Apart from oceans and rivers, underground flow is also a significant component of the water system on the earth. It is critical to study the subsurface flow because extractions of underground water, gas and oil largely depend on relevant research. Darcy Equation describes the relationship between the pressure in the field and velocity of the flow, and it has a wide industrial application. In our project, we use two deep learning-based methods, namely Physics Informed Neural Network (PINN) and Deep Ritz Method, to simulate the pressure distribution in the computational domain.

In our surf project, we focus on the following flow problem:

$$k^{-1}\boldsymbol{u} + \nabla p = 0 \quad in\ \Omega,$$
$$\nabla \cdot \boldsymbol{u} = 0 \quad in\ \Omega,$$

subject to the boundary condition $u|_{x=0} = -\frac{1}{2}y^2 + \frac{1}{2}y, u|_{x=1} = u|_{y=0} = u|_{y=1} = 0,$

where k is the heterogeneous permeability, u is the Darcy velocity, p is the pressure and Ω is the computational domain, a 1×1 square.

Physics Informed Neural Network (PINN) was proposed in 2017, which are trained to solve supervised learning tasks while respecting any given law of physics described by general nonlinear partial differential equations. It is claimed to be data-efficient and robust for both linear and nonlinear differential equations.
In this project, PINN is taken advantage of to solve for the numerical solution to the partial differential equation when
1. the permeability in Ω is constant $(\mathrm{k} = 1)$;
2. the permeability in Ω fluctuates in fine grids with high-contrast
3. there are cracks in Ω.

The idea behind PINN is that it regards points which lie on the boundaries (with known boundary condition) as the training set and let the physics law embodied in the area serve as the regularization agent.
For instance, in the case where permeability is uniformly equals to 1, the neural network is trained by minimizing the mean square error

$$MSE = MSE_u + MSE_f,$$

MSE_u is the mean squared error for training data lying on the boundary and MSE_f represents the constraint derived from the differential equation, where

$$\mathrm{MSE}_u = \frac{1}{N_u}\sum_{i=1}^{N_u}\left|u\left({x_u}^i, {y_u}^i\right) - u^i\right|^2,$$

$$\mathrm{MSE}_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f\left({x_f}^i, {y_f}^i\right)\right|^2.$$

Here, we define $f \coloneqq k \cdot (\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2})$, $\{{x_u}^i, {y_u}^i, u^i\}_{i=1}^{N_u}$ denote the initial and boundary training data on $u(x, y)$ and $\{{x_f}^i, {y_f}^i\}_{i=1}^{N_f}$ specify the collocations points for $f(x, y)$.

This simple yet powerful construction of neural network allows us to tackle a wide range of problems in computational science and introduces a potentially disruptive technology leading to the development of new data-efficient and physics-informed learning machines, new classes of numerical solvers for partial differential equations, as well as new data-driven approaches for model inversion and systems identification.

Firstly, PINN neural network is reappeared to solve the Darcy equation with uniform permeability k = 1. In all benchmarks considered in this work, the total number of training data N_u is relatively small (a few hundred up to a few thousand points), and we chose to optimize all loss functions using L-BFGS: a quasi-Newton, full-batch gradient-based optimization algorithm. We hope to achieve a satisfying prediction by a sufficiently expressive neural network architecture, namely training all 3021 parameters of a 9-layer deep neural network using the mean squared error loss. Each hidden layer contained 20 neurons and a hyperbolic tangent activation function. The activation function is tanh. We set 50000 iterations to minimize the loss function.
Set N_u = 100 and $N_f = 10000$, we derive the prediction generated from PINN and the resulting prediction error is measured at 1.205828e-02 in the relative L2-norm.
The comparison between the result form PINN and that from the PDE toolbox from MATLAB is as follows.

Since PINN is claimed to be capable of solving difficult partial differential equations, next we increase the complexity of the scenario with the boundary condition stay unchanged. Consider that the factual underflow permeability is heterogeneous, Ω is divided in 16 fine grids, each has an exclusive permeability k. Under different levels of contrast, k is randomly determined in the range of 1-50,1-100,1-250, respectively (denoted by k_{50}, k_{100}, k_{250} in the following context).
If we keep N_u = 100 and $N_f = 10000$, the relative error is 1.431129e-01 for k_{50}, 2.555351e-01 for k_{100}, 5.150456e-01 for k_{250}. The following is the comparison.

