Physics-aware Neural Network Approach to Inverse Problems in Groundwater Contamination

Hannah Lu, Ziyan Wang
Energy Resources Engineering
Stanford University
{hannahlu, ziyanw}@stanford.edu

Abstract

In this project, we solve the inverse problems in groundwater contamination based on neural networks and automatic differentiation. Given measurements of the contamination concentration, we apply neural networks to approximate the hidden aquifer properties. By comparing the neural network approach to a baseline model using gradient descent, it is demonstrated that the neural networks give much better approximations when the aquifer properties are discontinuous or the measured data is noisy, which are very common in inverse problems. The neural networks are then applied to two realistic scenarios, which shows that the neural networks can accurately recover both layered and mosaic aquifer properties.

1. Introduction

Over 50% of the U.S. population depends on groundwater for drinking water. Groundwater is also one of our most important sources of water for irrigation. Unfortunately, groundwater is susceptible to pollutants. The contamination occurs when man-made products such as gasoline, oil, road salts and chemicals get into the groundwater and cause it to become unsafe and unfit for human use. For science-informed regulation and remedy decision making in contamination, it is crucial to estimate the highly heterogeneous aquifer properties and to identify the contaminant sources (including the location and release history) for accurate prediction of the effects of the contaminants. This kind of problems are defined as inverse problems in groundwater contamination.

Inverse problems are highly challenging because the problem formulation is usually underdetermined (i.e. ill-posed). For a forward problem, e.g. predicting the contaminant distribution in the future given the aquifer properties and the initial/boundary conditions, the solution is uniquely existed and can be obtained through numerical schemes. The inverse problem, however, starts from measured data. We may want to infer some aquifer properties using the contaminate concentration measured at certain locations and/or times. Although the measured data could be huge, the information is usually not enough because the measurements are not sensitive to small-scale features. Another reason that inverse problem is ill-posed is that the measured data is usually noisy, which makes it impossible to perfectly estimate the hidden aquifer properties. In addition, it is very common that the aquifer is highly heterogeneous, thus its properties may be discontinuous at certain locations (e.g. between two layers of the formation). Such discontinuities are extremely difficult to capture since the measured concentration is generally continuous. Due to these reasons, the solution of an inverse problem is usually only the best estimation of the ground truth.

The underdetermined formulation and the data-driven nature make the inverse problem a perfect application for neural networks. Neural networks can provide best estimations without explicitly using the governing equations, thus the underdetermination issue can be avoided. In the project, we build our model based on the framework proposed in [8] and then extend it to more complex problems in realistic groundwater contamination scenarios.

2. Related Works

In recent years, a number of frameworks have been developed for solving forward and inverse problems in differential equations based on neural networks [5, 2, 3, 4, 7, 6]. We also refer to a reveiw paper [1] on geostatistical methods for inverse problems.

3. Problem Statement

We consider the transport of a contaminant dissolved in the groundwater. There are two processes controlling the contaminant migration: advection and dispersion. Advection describes the movement of the solute with the bulk fluid, while dispersion describes the spread of a solute

plume involving the mixing of the solute with native ground water. Overall, the contaminant transport is described by the following advection-dispersion equation:

$$\frac{\partial c}{\partial t} = \nabla \cdot (\mathbf{D}\nabla c) - \mathbf{v} \cdot \nabla c + f \tag{1}$$

where $c(\mathbf{x}, t)$ is the concentration of the contaminant; t is time; $\mathbf{x}$ is spatial coordinate; $\mathbf{v}(\mathbf{x})$ is the groundwater velocity. $\mathbf{D}(\mathbf{x})$ is the dispersion coefficient, which is the key aquifer property that controls solute migration, and $f(\mathbf{x}, t)$ is a source term representing the generation or consumption of the solute. In addition, the governing equation Eq. 1 is subjected to the following initial and boundary conditions:

$$\begin{aligned} c(\mathbf{x}, t = 0) &= c_0(\mathbf{x}), \\ c(\mathbf{x} \in \partial\Omega, t) &= c_b(\mathbf{x} \in \partial\Omega, t), \end{aligned} \tag{2}$$

where Ω is the domain of consideration.

The inverse problem is formalized as follows: we want to infer $\mathbf{D}(\mathbf{x})$, $\mathbf{v}(\mathbf{x})$ and $f(\mathbf{x}, t)$ based on $c(\mathbf{x}, t)$. Generally, it is impossible to make accurate predictions if all these variables are unknown. In this project we focus on approximate $\mathbf{D}(\mathbf{x})$, assuming the other two variables $\mathbf{v}$ and f are given, e.g. by other measurements or estimations.

