } \partial\Omega\}$$

Solving interface problems with deep learning

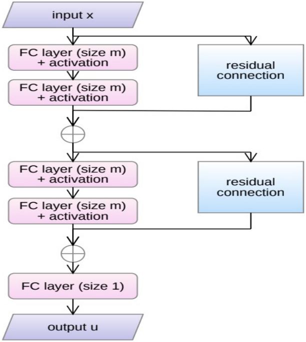


Figure 1: A network with two blocks and an output linear layer. Each block consists of two fully-connected layers and a skip connection.

Solving interface problems with deep learning

Building trial function

The basic component of DR method is a nonlinear transformation

$$\mathrm{x} \in \mathbb{R}^{\mathrm{n}} \;\rightarrow \mathrm{u}_\theta(\mathrm{x}) \in \mathbb{R}$$

defined by a deep neural network.
The i-th block can be expressed by

$$\mathrm{t} = \mathrm{f_i}(\mathrm{s}) = \phi\left(\mathrm{W_{i,2}} \cdot \phi\left(\mathrm{W_{i,1}s} + \mathrm{b_{i,1}}\right) + \mathrm{b_{i,2}}\right) + \mathrm{s}$$

where $\mathrm{W_{i,1}}, \mathrm{W_{i,2}} \in \mathbb{R}^{\mathrm{m}\times\mathrm{m}}, \mathrm{b_{i,1}}, \mathrm{b_{i,2}} \in \mathbb{R}^{\mathrm{m}}$ and ϕ is the activiation function.
The full n-block network can be expressed as

$$\mathrm{u}_\theta(\mathrm{x}) = \mathrm{f_n} \circ \mathrm{f_{n-1}} ... \circ \mathrm{f_1}(\mathrm{x})$$

where θ represents all the parameters in the neural network.

Solving interface problems with deep learning

Building trial function

Denote

$$h(x;\theta) = \frac{1}{2}|\nabla_x u(x;\theta)|^2 - f(x)u(x;\theta)$$

Then original problem

$$\begin{cases} \min_{u\in H} \ I(u), \\ I(u) = \int_\Omega \left(\frac{1}{2}|\nabla u(x)|^2 - f(x)u(x)\right) dx \end{cases}$$

will be converted to a numerical optimization problem:

$$\begin{cases} \min_{\theta} \ L(\theta), \\ L(\theta) = \int_\Omega h(x;\theta) dx \end{cases}$$

Solving interface problems with deep learning

Building trial function

Since u belongs to admissible set H, where

$$\mathrm{H} = \{\mathrm{u} \in \mathrm{H}^1(\Omega) : \mathrm{u} = \mathrm{g} \text{ on } \partial\Omega\}.$$

In real, we will use a penalty method and the numerical optimization problem should be:

$$\begin{aligned}\min_{\theta}\ \mathrm{L}(\theta), \mathrm{L}(\theta) &= \int_{\Omega} \mathrm{h}(\mathrm{x};\theta)\mathrm{dx} + \beta \int_{\partial\Omega} (\mathrm{u}-\mathrm{g})^2 \mathrm{dx} \\ &\approx \int_{\Omega} \left(\frac{1}{2}|\nabla_{\mathrm{x}} \mathrm{u}(\mathrm{x};\theta)|^2 - \mathrm{f}(\mathrm{x})\mathrm{u}(\mathrm{x};\theta)\right) \mathrm{dx} + \\ &\qquad \beta \int_{\partial\Omega} (\mathrm{u}(\mathrm{x};\theta) - \mathrm{g})^2 \mathrm{dx}\end{aligned}$$

where β is the penalty coefficient.

Solving interface problems with deep learning

Stochastic gradient descent and numerical quadrature rule

Combined with Monte Carlo Sampling, the optimization problem often takes the form of:

$$\min_{\theta}\ \mathrm{L}(\theta), \mathrm{L}(\theta) = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{L}_{\mathrm{i}}(\theta)$$

where each $\mathrm{L}_{\mathrm{i}}(\theta)$ corresponds to a data point and N is typically very large. SGD in this context is given by

$$\theta^{\mathrm{k}+1} = \theta^{\mathrm{k}} - \eta\nabla_{\theta}\frac{1}{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}} \mathrm{h}(\mathrm{x}_{\mathrm{j,k}}; \theta^{\mathrm{k}})$$

where $\{\mathrm{x}_{\mathrm{j,k}}\}$ is a set of points in Ω that are randomly sampled with uniform distribution.