The result is not satisfying when the contrast of permeability grows larger, and we note that the prediction is totally unacceptable for k_{250} since it lost the frontal surface.
To improve the prediction, the number of both training points and the collocations points is increased, at the expense of increased training time. Set N_u = 1000 and $N_f = 80000$, the relative error is 1.073231e-01 for k_{50}, 1.664050e-01 for k_{100}, 3.057628e-01for k_{250}. The following is the comparison. Clearly the relative error is reduced significantly. Reduced by 38%, 35% and 41% respectively.
This indicates that by increasing computation burden brutely, the prediction from PINN can be improved, at the cost of training time. +comparison

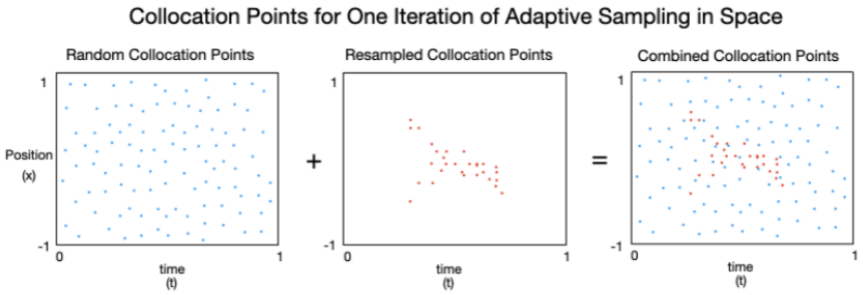
In the case when there are cracks in Ω, the permeability inside the cracks is in high contrast to that outside the cracks. The location of the cracks can be seen as follows.
Firstly, we adapt our previous PINN to fit in the new problem.
When the permeability inside the cracks is 100 and that outside the crack is 1, the prediction from PINN is totally wrong with N_u = 100 and $N_f = 10000$. (comparison)

In order to improve the PINN, we increase the number of training points (1000) and the (10w) collocation points as we previously did. The result is still not satisfying. It succeeded to retain the frontal surface when 10w, but failed to recognize the existence of the two cracks.

Error u: 2.630889e-01

Hence, we have to come up with other ideas to improve the PINN. Since the problem is that PINN cannot provide a good prediction around the crack, it may help to add sufficiently many collocation points around the crack.

Finally, we attempted to adopt an adaptive sampling strategy. Set the permeability inside the cracks is 100 and that outside the crack is 1.



An illustration of adaptive sampling the collocation points over one iteration. Consider the domain x $\in$ [0,1] and y $\in$ [0,1]. The blue points show the set of randomly sampled collocation points using Latin hypercube sampling. Training on these points keeps the solution of the equation accurate across the whole domain. The red points show an example of a set of resampled collocation points sampled after evaluating the f-prediction network for the highest areas of error. These points help to improve the solution accuracy over red-point zones. The combined set of points is the collocation points used to train the network. The network can repeat this process for multiple re-sampling iterations. The blue points will stay the same, but the red points may change to focus on other parts of the domain that are not being learned well.

Each time we choose 1w points by Latin hypercube sampling and add the top 2000 points with the highest f-prediction value to the final set of collocation points. Finally, we take 1000 training points and 5w adaptive sampled points to train the PINN. The result is much better than that from the simple Latin Hypercube Sampling. Comparison is as follows.

Error u: 1.533833e-01

However, the adaptive sampling method still cannot figure out the existence of the cracks. But an interesting phenomenon is discovered in the training process. Since the

L-BFGS optimizer is a gradient-based optimization algorithm, it prioritises to minimize part of the MSE with the larger weight ($MSE = 500 \cdot \left(\frac{\partial^2 u}{\partial x1^2} + \frac{\partial^2 u}{\partial y1^2}\right) + \left(\frac{\partial^2 u}{\partial x2^2} + \frac{\partial^2 u}{\partial y2^2}\right)$) where x1, y1 are collocation points inside the crack and x2, y2 are collocation points outside the crack. Since L-BFGS optimizer prioritises to minimize the error contributed by the cracks, when we add the top 1000 points with largest with the highest f-prediction value, we are not adding the points in the neighbourhood of the cracks as we presumed (in the traditional numerical methods, points near the cracks are considered relatively more difficult to be computed with small error). Actually, we are adding points other than the points near the cracks to the final collocation points.