4. Data Generation

The required data $c(\mathbf{x}, t)$ and $\mathbf{D}(\mathbf{x})$ are generated by the following steps:

- Step 1: Select representative dispersion fields $\mathbf{D}(\mathbf{x})$ generated from the Stanford Geo-statistical Modeling Software (SGeMS) with laterally correlated heterogeneity;
- Step 2: Use forward PDE solver to obtain contaminant concentration map $c(\mathbf{x}, t)$;
- Step 3: Add noise in the concentration map to mimic the real measurement data $\bar{c}(\mathbf{x}, t)$.

A two dimensional layered aquifer generated with laterally correlated $\mathbf{D}(\mathbf{x})$ and the corresponding concentration measurements $c(\mathbf{x}, t)$ are presented in Figure 1.

5. Methods

The inverse problem is solved by conventional gradient descent optimization and neural network optimization. First, we discretize the considered domain into $N = N_x \times N_y$ points, with each point represented by x_{ij}, $i = 1, 2, ..., N_x$ and $j = 1, 2, ..., N_y$. For simplicity, we assume the discretization is uniform and each grid is squire:

$$\begin{aligned} x_{i+1,j} - x_{ij} &= h, \\ x_{i,j+1} - x_{ij} &= h, \end{aligned} \tag{3}$$

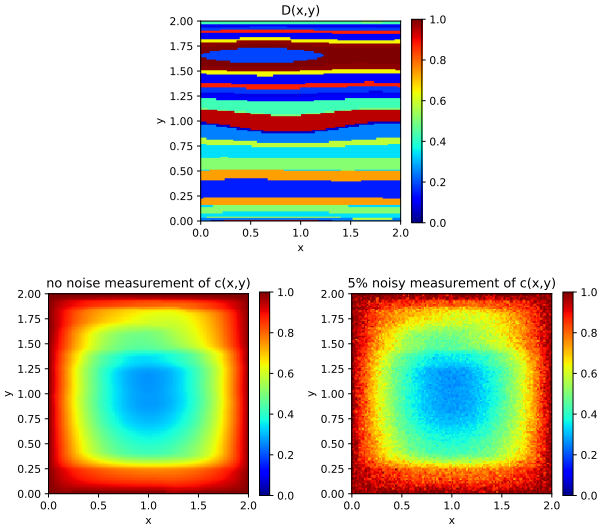

Figure 1: A layered aquifer generated by SGeMS (top) and the contaminated concentration obtained by forward simulations (bottom left). Noise is then added to mimic the real measurements (bottom right).

5.1. Gradient descent optimization

In the gradient descent approach, we want to find D_{ij} that minimize $J(D)$:

$$\begin{aligned} J(D) = \sum_{i=2}^{N_x-1} \sum_{j=2}^{N_y-1} \Bigg[& \frac{\mathbf{c}_{ij}^1 - \mathbf{c}_{ij}^0}{\Delta t} + u_{ij} \frac{\mathbf{c}_{i+1,j}^0 - \mathbf{c}_{i-1,j}^0}{2h} \\ & + v_{ij} \frac{\mathbf{c}_{i,j+1}^0 - \mathbf{c}_{i,j-1}^0}{2h} - \frac{1}{2}(f_i^1 + f_i^0) \\ & - \frac{D_{i+1/2,j}(\mathbf{c}_{i+1,j}^0 - \mathbf{c}_{ij}^0) - D_{i-1/2,j}(\mathbf{c}_{ij}^0 - \mathbf{c}_{i-1,j}^0)}{h^2} \\ & - \frac{D_{i,j+1/2}(\mathbf{c}_{i,j+1}^0 - \mathbf{c}_{ij}^0) - D_{i,j-1/2}(\mathbf{c}_{ij}^0 - \mathbf{c}_{i,j-1}^0)}{h^2} \Bigg]^2, \end{aligned}$$

$$\begin{aligned} \text{with } D_{i+1/2,j} &= \frac{1}{2}(D_{ij} + D_{i+1,j}), \\ D_{i,j+1/2} &= \frac{1}{2}(D_{ij} + D_{i,j+1}). \end{aligned} \tag{4}$$

The expression of $J(D)$ is inspired by the finite different method for forward problems. To minimize $J(D)$, the gradient descent optimization is employed:

$$D_{ij} := D_{ij} - \alpha \frac{\partial J(D)}{\partial D_{ij}}, \tag{5}$$

where α is the learning rate. The conventional gradient descent approach will serve as a baseline model for the neural network optimization.

