Composite Design Optimization via Neural Operator

Hanfeng Zhai[1]
Cornell University

Abstract: This is a short report containing my effort on **data generation** using COMSOL Multiphysics®, learning finite element modeling with **FEniCS**, and playing with the **toy models in FNO**. I generated 100 FEA simulations of composite materials stress mapping data using a linear elasticity model. I successfully trained FNO w/ their 2D prototype model on learning the Darcy flow problem and generated prediction results with the model, which is the most transferable to my linear elasticity stress mapping problem. This midterm is primarily focusing on *Forward Modeling*. The successful training of the operator learning model will eventually lead to the design optimization — an *Inverse Problem*. Based on my discussion w/ Prof. Earls and taking his opinion on operator regression models, I decided to switch to FNO for the surrogate modeling instead of my previous plan on DeepONet. Some potential future plans (based on my proposal) are also drawn accordingly from my current progress.

Data Generation

To begin with, I start with what I'm familiar with and use COMSOL to run the FEA for generating some training data. I defined the grid distribution of "fiber" inserted in the PDMS matrix as the cross-section of fiber-reinforced composite. The fiber can be randomly generated on a 5×5 grid on the basis of the preset materials — as I already elaborated in the proposal. I wrote a python program to generate 10 fibers among the 25 options (C_{25}^{10}) as a toy case. I generated a total of 50 different geometries. By rotating the composite in the horizontal direction[2] one can obtain 100 different geometries — as a data augmentation approach. I randomly generate 50 cases for running the linear elastic FE simulations.

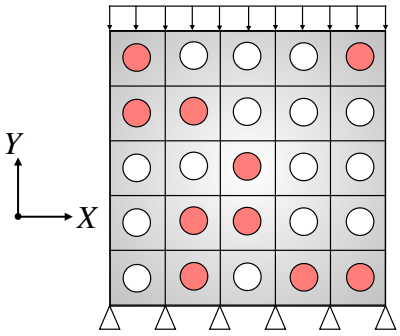

Figure 1. Basic boundary condition setup for the data generation cases. Note that the red circles denote the positions with filled fiber materials (randomly generated). There is a total of 25 possible positions in the material basis grid.

Figure 1 shows the basic setup of the geometric setup for the numerical simulation. The material basis in the 2D space undergoes a uniaxial compression loading where the bottom is fixed and the top side takes the load. By running the 50 simulations in COMSOL, one can obtain the meshing points data corresponding to the von Mises' stress as the raw training set. The postprocessing of the data is conducted according to the following section.

[1]Email: hz253@cornell.edu
[2]based on my BCs

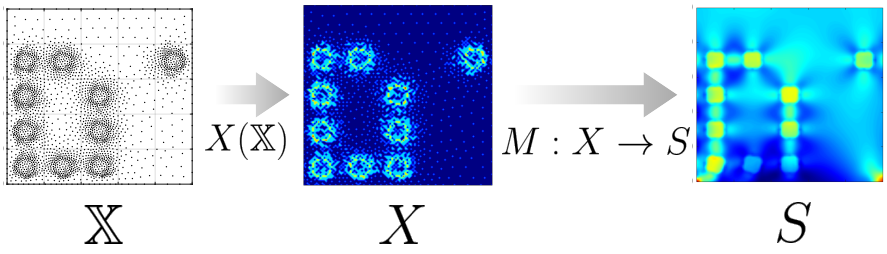

FIGURE 2. A schematic illustration for the *forward problem*. An input mesh is scanned as a density distribution on the material basis grid. The input mesh is represented in a 2D array and then transformed into an "image" via the scanning process. The forward problem is defined as the map between the mesh density scan to the corresponding von Mises' stress field.

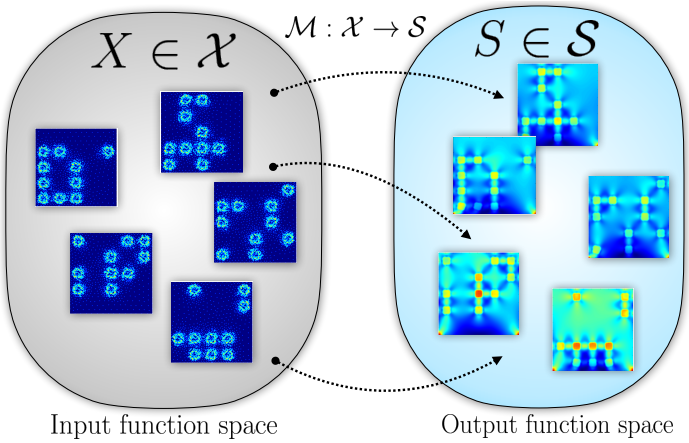

FIGURE 3. The schematic representation for the function space mapping. $X \in \mathcal{X}$ represents the input meshing density scan that lies in the meshing group containing different possible materials structural combinations. Their corresponding stress outputs lie in the output function space. The operator takes the form: $\mathcal{M} : \mathcal{X} \to \mathcal{S}$.

