## 1 Background

The problem being solved is the Poisson problem in one dimension:

$$-\nabla^2 \boldsymbol{u} = f$$

$$f(\boldsymbol{x}) = \boldsymbol{x}$$

Defined on the following domain with Dirichlet boundary conditions:

$$\Omega = [-1, 1]$$

$$\partial\Omega = 0$$

This is discretized using finite differences on a grid of $N = 31$ equispaced points and results in the following system:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{x}$$

where

$$\boldsymbol{A} = \frac{1}{h^2}\text{tridiag}\left([-1, 2, -1]\right)$$

$$h = \frac{1}{N+1}$$

The above linear system is solved using a two-level V cycle multigrid method using the following steps:

1. Generate an initial random guess, $\boldsymbol{u}_0$.
2. Pre-smooth the guess using $\nu = 5$ iterations of Jacobi using weight $\omega$.
3. Restrict the grid and residual using the operator $\boldsymbol{R} = \boldsymbol{P}^T$. The given interpolation operator $\boldsymbol{P}$ is constructed as the *ideal interpolation operator*.
4. Perform a linear solve on the restricted residual to obtain the coarse solution.
5. Interpolate the solution to the fine grid using operator $\boldsymbol{P}$.
6. Post-smooth using 5 iterations of Jacobi, with same weight $\omega$.

The Jacobi smoothing weight that is used, $\omega$, is the optimal weight that minimizes the overall factor of convergence for the method:

$$\omega = \arg \min_{\omega \in (0,1)} \lim_{k \to \infty} \frac{\boldsymbol{u}_{k+1} - \boldsymbol{u}^*}{\boldsymbol{u}_k - \boldsymbol{u}^*}$$

where $\boldsymbol{u}^*$ is the "optimal" solution, precomputed by a dense linear solve. Computationally, $\omega$ is approximated using a brute-force sweep of values in $(0, 1)$.

## 2 Generating Coarse Grids

To train the model, a set of 6016 C/F grids were randomly generated. A "reference" C/F grid was first generated such that every third grid point is a coarse point, and the rest are fine points. Educated readers may recognize this as a *coarsening by 3*.

This reference grid was randomly permuted such that each grid point had a random probability of being flipped to the opposite value. I.e. coarse point to fine, and fine point to coarse. Random trials of the following probabilities were used:

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

For each value of $p$, 1000 random grids were generated according to the above permutation strategy. Each random grid was then used to solve the defined Poisson problem, and the optimal Jacobi weight, $\omega$, and convergence factor were recorded.

Then, in an attempt to train the model on a few "sane" grids, 16 grids with uniform spacing between coarse/fine points were generated. The coarsening factors used were:

$$r = \left\{ \frac{1}{9}, \quad \frac{1}{8}, \quad \frac{1}{7}, \quad \frac{1}{6}, \quad \frac{1}{5}, \quad \frac{1}{4}, \quad \frac{1}{3}, \quad \frac{1}{2}, \quad 2, \quad 3, \quad 4 \quad 5, \quad 6, \quad 7, \quad 8, \quad 9 \right\}$$

where values of $r < 1$ refer to grids where every $\frac{1}{r}$ points is a fine point and the rest coarse, and $r > 1$ refers to grids with every $r$ points being a coarse point and the rest fine. These grids were then also used to solve the Poisson problem and their weights and convergence factors recorded.

The code used to generate these grids is given in `grids/gen_grids.py`. Note that it can take a decent amount of time to run, approximately 21 minutes on my machine to completely run.

## 3 CNN Model