5.2. Neural network optimization

In the neural network approach, we first rearrange the spatial coordinate a N dimensional vector:

$$\mathbf{x} = [x_{11}, x_{12}, ..., x_{1N_y}, x_{21}, ..., x_{N_x N_y}]^\top. \tag{6}$$

The vector $\mathbf{x}$ serves as the input variable for our neural network. The output is the hypothesis $h_\theta(\mathbf{x})$, which is also a N dimensional vector to approximate $D(\mathbf{x})$. Thus, the neural network has N input nodes and N output nodes, and we employ three hidden layers with d neurons in each. A schematic demonstration of the neural network we adopted is presented in Figure 2.

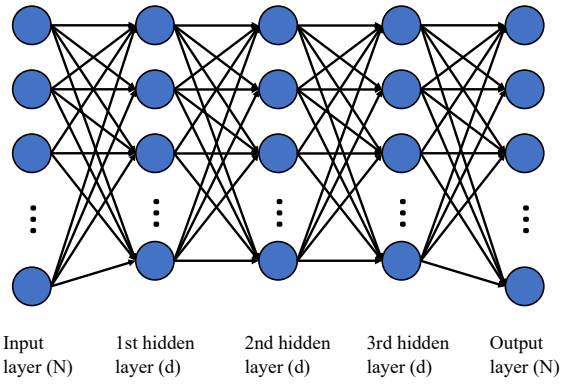

Figure 2: A schematic representation of the neural network structure.

The loss function is similar to Eq. 4, which is defined as:

$$\begin{aligned}\text{loss}(\theta) = \sum_{i=2}^{N_x-1}\sum_{j=2}^{N_y-1}\Bigg[&\frac{\mathbf{c}^1_{ij}-\mathbf{c}^0_{ij}}{\Delta t} + u_{ij}\frac{\mathbf{c}^0_{i+1,j}-\mathbf{c}^0_{i-1,j}}{2h} \\ &+ v_{ij}\frac{\mathbf{c}^0_{i,j+1}-\mathbf{c}^0_{i,j-1}}{2h} - \frac{1}{2}(f^1_i + f^0_i) \\ &- \frac{h_\theta(x_{i+1/2,j})(\mathbf{c}^0_{i+1,j}-\mathbf{c}^0_{ij}) - h_\theta(x_{i-1/2,j})(\mathbf{c}^0_{ij}-\mathbf{c}^0_{i-1,j})}{h^2} \\ &- \frac{h_\theta(x_{i,j+1/2})(\mathbf{c}^0_{i,j+1}-\mathbf{c}^0_{ij}) - h_\theta(x_{i,j-1/2})(\mathbf{c}^0_{i,j}-\mathbf{c}^0_{i,j-1})}{h^2}\Bigg]^2.\end{aligned} \tag{7}$$

The neural network is constructed as follows. We adopt $tanh$ as the activation function and the parameters in the neural network is optimized by backpropagation. We let the number of neurons in each hidden layer $d = 40$ ($d = 20$ in one dimensional simulations), the number of hidden layers $n_l = 3$, and the size of input and output layer $N = 128 \times 128$ ($N = 1000$ in one dimensional simulations). The neural network is optimized by the following steps:

- Step 1: Randomly initialize neural network parameters;
- Step 2: Compute the hypothesis $h_\theta(\mathbf{x})$;
- Step 3: Compute the loss function;
- Step 4: If the loss is larger than the tolerance, update the neural network parameters by gradient decent following the backpropagation rules;
- Step 5: Go back to step 2 until the loss is smaller than the tolerance.

It should be emphasized that in the optimization, we take an observed contaminant concentration $\mathbf{c} = [c_1, \cdots, c_N]^\top$ to compute the loss function Eq. 7. However, unlike most cases, the neural network input is not the observation $\mathbf{c}$ but $\mathbf{x}$. This is because our inverse problem is an optimization problem, rather than a generalization problem. We do not intend to compute a new aquifer property given a new concentration observation (i.e. generalization), as the aquifer property does not change with the observations. We want to utilize all the concentration data and give the best estimation of the aquifer property (i.e. optimization).