NUMERICAL MATHEMATICAL EVALUATION

The COMSOL generates the FEM calculated data in 3 dimensions: $\texttt{data} = [x, y, \sigma]$. Of course, one can simply create a DNN that maps the two inputs to the one output for learning: $\mathsf{DNN} : [x, y] \in \mathbb{R}^2 \to \sigma \in \mathbb{R}$. This would usually be the case for a standard ML practice toy case. Here, we first transform the stress data by mapping the σ values on the $[x, y]$ meshes using the MATLAB `griddata` function to transform it input spatially distributed data matrices (or images in a simpler way). The stress field is then mapped in the 2D space with size 101×101. Here, we denote this stress field as S in Figure 2. With the given meshing spatial distribution, one can scan its density and project the density distribution in the material basis as an image representation[3]. The spatial distribution can then be represented also in the form of 101×101 matrices. We now assume the initial 2-dimensional matrix of meshing coordinates as $\mathbb{X}$ and the projected density distribution as X, then this transformation takes the form: $X = X(\mathbb{X})$. The forward problem of solving for the stress based on linear elasticity then can be stated as $M : X \to S$. This process is illustrated in Figure 2.

Since we have 50 von Mises' stress field data, if assume $\mathbb{Z}$ to be the discretized space representing the memory, the total output function space $\in \mathbb{Z}^{50\times101\times101}$, their corresponding input function space then possesses the same size. This mapping then can be denoted as: $\mathcal{M} : \mathcal{X} \to \mathcal{S}$. This process is illustrated in Figure 3.

[3]I personally don't favor the nomination of "image" but use it here for clearer explanation

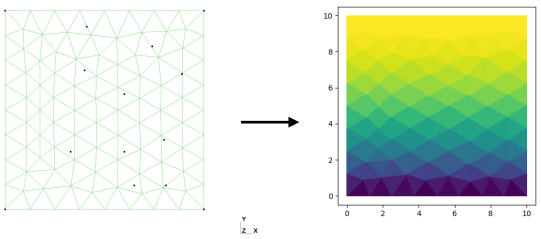

FIGURE 4. A simple schematic for my exercise using FEniCS for FE simulations.

FEniCS Modeling

I have always wanted to learn to do FEA with FEniCS, due to my personal preference for open-source software and my previous experiences with LAMMPS. Since even with the COMSOL-generated data I can finish this project I still decided to borrow this chance to play around with FEniCS.

I successfully install FEniCS and ran a few simple cases using the linear elasticity solver. Herein I showed a simple FEA implementation using FEniCS — I still couldn't figure out how to generate multi-material models and convert the refined mesh to `.xml` file[4]. The simulation begins with the geometry generated by Gmesh and is solved by the python `fenics` module directly. The BCs follow our preset problem formulation. The solving process simply follows Figure 4.[5]

Fourier Neural Operator

I trained FNO on their Darcy flow problem, considering my 2D linear elasticity problem is just a static problem, I can simply transfer their model to a time-independent `image` → `image` mapping case. Very interestingly[6], someone has already published a paper in a fancy journal by training FNO for mapping[7]. Even if I come up with this idea myself I should expect somebody could have done similar work since it is pretty intuitive for researchers in the mechanics & materials communities to transfer the FNO learning for predicting the properties of composite structures. Also, Gu and Buehler[8] have also done much similar work before FNO was developed — they apply CNN for the `structure` → `properties` mapping (using images) and published a dozen papers based on this idea. But what makes my idea more innovative? (1) The neural operator regression model does the learning on the high-dimensional space — the surrogate model employed is innovative. (2) I plan to couple the operator regression surrogate model for materials design optimization — potentially coupled with Bayesian optimization and/or genetic algorithms. Coupling the state-of-the-art ML model with heuristic design optimization for actual materials design loops is the novelty. Training the composite `structure` → `stress` mapping follows Figure 5.

I played w/ the Darcy flow inference case by benchmarking the training process: and comparing the cases of 50, 100, and 500 epochs. The results are shown in Figure 6. For benchmarking more testing data should be considered in future works. This problem is the easiest to transform to my composite stress mapping problem as the input and output of the FE simulations can also be categorized as `coefficient` → `solution` mapping as a time-independent static problem.

Future Plan

A short overlook of what I've done till now: (1) I **generated training data** for the machine learning model by assuming the map between the input mesh to output von Mises stress field — the mapping was originally calculated using FE solver and we hope to build a surrogate of this solver for on-the-fly evaluation. (2) I learn the basics of FEniCS modeling. (3) I successfully re-implement FNO on my workstation and did some benchmarking on the author's prototype cases.

