Accelerating GMRES with Deep Learning in Real-Time

Kevin Luna[1], Johannes Blaschke[2]
[1]University of Arizona, [2]NERSC, Lawrence Berkeley National Laboratory

Abstract

We present progress toward a methodology that uses deep learning (DL) methods in real-time to speed up iterative linear solvers for Partial Differential Equations (PDE). We found that this is possible, and that meaningful 2x speed-ups can be obtained.

Background

- PDE-based simulations often compute solutions through iterative methods for linear systems such as GMRES
- The performance of these methods depend highly on preconditioners -- the "initial guess" provided to the algorithm
- The state of the art for constructing preconditioners requires an experienced user with physical and empirical insight

Key Ideas

- **Key Idea**: Use DL to generate tuned preconditioners in real-time to speed up performance of GMRES
- **Novel Aspect:** Using DL methodologies typically used for "offline" learning in an "online" setting
- **Mechanics of Implementation:** Use PyTorch DL framework to monitor GMRES, and train a neural network in short bursts as data is collected so that consecutively improved initial guesses are provided by the neural network

Experimental Platform

- Experiments were performed on single NERSC Cori GPU node
- The neural network was trained using a NVIDIA V100 GPU
- The un-preconditioned GMRES algorithm ran on a single thread on the node's CPU

Methodology

- **Outline:** We train a neural network $N(b)$ in real-time as linear problems $\mathbf{A}x_i = b_i$ are solved using GMRES. The training goal is to have the neural network provide an initial guesses $N(b_i) = x_i^0$ that reduces the time-to-solution given the RHS b_i as the input
- Physically relevant application: discrete Poisson problem

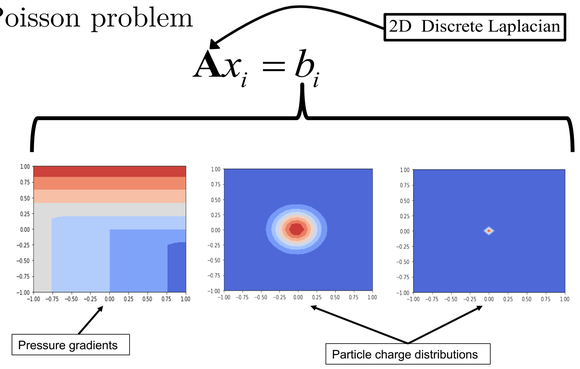

Figure 1: Examples of RHS used. The choice of localized sources and fields with linear gradients are motivated by physical applications

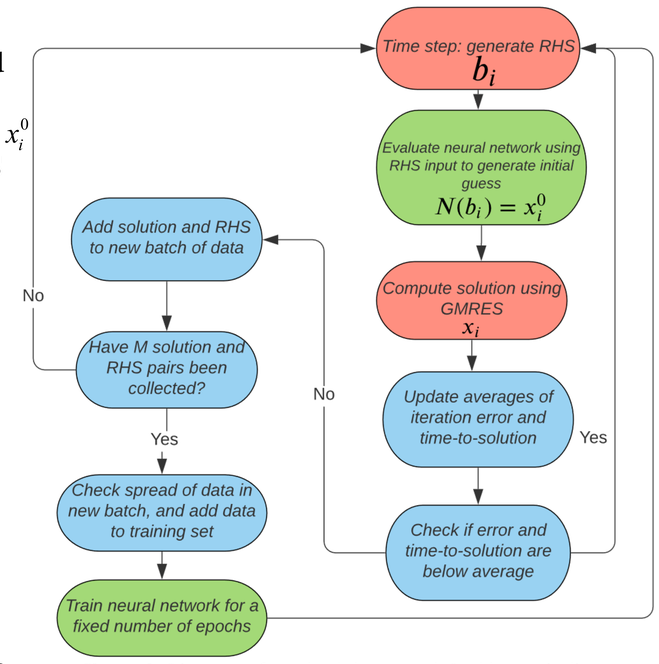

Figure 2: Diagram of methodology used. Red boxes indicate traditional computational steps, blue boxes indicate data curation steps, and green boxes indicate deep learning steps