Solving interface problems with deep learning

Conclusion

Advantages:

- It is less sensitive to the dimensionality of the problem and has the potential to work in rather high dimensions.
- The method is reasonably simple and fits well with the stochastic gradient descent framework commonly used in deep learning.

Solving interface problems with deep learning

Interface problems (usually in molecular solvation)

Interface problems have many applications in physics and biology.

- Heterogeneous porous medium in the reservoir simulation, the permeability field is often assumed to be a multiscale function with high-contrast and discontinuous features.
- Evolution of the shape and location of fibroblast cells under stress. The cell is modeled as a transformed inclusion in a linear elastic matrix and the cell surface evolves according to a kinetic relation.

Today we will mainly talk about the first type of the PDE: It is an elliptic PDE with a discontinuous and high-contrast coefficient.

Solving interface problems with deep learning

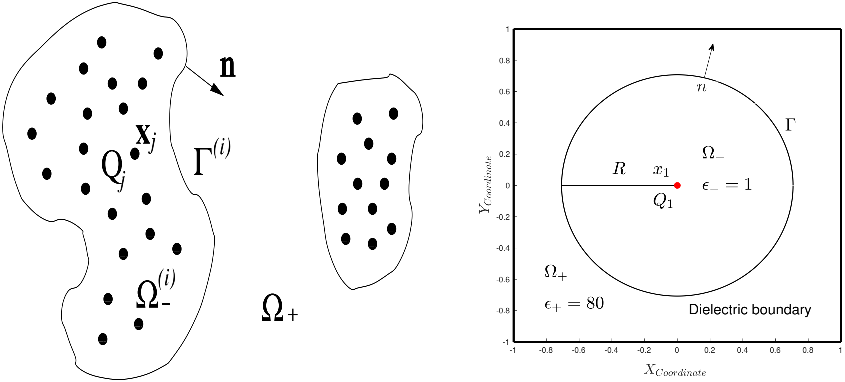


Figure 2: (Left) A schematic diagram of charged molecules immersed in an aqueous solvent. The region of solvation Ω is divided by the solute. (Right) A diagram of charged molecules where the region Ω is the cube $[-1, 1]^3$ and the dielectric boundary Γ is a sphere.

Solving interface problems with deep learning

Interface problems

Some numerical methods for interface problems include finite element method and finite difference method:

- Immersed-interface FEM: Second-order convergence in L_2 norm and first-order convergence in H_1 semi-norm. The constants in the error estimate may depend on the contrast of the coefficient.
- Another FEM: Use coarse quasi-uniform meshes. The constants in the error estimate are independent of the contrast of the coefficients.
- Immersed boundary method: Study the motion of surfaces immersed in an incompressible fluid.
- Immersed interface method: Combine the jump condition with finite difference schemes near the interface, second order convergence.
- Ghost fluid method: Incorporated the jump condition into the finite difference schemes with a level set function.

Solving interface problems with deep learning

Interface problems

Let Ω be a C^2, closed surface such that $\Gamma \subset \Omega$. Denote Ω_- the interior of Γ and $\Omega_+ = \Omega \backslash \Omega_-$. So, $\Omega = \Omega_- \cup \Omega_+ \cup \Gamma$. Here, Ω_- and Ω_+ are the solute and solvent regions, respectively, and Γ is the dielectric boundary. As before, we denote by n the unit normal at Γ pointing from Ω_- to Ω_+. Dielectric coefficient $\varepsilon_\Gamma : \Omega \to \mathbb{R}$ defined as $\varepsilon_\Gamma = \varepsilon_-$ in Ω_- and $\varepsilon_\Gamma = \varepsilon_+$ in Ω_+, where ε_- and ε_+ two distinct positive constants.
Typically, interface problems will be like:

$$\begin{cases} -\nabla \cdot \varepsilon_\Gamma \nabla \mathrm{u} = \mathrm{f} \text{ in } \Omega, \\ \mathrm{u} = \mathrm{g} \text{ in } \partial\Omega. \end{cases}$$

with some additional assumptions: u and $\varepsilon_\Gamma \nabla \mathrm{u} \cdot \mathrm{n}$ is continuous over Γ. This implies $[[\mathrm{u}]] = [[\varepsilon_\Gamma \nabla \mathrm{u} \cdot \mathrm{n}]] = 0$ on Γ.