---

[4] might ask for help later

[5] currently still learning the software and hopefully can generate some data using it

[6] unexpected

[7] Rashid et al., 2022, *iScience*. 10.1016/j.isci.2022.105452

[8] 1st, 2nd, 7th, 9th, 10th, ... top cited papers of Grace Gu and her most recent papers (keywords ML & composite) are mostly using this idea

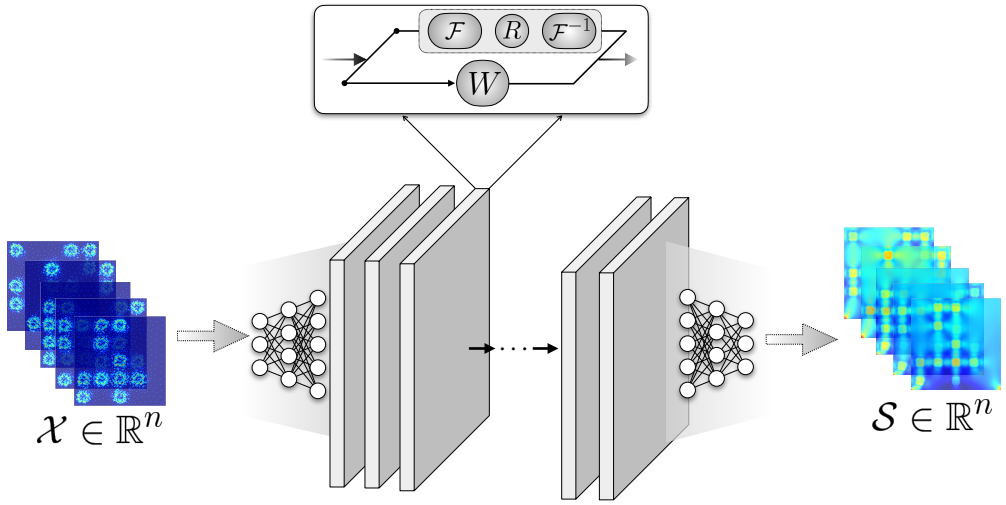

FIGURE 5. The schematic for using FNO to map the input-output function of the properties of the composite material.

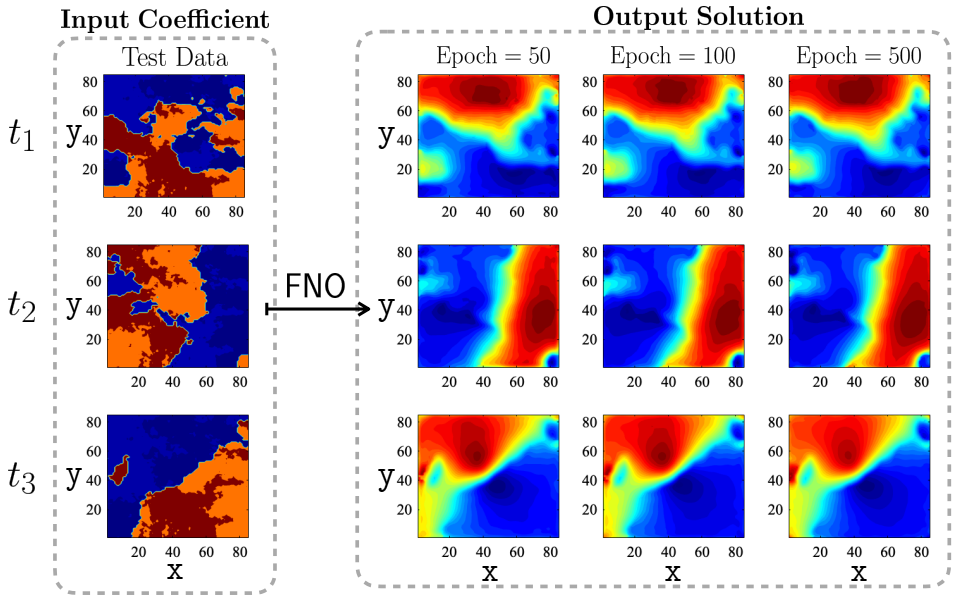

FIGURE 6. Some preliminary results generated from the Darcy flow example provided by the original FNO formulation. I tested training by three different epochs to check the convergence of training.

Based on the current progress, some further work can be expected in the short term by continuing the efforts: It would not be extremely hard to train the FNO on the composite material database. One should expect to see the prediction of stresses according to the meshing density scan from FNO for the next step.

Some more ambitious plans can then be imagined for a longer term: train the FNO as the surrogate model. Now, this model M can evaluate any given X and predict its corresponding S on-the-fly. Now this M can be the fitness function in the genetic algorithm (GA). One can initiate the GA to search for the composite structure with the lowest maximum v/M stress. A group of initial design points can then be generated as the *population*: $[X_1, X_2, ..., X_k]$. The population then mutates, and crossover, ..., for obtaining a good structure with the targeted properties. Similarly, one can also adopt a similar strategy to couple with Bayesian optimization to design targeted structures. The strategy may follow my previous work[9].

[9] arXiv:2209.00055