A Supervised Learning Approach to Predicting Multigrid Convergence

Nicolas Nytko

Matthew West, Luke Olson, Scott MacLachlan

March 17, 2021

Overview

Poisson Problem

- Look at the 1D variable coefficients case w/ homogeneous Dirichlet conditions

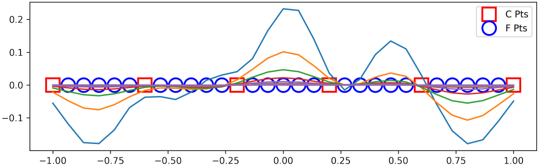
$$-\nabla \cdot (k(\mathbf{x}) \nabla \mathbf{y}) = f$$

$$\Omega = [-1, 1] \quad \partial\Omega = 0$$

- Discretized on $N = 31$ internal points using finite differences, $k(\mathbf{x})$ is discretized on midpoints to preserve symmetry.
- For arbitrary C/F splitting, can we predict convergence rate and optimal relaxation weight?

Training Dataset

- For "traditional" machine learning we need a dataset.
- Idea: Run a *whole lot* of multigrid iterations.
- Run multigrid iterations and record convergence rate and relaxation weight for randomly generated C/F splittings and problem setups.



Dataset Generation

- Start from "reference" splittings, which are evenly spaced coarse points on a grid.
- Randomly perturb each reference in several trials, according to

$$p = \{0.01 \quad 0.05 \quad 0.1 \quad 0.25 \quad 0.5 \quad 0.75\}.$$

- Generate variable coefficients such that

$$k(\mathbf{x}) = \begin{cases} \alpha \\ \text{rand}()\,(\alpha + 1) \\ \alpha \cos(\pi x \beta) + \gamma \\ \left| \sum_{i=1}^{5} \alpha_i x^i \right| + 0.01 \end{cases}.$$

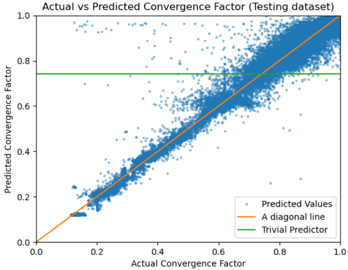
Convolutional Network

- Take the C/F splittings, run in multigrid solver to find convergence rate and relaxation weight that maximizes the former.
- Use the data to train a *1D convolutional network* that predicts convergence, Jacobi relaxation.

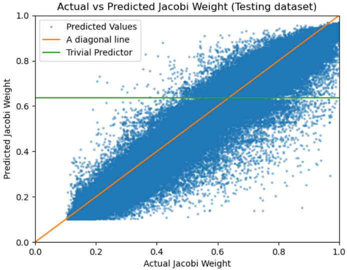
Model	Dataset	Value
Jacobi Weight	Training	1.8331×10^{-3}
Jacobi Weight	Testing	1.8396×10^{-3}
Convergence Factor	Training	1.4839×10^{-3}
Convergence Factor	Testing	1.5171×10^{-3}

Table: Mean squared error (MSE) between predicted and true Jacobi weight, convergence factor.

CNN Performance



(a) Testing predictions



(b) Training predictions

Figure: (a) Predicted convergence values vs. true convergence values, (b) Predicted relaxation weights vs true relaxation weights. Values closer to the diagonal represent more accurate predictions.

What we learned: Poisson is too easy!

Let's try learning a more difficult problem.

Convection-Diffusion Problem

Try out a specific convection-diffusion problem,

$$\mathbf{w} \cdot \nabla u - k\nabla^2 u = f,$$

on 2D square, $\Omega = [-1, 1]^2$, discretized as quadrilateral finite elements. Use $k = 0.1$, $\mathbf{w} = \begin{bmatrix} 2y(1 - x^2) & 2x(1 - y^2) \end{bmatrix}$.

Dataset Generation, Convection-Diffusion

- Discretize on a 25×25 structured grid.
- Start from "reference" splittings, all fine, all coarse, AMG output, etc.
- Randomly perturb each reference in several trials, according to

$$p = \{0.01 \quad 0.05 \quad 0.1 \quad 0.25 \quad 0.5 \quad 0.75\}.$$

- Don't generate coefficient values for now.
- Take output and run through 50 iteration multigrid solver to find convergence rate.

Convection-Diffusion Convolution

- 2D structured grid $\Rightarrow$ train 2D convolutional network to predict convergence.

- Classical convolution techniques work okay on structured, grid-like inputs.
- Very restrictive in terms of mesh data we can use for FEM solvers.
- Take a look at some network architectures that allow for unstructured data: introduce *graph-nets*.