Solving interface problems with deep learning

Interface problems

Above interface problems can be rewritten as:

$$\begin{cases} \varepsilon_- \Delta \mathrm{u} = -\mathrm{f} \text{ in } \Omega_-, \\ \varepsilon_+ \Delta \mathrm{u} = -\mathrm{f} \text{ in } \Omega_+, \\ [[\mathrm{u}]] = [[\varepsilon_\Gamma \nabla \mathrm{u} \cdot \mathrm{n}]] = 0 \text{ on } \Gamma, \\ \mathrm{u} = \mathrm{g} \text{ in } \partial\Omega. \end{cases}$$

To apply Deep Ritz method to interface problems, we need to formulate the elliptic PDEs into the variational form. The energy of the system should be

$$\mathrm{I}(\mathrm{v}) = \int_\Omega \left(\frac{\varepsilon_\Gamma}{2} |\nabla \mathrm{v}(\mathrm{x})|^2 - \mathrm{f}(\mathrm{x})\mathrm{v}(\mathrm{x}) \right) \mathrm{dx},$$

and v belongs to admissible set $\mathrm{H} = \{\mathrm{v} \in \mathrm{H}^1(\Omega) : \mathrm{v} = \mathrm{g} \text{ on } \partial\Omega\}$. So $\mathrm{u} = \text{argmin}_{\mathrm{v} \in \mathrm{H}} \mathrm{I}(\mathrm{v})$.

Solving interface problems with deep learning

Inhomogeneous boundary conditions

Instead of penalty method, we will use a shallow neural network to approximate the boundary condition $g(x)$.

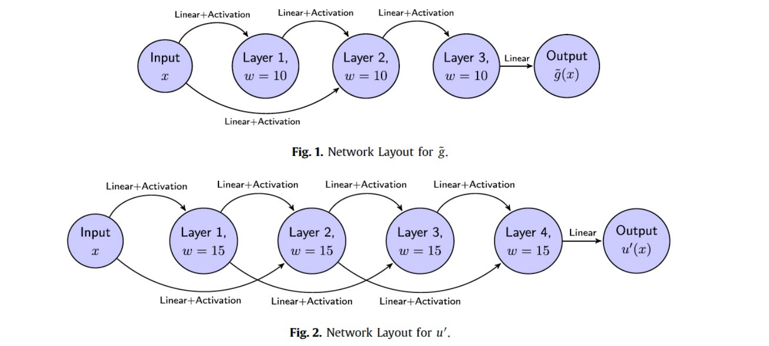


Figure 3: Neural Network configuration for $\tilde{g}$ and u'.

Solving interface problems with deep learning

Inhomogeneous boundary conditions

Here $\tilde{g}$ is the approximation of g. It is defined as

$$\tilde{g} = \text{argmin}_{G \in \mathbb{G}} \left(\int_{\partial\Omega} (G - g(x))^2 dx \right).$$

where $\mathbb{G}$ denotes the set of all expressible functions by a shallow neural network. This can be approximated by

$$\frac{\text{vol}(\partial\Omega)}{N_0} \sum_{i=1}^{N_0} (G(y_i) - g(y_i))^2.$$

where $y_i \sim \text{Uniform}(\partial\Omega)$.

Solving interface problems with deep learning

Inhomogeneous boundary conditions

For inhomogeneous Dirichlet problem

$$\begin{cases} \mathrm{Lu} = \mathrm{f} \text{ in } \Omega, \\ \mathrm{u} = \mathrm{g} \text{ on } \partial\Omega. \end{cases}$$

Once we have approximation $\tilde{\mathrm{g}}$, we can solve the following homogeneous Dirichlet problem:

$$\begin{cases} \mathrm{Lu}' = \mathrm{f} - \mathrm{L}\tilde{\mathrm{g}} \text{ in } \Omega, \\ \mathrm{u}' = 0 \text{ on } \partial\Omega. \end{cases}$$

Final solution u of original problem can be expressed by:

$$\mathrm{u} = \mathrm{u}' + \tilde{\mathrm{g}}$$

Solving interface problems with deep learning

Conclusion for now

Advantages:

- We solve PDEs through their corresponding variational problems, which avoids the need to compute high-order derivatives of the solution.
- A shallow neural network to approximate boundary condition allows us to simply impose inhomogeneous boundary conditions and reduce computational costs in the training process.

