CS598: Deep Generative & Dyn. Models
Project Proposal: Learning Structured Mappings for Finite Element Problems

Nicolas Nytko, Sean Farhat, Nathanael Assefa, (Alexey Voronin)

March 28, 2023

1 Introduction

Partial Differential Equations (PDEs) provide a mathematical framework for describing and understanding many physical phenomena in science and engineering, from the behavior of fluids and materials to the dynamics of electric and magnetic fields. The solution of PDEs is essential for making predictions, designing and optimizing systems, and developing new technologies. Solving PDEs analytically can be challenging, and in many cases, it is impossible. Therefore, numerical methods are often used to approximate the solutions of PDEs.

To solve numerical PDEs, one must turn the continuous problem into something discrete that is solvable on a computer: this is commonly done by the finite element method (FEM), where a problem domain is approximated by a discrete triangulation (mesh) upon which the PDE is satisfied discretely (think Taylor approximation). Once the PDE has been discretized and approximated, the resulting system of equations is solved numerically such that the solution minimizes the residual on the mesh. What algorithm we use to solve the system, and thus how fast we can solve it, depends on various properties of the mesh and the problem itself.

We want to learn how to map PDE discretizations from unstructured (irregular) mesh domains to structured (grid-like) ones, just like in **??**. To make this happen, we will treat it as a generative process where we wish to generate data from a source distribution: the structured operators. This can be achieved via a diffusion process where we take the structured operator, iteratively add "noise", and then learn the "de-noising" process. Our noise is not the simple Gaussian noise applied to images; instead, we will perturb each of the coordinates and re-triangulate (via an out-of-the-box triangulation algorithm such as Delaunay) to obtain a new discretization. Additionally, we will add a regularization term to keep the solution of the new "noised" operator close to the solution of the original structured one. We will ideally have a network that will take in any unstructured operator (analogous to a fully noised operator), and it will "de-noise" the linear system until it reaches a structured operator that realizes the same solution.

2 Motivation

Multigrid methods [**?**, **?**] are a family of fast iterative solvers for sparse linear systems, such as ones that arise from the numerical discretization of PDEs using the finite element method [**?**]. Much of the speedup in using multigrid solvers is obtained by constructing a hierarchy of smaller, easier-to-solve problems, then interpolating approximate solutions between different levels. There has been a wealth of theory dedicated to creating such a problem hierarchy on regular, well-structured problem domains; however, for more complex problem domains, we often must switch to using algebraic methods, which are often based on heuristics and can vary in efficacy.

We propose to learn a mapping from unstructured problems to structured ones in order to create a more efficient multigrid solver. By transferring solutions to and from a related structured problem, we can reuse existing black-box geometric multigrid algorithms that will converge very well for the overall problem.

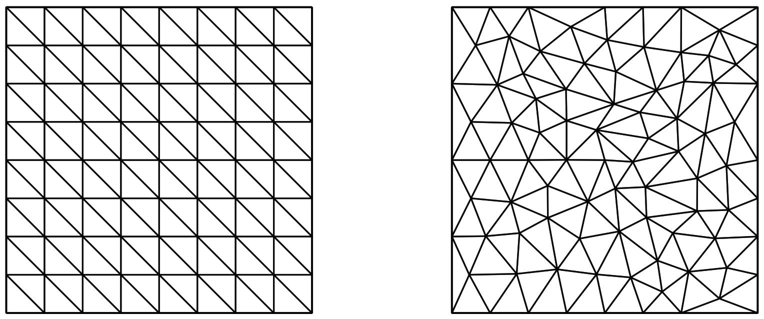

Figure 1: Example of triangular meshes: structured (left) and unstructured (right).

3 Data

We will generate structured problems using triangular finite-element discretizations of the diffusion equation,

$$\nabla \cdot (\kappa(x, y)\nabla u) = 0, \tag{1}$$

where $\kappa(x, y)$ is the diffusivity of the material at some point and $u(x, y)$ is the unknown density we would like to solve for. To add variance in the set of problems we want to *diffuse from*, we will randomly generate $\kappa(x, y)$; for example, by using simplex noise [?] or polynomial interpolation to obtain the function values. To generate the actual matrices themselves, we will use the Firedrake library [?, ?] on **??**, giving us a set of linear systems

$$A^{(i)}u^{(i)} = b^{(i)}. \tag{2}$$

4 Plan of Work

- Methodology for generating the dataset:
  - Use triangular finite-element discretizations of the diffusion equation to generate structured problems.
  - Randomly generate diffusivity values to add variance to the set of problems.
  - Use Firedrake library to generate linear systems from the diffusion equation.
- Learning approach for mapping unstructured problems to structured ones
  - Use a generative process that involves iteratively adding noise to the structured operator and learning the de-noising process.
  - Perturb each coordinate and re-triangulate to obtain a new discretization.
- Loss function:
  - Design a loss function that measures the difference between the predicted structured operator and the actual structured operator.
  - Regularize the loss function to ensure the predicted operator is close to the actual solution.
- Training the model:
  - Train the model on the generated dataset of unstructured operators and corresponding structured operators.

  - Validate the model on a separate test dataset.
- Expected outcome:
  - A deep neural network that can map unstructured PDE discretizations to structured ones.
  - Improved efficiency of multigrid solvers for PDEs, particularly for complex problem domains.

5 Related Works

https://arxiv.org/abs/1104.0261?