Two-level FAS with Neural Networks for Coarse Operators

Tuyen Tran[1] , Panayot Vassilevski[2] and Aidan Hamilton[3]
[1] Portland State University, [2, 3] Lawrence Livermore National Laboratory/CASC

Abstract

We are solving nonlinear equations where only the action of the nonlinear mapping and its Jacobians are computationally available. The solution algorithm is a standard two-level full approximation scheme (FAS) multigrid. A coarse counterpart of the nonlinear mapping is constructed utilizing suitable deep neural networks (DNNs).

I. Problem Formulation

Given the system of nonlinear equations

$$F(u) = f, \tag{1}$$

where F: $\mathbb{R}^n \to \mathbb{R}^n$, and we have access only to its actions. We also assume that for any $\boldsymbol{u}$ and $\boldsymbol{g} \in \mathbb{R}^n$, $J_F(\mathbf{u})\boldsymbol{g} \in \mathbb{R}^n$ where J_F is the Jacobian.

A standard approach for solving (1) is the Inexact Newton method as below:

Algorithm 1 (Inexact Newton)

For a current approximation $\boldsymbol{u}$ of (1), we perform the iterative process:

- Compute the residual $\boldsymbol{r} = \boldsymbol{f} - F(\boldsymbol{u})$
- Find $\boldsymbol{y_m}$ by performing m iterations of the GMRES[a] that approximately solves
$$\min_y \|\boldsymbol{r} - J_F(\mathbf{u})y\|$$
- Update $\boldsymbol{u} = \boldsymbol{u} + \boldsymbol{y_m}$.

[a] GMRES: Generalized minimal residual.

II. Full Approximation Scheme (FAS)

a) The FAS

We are interested in the two-level FAS algorithm for solving (1). We define a coarse version of F, $F_c: \mathbb{R}^{n_c} \to \mathbb{R}^{n_c}$, for some $n_c < n$ and its Jacobian $J_c = J_{F_c}$. To communicate between the *fine level*, $\mathbb{R}^n$ and the *coarse level*, $\mathbb{R}^{n_c}$, we introduce two linear mappings:

- Coarse-to-fine mapping $P: \mathbb{R}^{n_c} \to \mathbb{R}^n$.
- Fine-to-coarse projection $\pi: \mathbb{R}^n \to \mathbb{R}^{n_c}$ such that $\pi P = I$.

For $\boldsymbol{u_c}$ and $\boldsymbol{g_c} \in \mathbb{R}^{n_c}$, the standard approximation for F_c and J_c are $P^T F(P\boldsymbol{u_c})$ and $P^T J_F(P\boldsymbol{u_c})P\boldsymbol{g_c}$, respectively.

Algorithm 2 (Two-level FAS)

For the current approximation $\boldsymbol{u}$ of (1), the two-level FAS method performs :

- Compute $\boldsymbol{y_m}$ using **Algorithm 1** and let $\boldsymbol{u_{\frac{1}{3}}} = \boldsymbol{u} + \boldsymbol{y_m}$.
- Form the coarse-level nonlinear problem for $\boldsymbol{u_c}$
$$F_c(\boldsymbol{u_c}) = \boldsymbol{f_c} := F_c\left(\pi \boldsymbol{u_{\frac{1}{3}}}\right) + P^T(\boldsymbol{f} - F(\boldsymbol{u_{\frac{1}{3}}})). \tag{2}$$

Solve (2) using **Algorithm 1** with the coarse Jacobian, and $\boldsymbol{u_c} := \pi \boldsymbol{u_{\frac{1}{3}}}$.

- Update fine-level approximation $\boldsymbol{u_{\frac{2}{3}}} = \boldsymbol{u_{\frac{1}{3}}} + P^T(\boldsymbol{u_c} - \pi \boldsymbol{u_{\frac{1}{3}}})$.
- Repeat the FAS cycle starting with $\boldsymbol{u} := \boldsymbol{u_{\frac{2}{3}}}$.

b) **Our choice for $\boldsymbol{F_c}$**

For a given P, we train a DNN which takes any coarse vector $\boldsymbol{v_c} \in \mathbb{R}^{n_c}$ and produces $P^T F(P\boldsymbol{v_c}) \in \mathbb{R}^{n_c}$. The trained this way DNN gives the action of our coarse nonlinear mapping $F_c(.)$.

III. Numerical Example

Given the nonlinear PDE

$$-\nabla \cdot (k(u)\nabla u) + u = f \text{ on } \Omega$$
$$\text{and} \quad \nabla u \cdot \vec{n} = 0 \text{ on } \partial\Omega. \tag{3}$$

Here, $\Omega \subset \mathbb{R}^2$.

The variational formulation for (3) is:

$$\int_\Omega (k(u)\nabla u \nabla v + uv)dx = \int_\Omega fv dx \text{ for } u, v \in H^1.$$

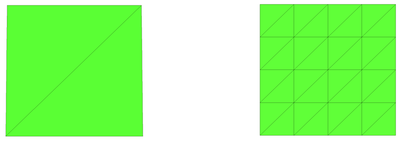

Figure 1: An example of coarse mesh and fine mesh for problem (3).

This form is discretized by the Finite Element Method.