6. Experiments and Results

6.1. One dimensional continuous aquifer

First, we consider a basic example of stationary diffusion described by:

$$\frac{\partial}{\partial x}\left(D(x)\frac{\partial c}{\partial x}\right) + f(x) = 0, \tag{8}$$

where $D(x)$ and $f(x)$ are continuous and have the following analytical expressions:

$$\begin{aligned}&D(x) = 1 + e^{-(x-0.5)^2}, \\ &f(x) = [\pi^2 sin(\pi x) + 2\pi(x-0.5)cos(\pi x)]e^{-(x-0.5)^2}.\end{aligned} \tag{9}$$

It can be easily verified that the following analytical solution satisfies Eq. 8:

$$c(x) = sin(\pi x). \tag{10}$$

We adopt both the gradient descent and neural network approaches to infer the aquifer property $D(x)$ based on $c(x)$. As shown in Figure 3, both methods can accurately capture the hidden $D(x)$. However, if the concentration data is noisy (Figure 4), the gradient descent optimization fails because it requires extremely small learning rate to prevent the algorithm from blowing up, which makes the approach computationally prohibitive. The neural network optimization can still accurately infer $D(x)$ even with noisy data.

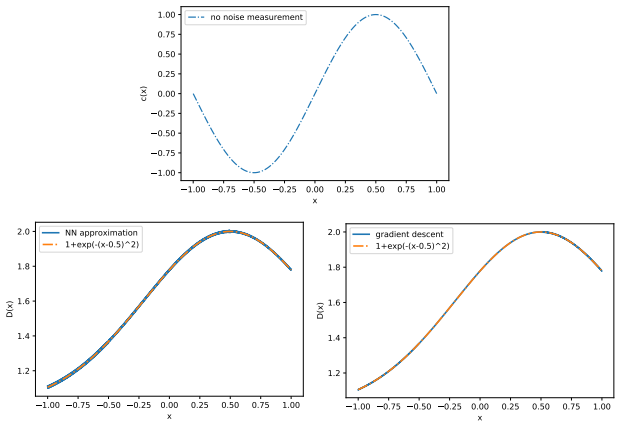

Figure 3: One dimensional continuous aquifer without noise: the concentration data (top) and the aquifer property $D(x)$ inferred by neural network optimization (bottom left) and gradient descent optimization (bottom right), compared with the ground truth.

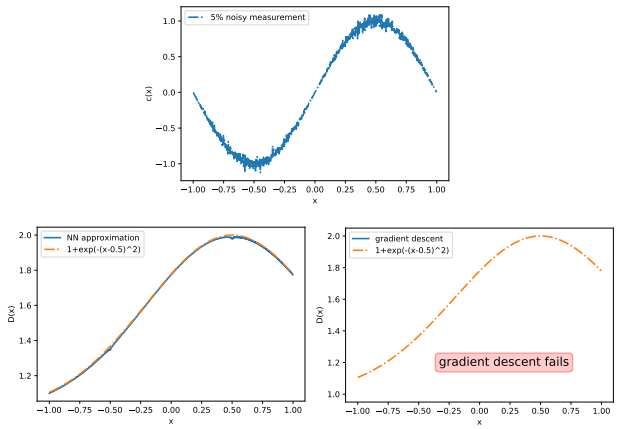

Figure 4: One dimensional continuous aquifer with noise: the concentration data (top) and the aquifer property $D(x)$ inferred by neural network optimization (bottom left), compared with the ground truth. The gradient descent optimization fails because it requires computationally prohibitive small learning rate to prevent blowing up (bottom right).

6.2. One dimensional discontinuous aquifer

Next we consider an example in which the aquifer property $D(x)$ is discontinuous, as demonstrated in Figure 5. We first solve Eq. 8 with a step function $D(x)$, then solve the inverse problem using the solution $c(x)$. Although $D(x)$ is discontinuous, the solution $c(x)$ is still continuous, which makes the inverse problem much more difficult. However, the neural network approach can still precisely recover the hidden $D(x)$, while the gradient descent approach converges to a physically incorrect solution.