Solving interface problems with deep learning

Elliptic PDEs with discontinuous and high-contrast coefficients

We consider homogeneous elliptic PDEs with discontinuous coefficients:

$$\begin{cases} \mathrm{Lu} = -\nabla \cdot (\mathrm{a}(\mathrm{x})\nabla \mathrm{u}(\mathrm{x})) = \mathrm{f} \text{ in } \Omega, \\ \mathrm{u} = 0 \text{ on } \partial\Omega. \end{cases}$$

$\mathrm{a}(\mathrm{x})$ can be piecewise constant function. The variational problem is

$$\mathrm{J}(\mathrm{v}) = \int_\Omega \left(\frac{\mathrm{a}(\mathrm{x})}{2} |\nabla \mathrm{v}(\mathrm{x})|^2 - \mathrm{f}(\mathrm{x})\mathrm{v}(\mathrm{x}) \right) \mathrm{dx}, \mathrm{v} \in \mathbb{H}_0^1(\Omega),$$

and the total functional in terms of θ is

$$\mathrm{J}(\mathrm{x};\theta) = \int_\Omega \left(\frac{\mathrm{a}(\mathrm{x})}{2} |\nabla_\mathrm{x} \mathrm{v}(\mathrm{x};\theta)|^2 - \mathrm{f}(\mathrm{x})\mathrm{v}(\mathrm{x};\theta) \right) \mathrm{dx} + \frac{1}{\varepsilon} \int_{\partial\Omega} \mathrm{v}(\mathrm{x};\theta)^2 \mathrm{dS}.$$

Solving interface problems with deep learning

Elliptic PDEs with discontinuous and high-contrast coefficients

In order to exploit SGD method, we need calculate following:

$$\begin{aligned}\frac{\partial J(x;\theta)}{\partial \theta_k} &= \int_\Omega \frac{\partial}{\partial \theta_k}\left(\frac{a(x)}{2}|\nabla_x v(x;\theta)|^2 - f(x)v(x;\theta)\right)dx \\ &\quad + \frac{1}{\varepsilon}\int_{\partial\Omega}\frac{\partial\left(v(x;\theta)^2\right)}{\partial\theta_k}dS, \\ &\approx \frac{\mathrm{vol}(\Omega)}{N_1}\sum_{i=1}^{N_1}\frac{\partial}{\partial\theta_k}\left(\frac{a(x_i)}{2}|\nabla_x v(x_i;\theta)|^2 - f(x_i)v(x_i;\theta)\right) \\ &\quad + \frac{\mathrm{vol}(\partial\Omega)}{\varepsilon N_2}\sum_{j=1}^{N_2}\frac{\partial\left(v(y_j;\theta)^2\right)}{\partial\theta_k}.\end{aligned}$$

Elliptic PDEs with discontinuous and high-contrast coefficients

Here $\mathrm{x_i} \sim$ Uniform(Ω) and $\mathrm{y_j} \sim$ Uniform$(\partial\Omega)$. $\mathrm{N} = \mathrm{N_1} + \mathrm{N_2}$ is called the batch number in the context of deep learning, which implies the number of collocation points used in one iteration.

The parameters θ_k can be updated by

$$\theta_\mathrm{k}^{\mathrm{n}+1} = \theta_\mathrm{k}^\mathrm{n} - \eta \frac{\partial \mathrm{J}(\mathrm{x};\theta)}{\partial \theta_\mathrm{k}}\Big|_{\theta_\mathrm{k}=\theta_\mathrm{k}^\mathrm{n}}.$$

Numerical tests

Let $\Omega = [-1, 1]^2$ and $x = (x_1, x_2)$. The coefficient $a(x)$ is a piecewise constant defined by

$$a(x) = \begin{cases} a_1, r < r_0, \\ a_0, r \geq r_0. \end{cases}$$

where $r = (x_1^2 + x_2^2)^{\frac{1}{2}}$ and $r_0 = \pi/6.28$. Source function $f(x) = -9r$ and boundary function $g(x) = \frac{r^3}{a_0} + (\frac{1}{a_1} - \frac{1}{a_0})r_0^3$.
Based on the the info we have, the unique exact solution is

$$u(r, \theta) = \begin{cases} \frac{r^3}{a_1}, r < r_0, \\ \frac{r^3}{a_0} + (\frac{1}{a_1} - \frac{1}{a_0})r_0^3, r \geq r_0. \end{cases}$$

Solving interface problems with deep learning

Numerical test1

Choose $a_0 = 10^3, a_1 = 1$.

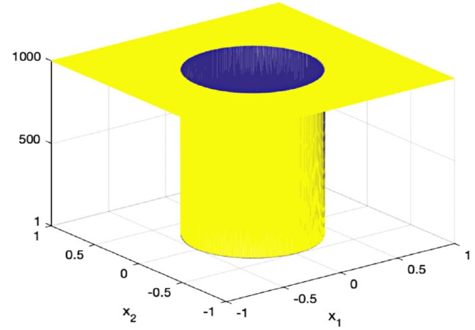
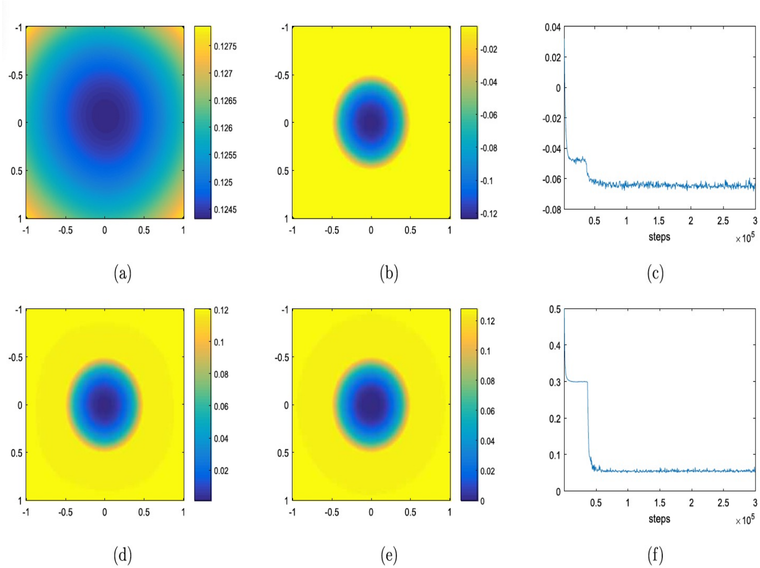


Fig. 3. Profile of a high-contrast coefficient on *D*.

Figure 4: Profile of the coefficient.

Solving interface problems with deep learning



Numerical test1

Batch size is $4096 = 3840 + 256, \eta = 5 \times 10^{-4}$. Data size is about 1GB and iteration for 3×10^5 steps costs about 3700 seconds.

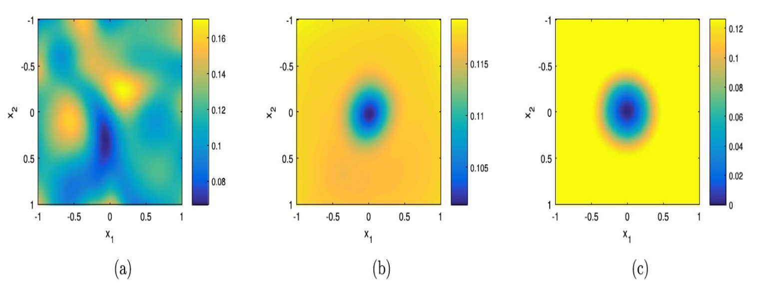
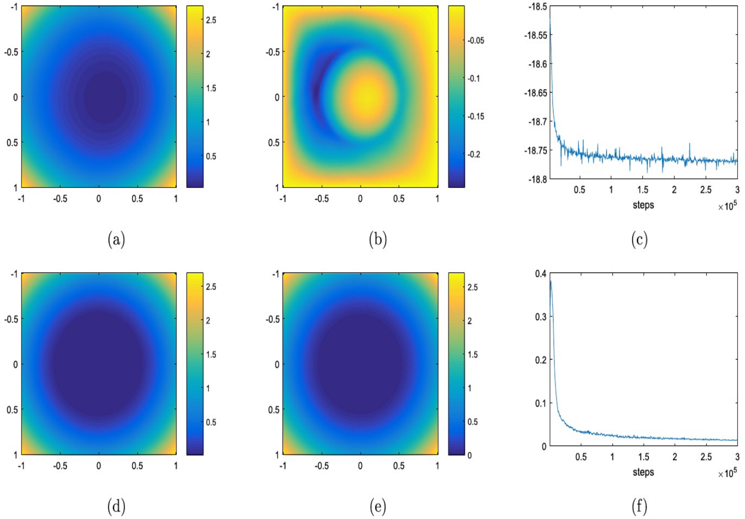