We use 2 layers with n_C hidden nodes (denoted by N) at each layer along with 10000 training examples. Here, $n = 9$ and $n_c = 4$.

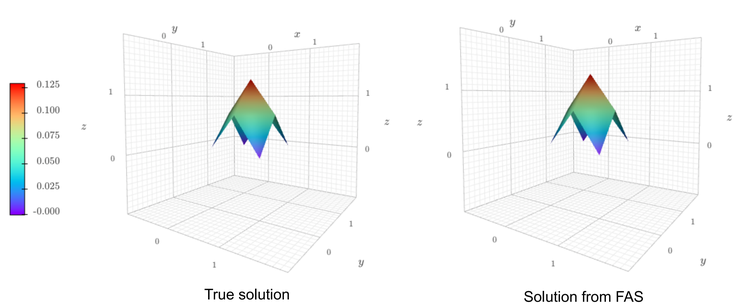

Figure 2: A visualization of solutions on the fine level with $n = 9$.

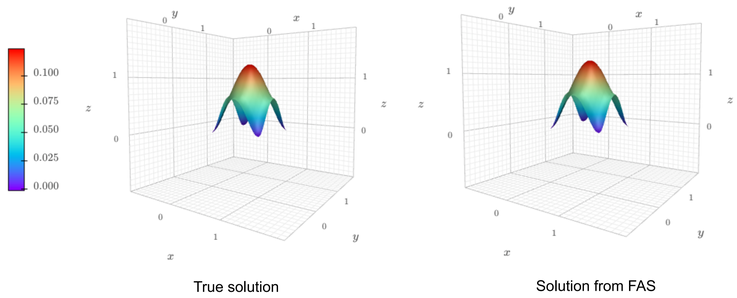

Figure 3: A visualization of solutions on the fine level with $n = 15$.

Number of layers	N	Training accuracy	Testing accuracy	Validation accuracy
1	4	0.5744	0.5760	0.5620
1	8	0.5125	0.5055	0.5400
2	4	0.7528	0.7540	0.7230
2	8	0.9608	0.9560	0.9680
3	4	0.9435	0.9530	0.9433
3	8	0.9634	0.9389	0.9216

Table 4: Comparison between neural network structures.

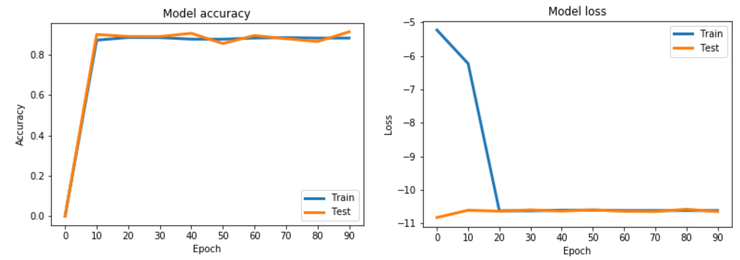

Figure 5: Accuracy and Loss (log scale) graphs using 100 epochs

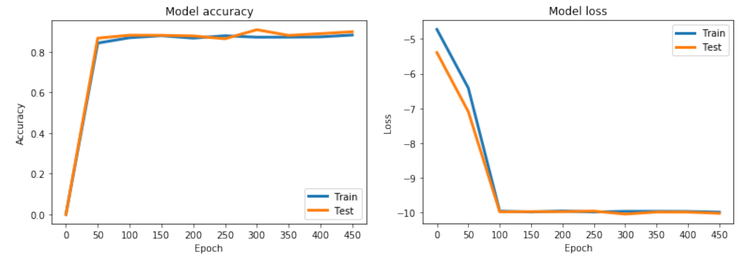

Figure 6: Accuracy and Loss (log scale) graphs using 500 epochs

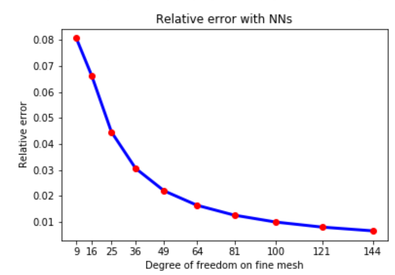

Figure 7: Relative error between true solution and FAS solution with $n_c = 4$.

Error	Value
$\|F_c(u_c) - F_{c_{DNN}}(u_c)\|_2$ min	1.5076e-06
$\|F_c(u_c) - F_{c_{DNN}}(u_c)\|_2$ max	4.9307e-06
$\|F_c(u_c) - F_{c_{DNN}}(u_c)\|_2$ average	3.4557e-06
$\|F_c(u_c) - F_{c_{DNN}}(u_c)\|_\infty$ average	8.5480e-06

Table 8: Error norms between the nonlinear operator and its approximation over 1000 examples.

IV. Conclusion & Future Work

- We replace the nonlinear operator with the neural network approximations.
- Improve the networks to compute $\boldsymbol{F_c}$ on the coarse level more efficiently.
- The approach is *feasible* on an element level and *parallelizable*.

Software

Python using FEniCS and Keras.

References

1. Vassilevski, Panayot S. *Multilevel block factorization preconditioners: Matrix-based analysis and algorithms for solving finite element equations.* Springer Science & Business Media, 2008.
2. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.
IM Release number: 782216