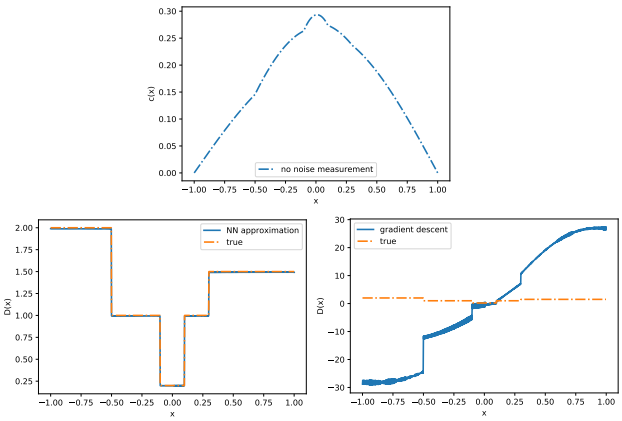

Figure 5: One dimensional discontinuous aquifer: the concentration data (top) and the aquifer property $D(x)$ inferred by neural network optimization (bottom left) and gradient descent optimization (bottom right), compared with the ground truth.

6.3. Two dimensional layered aquifer

We then consider a two dimensional problem, in which the aquifer has layered structure. Like before, our objective is to infer the heterogeneous dispersion field $D(x, y)$ based on measured concentration field $c(x, y)$. Since the gradient descent approach has been proven unreliable for aquifer with discontinuity, we only employ the neural network approach in the two dimensional problems.

Figure 6 presents the concentration measurements, the true dispersion fields, the dispersion fields inferred by neural networks, and the errors between the inferred and the true dispersion fields. Figures on the left represent the case without noise, while figures on the right represent the case with noisy concentration measurements. For both cases, the neural network approach can recover the hidden layered dispersion field with very good accuracy. We emphasize that this is not a trivial task because the concentration field shows no layered features, however the layered aquifer can still be precisely captured through the neural network optimization.

6.4. Two dimensional aquifer with mosaic $D(x, y)$

Finally we try to infer a mosaic dispersion field $D(x, y)$ based on the measured concentration field $c(x, y)$. This is another type of heterogeneity that is very common for rocks with several compositions. Figure 7 shows the measured concentration fields, the true dispersion fields, the dispersion fields inferred by neural networks, and the errors. Like before, figures on the left represent the case without noise,

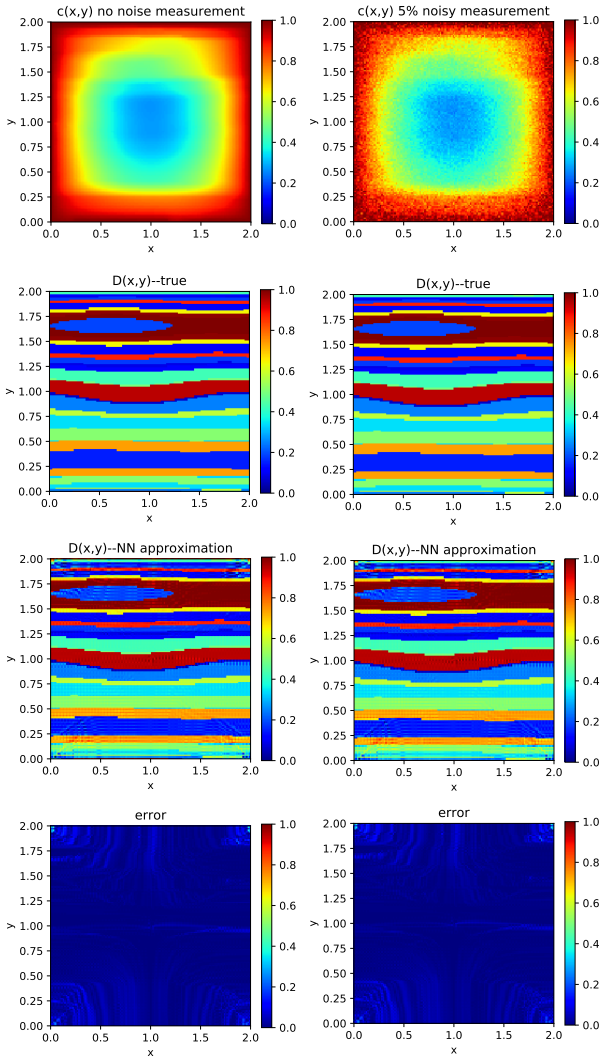

Figure 6: Two dimensional layered aquifer: the concentration fields, the true dispersion fields, the dispersion fields inferred by neural networks, and the errors (from top to bottom). Figures on the left represent the case without noise and figures on the right represent the noisy case.