Figure 6: Solution profile at different stages: initial guess, local minimum, global minimum.

Solving interface problems with deep learning

Choose $a_0 = 1, a_1 = 10^3$. The other setting is the same as the first experiment.

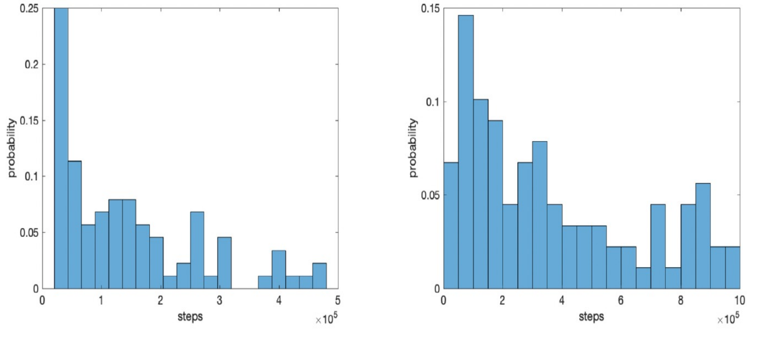


Numerical test2

The Lagrangian functional has instant fluctuations during the optimization process. However, it does not get stuck at a local minimum. The error function is a monotonic decreasing function. Finally, the error is reduced to about 2%.

Numerical test3

For $a_0 > a_1$, we want to investigate the convergence speed when the DNN method gets stuck at a local minimum.



(a) $\alpha_0 = 1000$, $\alpha_1 = 1$.

(b) $\alpha_0 = 10000$, $\alpha_1 = 1$.

Figure 7: Histogram of the number of steps to get out of local minima.

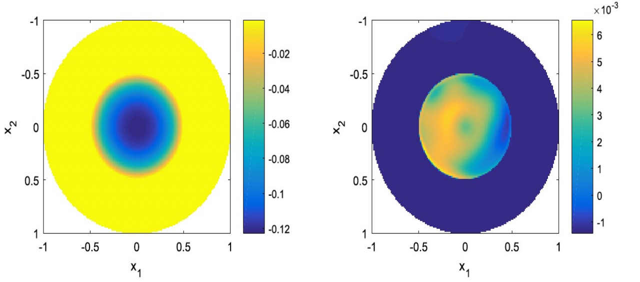
Solving interface problems with deep learning

Numerical test3

We observe that a higher contrast in the coefficient will lead to a slower convergence in the DNN method. When the contrast is higher, the optimization process of the DNN method has a bigger chance to get stuck at a local minimum. We also observe that about 7% of trials failed to converge within the designed steps.

Numerical test4

To show the benefit of the mesh-free nature of the DNN method, we consider following 2D elliptic PDE defined on a closed disk $\Omega = \{x : |x| < 1\}$, where $a_0 = 10^3, a_1 = 1, f(x) = -9r, g(x) = 0$.



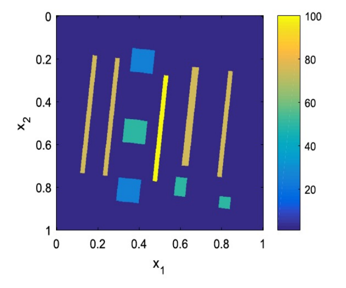
(a) Profile of u (b) Profile of error

DNN method can be used to solve PDEs defined in irregular domains.

