1 Loss Function

To train the networks, we will use a loss function based on the spectral radius of the error propagator:

$$\boldsymbol{E} := \boldsymbol{R}^{\nu} \boldsymbol{G} \boldsymbol{R}^{\nu} \tag{1}$$

where $\boldsymbol{R}$ is the error propagation per iteration of the relaxation scheme, ν is the number of pre-and-post-relaxation steps, and $\boldsymbol{G}$ is the coarse-grid correction error propagator,

$$\boldsymbol{G} := \boldsymbol{I} - \boldsymbol{P} \left(\boldsymbol{P}^T \boldsymbol{A} \boldsymbol{P} \right)^{-1} \boldsymbol{P}^T \boldsymbol{A}. \tag{2}$$

In this case, weighted Jacobi iteration is used for the relaxation scheme with a weight of $\omega = \frac{2}{3}$, giving an error propagator of

$$\boldsymbol{R} := \boldsymbol{I} - \frac{2}{3} \boldsymbol{D}^{-1} \boldsymbol{A}, \tag{3}$$

where $\boldsymbol{D} = \operatorname{diag}(\boldsymbol{A})$. In theory, an optimal interpolation operator should minimize the spectral radius of the error propagator, $\rho(\boldsymbol{E})$. However, backpropagation of the maximal eigenvector computed through, power iteration for example, tends to be rather unstable and can lead to numerical overflow[2]. Therefore, as proposed in [1], minimizing the squared Frobenius norm, $\|\boldsymbol{A}\|_F^2 = \sum \lambda_i^2(\boldsymbol{A})$ for SPD $\boldsymbol{A}$ can be done as a proxy.

Using purely the error propagator as the loss will simply minimize the number coarse points in the final interpolation operator, which is unwanted. Therefore, we will want to add a penalty proportional to the number of coarse points. If we allow $\boldsymbol{c}$ to be the vector containing the C/F splitting and whose entries float between 0 (fine) and 1 (coarse), we can use the L_1 norm $\|\boldsymbol{c}\|_1$ to penalize how "dense" the vector is.

Bringing this all together, we obtain the final loss function

$$\ell := \|\boldsymbol{E}\|_F^2 + \alpha \|\boldsymbol{c}\|_1 \tag{4}$$

for some scaling coefficient α.

There still remains the question of how do we represent the interchange of data between the "CF" network and the "P" network? If we were to discretely split the C/F space at 0.5, and for example take $c_i \geq 0.5$ to be coarse points and $c_i < 0.5$ to be fine points, we would run into the issue of having a discontinuous loss function in that gradient-descent based methods could not effectively train our pair of networks. Hence in the following section, we will introduce the *continuous formulation* of the error propagator.

1.1 Continuous Formulation

Let $\hat{\boldsymbol{P}} \in \mathbb{R}^{N_F \times N_F}$ be the "full" interpolation operator that is obtained from the output of the *P-net* on every node. Define $\boldsymbol{c}$ to be the vector encoding the coarse/fine selection such that

$$\begin{aligned} c_i &= 1 \text{ if node } i \text{ is coarse} \\ c_i &= 0 \text{ if node } i \text{ is fine,} \end{aligned}$$

and let $\boldsymbol{C} := \operatorname{diag}(\boldsymbol{c})$. Let $\boldsymbol{P} \in \mathbb{R}^{N_F \times N_C}$ be the *conventional* operator that one would obtain if the *P-net* was run on *only coarse nodes*. We will show in the remainder of this section that defining

$$\bar{\boldsymbol{P}} = \hat{\boldsymbol{P}}\boldsymbol{C} \tag{5}$$

$$\bar{\boldsymbol{G}} = \boldsymbol{I} - \bar{\boldsymbol{P}}\left(\bar{\boldsymbol{P}}^T\boldsymbol{A}\bar{\boldsymbol{P}} + \boldsymbol{I} - \boldsymbol{C}\right)^{-1}\bar{\boldsymbol{P}}^T\boldsymbol{A} \tag{6}$$

results in $\boldsymbol{G} = \bar{\boldsymbol{G}}$ in the discrete case of c_i being *either* 0 or 1.

First, let us consider the matrix $\bar{\boldsymbol{P}} = \hat{\boldsymbol{P}}\boldsymbol{C}$. Its structure is defined such that it has either the columns of $\boldsymbol{P}$ when $c_i = 1$ or columns of zeros when $c_i = 0$. Thus multiplying $\boldsymbol{B}\bar{\boldsymbol{P}}$ for some matrix $\boldsymbol{B} \in \mathbb{R}^{N_F \times N_F}$ will give the same product as $\boldsymbol{B}\boldsymbol{P}$ except with extra columns of zeros added. Equivalently, $\bar{\boldsymbol{P}}^T\boldsymbol{B}$ will result in $\boldsymbol{P}^T\boldsymbol{B}$ except with rows of zeros added.