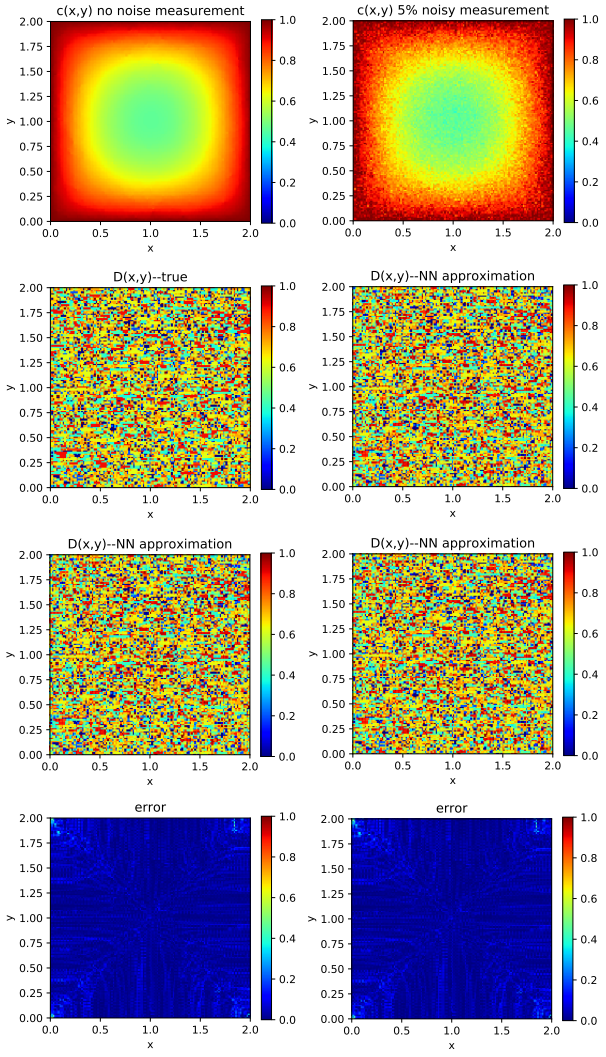

Figure 7: Two dimensional aquifer with mosaic $D(x, y)$: the concentration fields, the true dispersion fields, the dispersion fields inferred by neural networks, and the errors (from top to bottom). Figures on the left represent the case without noise and figures on the right represent the noisy case.

while figures on the right represent the case with noisy concentration measurements. Once again, the neural network approach gives pretty good approximations of the mosaic $D(x, y)$.

7. Conclusion and Future Work

We employed a physics-aware framework for inverse problems based on neural networks and automatic differentiation. Compared to conventional optimization tools, this framework is more stable and powerful, especially for noisy measurements and heterogeneous targets. Our numerical examples show the robustness of this framework in heterogeneous aquifer property identification, even with noisy contaminate concentration measurements.

Future work: 1) We want to model the complete time-dependent advection-dispersion problem and use this framework to infer other aquifer properties as well; 2) Comparison between our NN approximation and the popular adjoint method from the geophysics community will be included; 3) Sensitivity analysis and error estimation will be conducted to quantify the uncertainty under such framework in practical problems.

References

[1] P. K. Kitanidis. Recent advances in geostatistical inference on hydrogeological variables. *Reviews of Geophysics*, 33(S2):1103–1109, 1995.

[2] S. Mo, N. Zabaras, X. Shi, and J. Wu. Deep autoregressive neural networks for high-dimensional inverse problems in groundwater contaminant source identification. *Water Resources Research*, 55(5):3856–3881, 2019.

[3] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[4] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[5] M. Raissi, 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. *Journal of Computational Physics*, 378:686–707, 2019.

[6] M. Tang, Y. Liu, and L. J. Durlofsky. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *arXiv preprint arXiv:1908.05823*, 2019.

[7] G. Wen, M. Tang, and S. M. Benson. Multiphase flow prediction with deep neural networks. *arXiv preprint arXiv:1910.09657*, 2019.

[8] K. Xu and E. Darve. The neural network approach to inverse problems in differential equations. *arXiv preprint arXiv:1901.07758*, 2019.

Contributions

- Hannah Lu: Literature review, model construction, data generation, neural network coding, poster writing, final report revision
- Ziyan Wang: Literature review, model construction, baseline model coding, final report writing, poster revision

Link to Project Code

See here.