Solving interface problems with deep learning

Numerical test5

To study the performance of our method on parameters (batch size, learning rate) that may that determine the accuracy of the DNN method, we consider a 2D interface problem, where $\Omega = [0, 1]^2$, and coefficient $a(x)$ contains high-contrast channels, in order to mimic complicated permeability fields.



Numerical test5

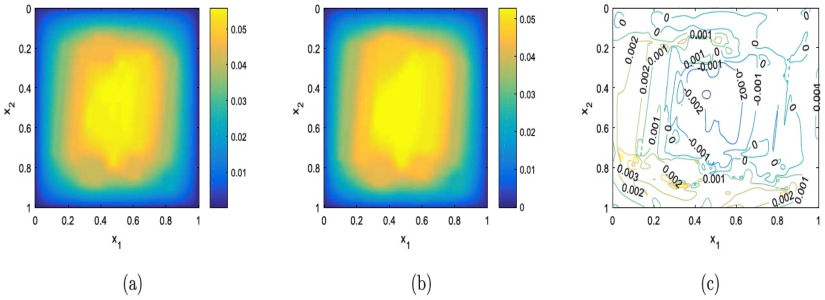


Fig. 10. (a) reference solution; (b) numerical solution; (c) contour plot of the error.

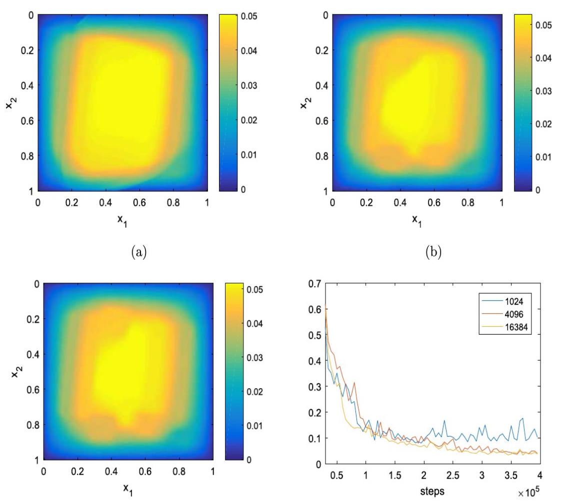
Figure 8: Profile of FEM solution, DNN solution, error

Batch size is $4096 = 3840 + 256, \eta = 2 \times 10^{-3}, \mathrm{L}_2$ relative error is 3%.

Solving interface problems with deep learning

Numerical test5

If we choose different batch size N, this will affect the performance.



Numerical test5

Different learning rate η will also affect the performance.

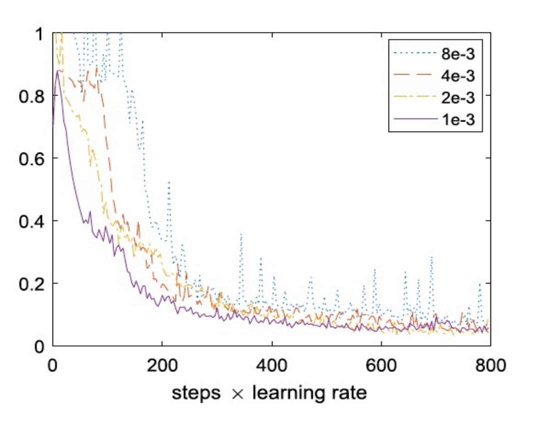


Figure 9: L_2 error with different learning rate η.

Conclusion

We parameterize the PDE solutions by using the ReLU-DNNs and solve the interface problems by searching the minimizer of the associated optimization problems.

- The proposed method is easy to implement and mesh-free since we do not need any special treatment to deal with the interface and boundary.
- We use the DNN method to solve elliptic PDEs with discontinuous and high-contrast coefficients. ReLU-DNN with enough hidden layers and neurons can approximate the solutions well.
- The batch number in the SGD affects the accuracy of the approximation and DNN method is not sensitive to the learning rate.
- The convergence rate for the DNN method is unknown. The issue of local minima and saddle points in the optimization problem is highly nontrivial.

Thank you!