Using this information, let us look at the $\bar{\boldsymbol{P}}^T\boldsymbol{A}\bar{\boldsymbol{P}} + \boldsymbol{I} - \boldsymbol{C}$ term. The coarse projection $\bar{\boldsymbol{P}}^T\boldsymbol{A}\bar{\boldsymbol{P}}$ is nothing more than $\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}$ except with rows and columns of zeros added. Such a matrix is singular, however, because of these new zero entries. To remedy this, we add 1 along the diagonal via $(\boldsymbol{I} - \boldsymbol{C})$ where these singularities exist and get a matrix that is invertible.

The expression $\left(\bar{\boldsymbol{P}}^T\boldsymbol{A}\bar{\boldsymbol{P}} + \boldsymbol{I} - \boldsymbol{C}\right)^{-1}$ is equivalent to $\left(\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}\right)^{-1}$ with columns of the $N_F \times N_F$ identity matrix inserted. The sparsity pattern is preserved.

Proof. Define the permutation matrix $\boldsymbol{Q}$ that permutes the rows and columns of $\boldsymbol{S} := \bar{\boldsymbol{P}}^T\boldsymbol{A}\bar{\boldsymbol{P}} + \boldsymbol{I} - \boldsymbol{C}$ to get a block matrix structure in which the top-left entry is the $N_C \times N_C$ identity matrix:

$$\boldsymbol{Q}^T\boldsymbol{S}\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P} \end{bmatrix}. \tag{7}$$

Because $\boldsymbol{Q}$ is unitary, $\boldsymbol{Q}^T = \boldsymbol{Q}^{-1}$. This gives us the relationship

$$\left(\boldsymbol{Q}^T\boldsymbol{S}\boldsymbol{Q}\right)^{-1} = \boldsymbol{Q}^{-1}\boldsymbol{S}^{-1}\left(\boldsymbol{Q}\right)^{-1} = \boldsymbol{Q}^T\boldsymbol{S}^{-1}\boldsymbol{Q}. \tag{8}$$

Following the block structure, we also have that

$$\left(\boldsymbol{Q}^T\boldsymbol{S}\boldsymbol{Q}\right)^{-1} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \left(\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}\right)^{-1} \end{bmatrix}. \tag{9}$$

Substituting (9) into (8) and re-arranging gives

$$\boldsymbol{S}^{-1} = \boldsymbol{Q}\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \left(\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}\right)^{-1} \end{bmatrix}\boldsymbol{Q}^T. \tag{10}$$

Of course, $\boldsymbol{Q}^T$ is also a permutation matrix. Thus, the inverse of $\bar{\boldsymbol{P}}^T\boldsymbol{A}\bar{\boldsymbol{P}} + \boldsymbol{I} - \boldsymbol{C}$ is equal to the inverse of $\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}$ with columns of the identity added. □

Finally, we can show $\bar{\boldsymbol{P}}\left(\bar{\boldsymbol{P}}^T\boldsymbol{A}\bar{\boldsymbol{P}} + \boldsymbol{I} - \boldsymbol{C}\right)^{-1}\bar{\boldsymbol{P}}^T = \boldsymbol{P}\left(\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}\right)^{-1}\boldsymbol{P}^T$. We have already proven that $\left(\bar{\boldsymbol{P}}^T\boldsymbol{A}\bar{\boldsymbol{P}} + \boldsymbol{I} - \boldsymbol{C}\right)^{-1}$ is equivalent to $\left(\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}\right)^{-1}$ except for columns of the identity inserted. Left multiplying by $\bar{\boldsymbol{P}}$ and right multiplying by $\bar{\boldsymbol{P}}$ will interpolate the columns and rows back into the fine-grid space. Because of the 0 pattern in $\bar{\boldsymbol{P}}$, any 1's on the diagonal of the inverse term simply get eradicated. Therefore when the values of c_i are discrete, $\bar{\boldsymbol{G}} = \boldsymbol{G}$.

1.2 Additional Considerations

When we find $\left(\bar{\boldsymbol{P}}^T \boldsymbol{A} \bar{\boldsymbol{P}} + \boldsymbol{I} - \boldsymbol{C}\right)^{-1}$, we are left with 1's along the diagonal for coarse nodes. For the discrete case, this does not matter. However for continuous c_i, I wonder if the correct expression should be

$$\left(\bar{\boldsymbol{P}}^T \boldsymbol{A} \bar{\boldsymbol{P}} + \boldsymbol{I} - \boldsymbol{C}\right)^{-1} - \boldsymbol{I} + \boldsymbol{C}, \tag{11}$$

which subtracts out the $\boldsymbol{I} - \boldsymbol{C}$ that was added and would be closer to a "coarse-grid-inverse". It may be the case that this doesn't even matter, though, if the neural network can still learn anyway.

References

[1] I. Luz, M. Galun, H. Maron, R. Basri, and I. Yavneh, *Learning algebraic multigrid using graph neural networks*, 2020.

[2] W. Wang, Z. Dang, Y. Hu, P. Fua, and M. Salzmann, *Backpropagation-friendly eigendecomposition*, 2019.