- **Deep Learning:** A simple network that can trained quickly is key for working online. For example, 4-6 layer CNNs with dilated kernels and circular padding that encode non-local information for the solution work well in practice
- **Crucial**: maximizing the "value" of each sample in the training set by checking the spread of the data frequently
- **Numerical Experiments:** We measure performance of un-preconditioned GMRES and GMRES assisted by the neural network (MLGMRES) on a sequence of 1000 different RHS

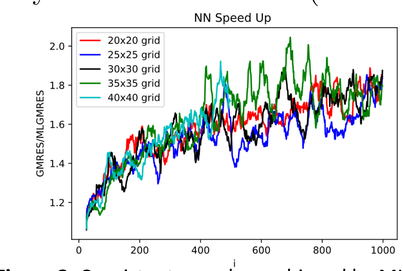

Figure 3: Consistent speed up achieved by MLGMRES for different resolutions

Grid size ($n \times n$)	20	25	30	35	40
Total MLGMRES Time (s)	667	1238	1971	3022	5352
Total Training Time (s)	95.1	86.2	95.2	101.5	108.1
Total Training Time/Total Solver Time	0.14	0.07	0.05	0.033	0.023

Table 1: Training time is a fraction of solver runtime

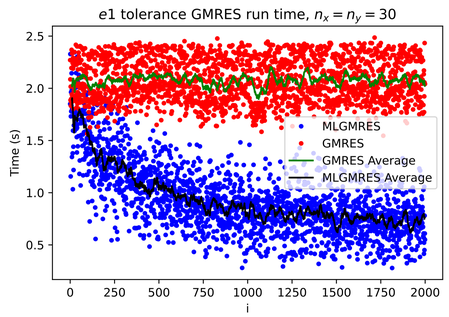

Figure 4: Speed up achieved by MLGMRES as it learns

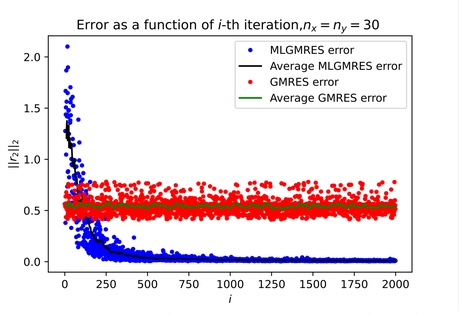

Figure 5: Convergence of error after two solver iterations

Discussion

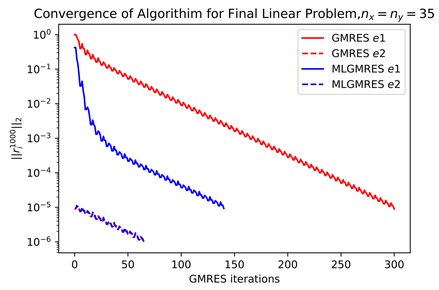

Figure 6: Convergence of residual for every GMRES and MLGMRES iteration

- **Take-away:** The approach used here accelerated the GMRES solver by providing an initial guess that temporarily increased the rate of convergence
- We have found that for our test problems we can utilize deep learning to accelerate an iterative linear solver in real-time (without human supervision)
- The training objective is not a model that produces an accurate first guess. Instead we train it to produce a first guess that is "close enough" *and* converges quickly (cf. fig 6)
- Maintaining "quality" of the training set (i.e minimizing redundant data) is crucial

Future Work

- Currently working on accelerating distributed-memory GMRES solver based on AMReX. This solver is part of the FHDeX software. We will target data parallel training on a single node of Cori GPU
- We plan to explore evolutionary Monte Carlo training of neural networks for right-preconditioned GMRES
- The prototype repo can be found at: https://github.com/ML4FnP/GMRES-Learning

Acknowledgements

We thank the Computing Sciences Summer Student Program at Berkeley Lab, and the DOE Office of Science. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility.