Project #2

assigned: Tuesday, Feb 25, 2025
due: Tuesday, March 18, 2025, 17:00
Please submit your project online through **Moodle**!

---

collaboration policy

You are encouraged to discuss the homework problems and solution strategies with your classmates, but you must find the solutions and write down your answers to all problems by yourself.

Problem 1: Solving PDE: Plate with a hole (45 points).

The objective of this project is to write your own physics informed neural network (PINN) solver to **solve** linear elastic boundary value problems in 2D (under plane stress (2D stress) assumptions). Let us solve this boundary problem numerically with our PINN code. By symmetry, it is sufficient to simulate only one quadrant of the problem with uniform traction applied on two edges and sliding essential boundary conditions applied on the other two edges, as shown in Figures 1-2. You are provided with a Matlab script `Plate_holem` , to generate the Collocation points to `Plate.mat` as the input to the PINN, as well as to provide you with a reference truth solution of the problem from the Finite Element method. Young's modulus, Poisson's ratio and plate thickness are $E = 10$ N/m^2, $\nu = 0.3$ and $t = 1$ m, respectively. Use $\sigma_1 = 0.1$ N/m^2 and $\sigma_2 = 0$ N/m^2.

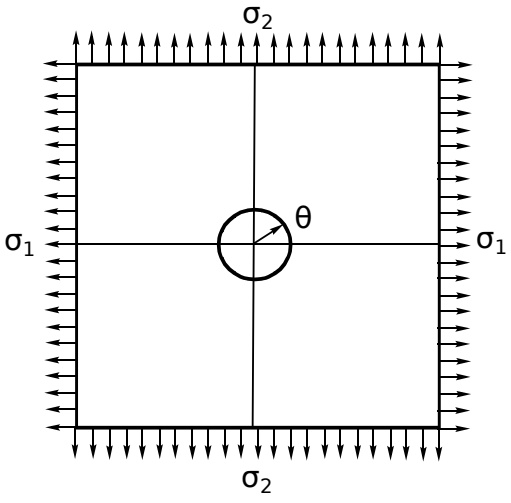

Figure 1: Plate with circular hole

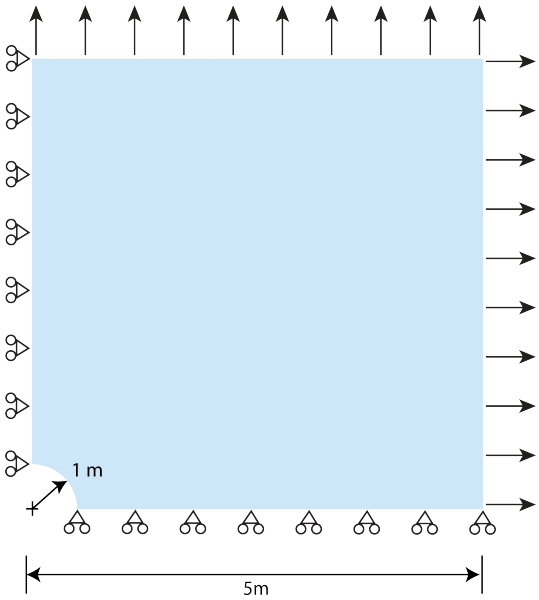

Figure 2: Numerical model with dimensions

(a) Write down the governing partial differential equation for this problem and specify the boundary conditions.

(b) Define the loss function for the PINN.

(c) Design, build and train a Physics Informed Neural Network for this PDE, based on the skeleton code "PINN_skeleton.py". Plot the training error against training epochs

(d) Explain your hyper-parameter tunning strategy, and comment on the performance of your PINN.

(e) To improve the performance of PINN, you are provided with measurement data observing the displacement field $u(x)$ from 50 points in the plane. Use this information to enhance training of the PINN, and plot the training error against training epochs.

(f) Visualise your result by plotting the tensile stress field $\sigma_{11}(x)$. Comment on the performance of trained PINN.

Hint: For PINN to work, one need huge training epochs (order of 50-100k). You might also find PINN trained with pure PDE loss gives an unsatisfying result - and this is totally expected. You will see that adding measurement data to it really helps PINN to converge to the truth solution (which is given in the matlab code).

Problem 2: Learning PDE: 2D Darcy flow (55 points).

Let us consider **learning** the solution operator of 2D Darcy flow problem. Consider the following 2-D Darcy Flow equation on the unit box

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x) \quad x \in (0,1)^2,$$

with Dirichlet boundary condition

$$u(x) = 0 \quad x \in \partial(0,1)^2.$$

where $a(x)$ is the diffusion coefficient generated from a random Fourier field while $f(x)$ is the forcing function which we keep as a constant throughout the domain. Our aim is to learn the operator mapping the diffusion coefficient to the solution defined by

$$\psi^\dagger : a \mapsto u,$$

using the operator learning methods we discussed in the class.

You are provided a training data set Darcy_2D_data_train.mat, and a test data set Darcy_2D_data_test.mat, which are generated from the MATLAB script "Darcy_2D.m".

(a) Design, build and train a convolution neural network (CNN) to learn the solution operator. Plot the train and test losses versus the training epochs as well as contour plots for truth and approximate solutions of the solution u for a single sample from the test data, and explain your choice of the CNN architecture, optimizer, and learning rate. You may use the provided python Skeleton code "Darcy_CNN_skeleton.py".

(b) Design, build and train a Fourier Neural Operator (FNO) architecture to learn the solution operator. Plot the train and test losses versus the training epochs as well as contour plots for truth and approximate solutions of the solution u for a single sample from the test data, and explain your choice of the FNO architecture, optimizer, and learning rate. You may use the provided python Skeleton code "Darcy_FNO_skeleton.py".

Submission requirements:

Your submission should include

- the **Python** script problems 1 and 2: `PINN.py, PINN_data.py, Darcy_CNN.py` , `Darcy_FNO.py`
- a **PDF** file required to answer parts of problems 1 and 2.

Please note that commenting your code to make it clearly understandable will help the grading process.

Package all of the above in a **single** `.zip` **file** and submit it using the *Assignments* tool in Moodle.