Research in Industrial Projects for Students

Sponsor

Lawrence Livermore National Laboratory

Work Statement

Non-Intrusive Parallel-in-Time Solvers for Challenging Problems

Student Members

Margaret Luo (Project Manager), *University of California, San Diego*

Bryan Li, *University of California, Berkeley*

Daniel Agraz Vallejo, *CETYS University*

Tran Duy Anh Le, *University of Rochester*

Academic Mentor

Jean-Michel Maldague

Sponsoring Mentors

Daniel Osei-Kuffuor

Rob Falgout

Rui Peng Li

Wayne Mitchell

Date: July 12, 2024

This project was jointly supported by Lawrence Livermore National Laboratory and NSF Grant DMS 1925919.

Contents

1 Problem Statement

This project develops parallel-in-time solvers for challenging time-dependent problems. Many phenomena are modelled with time-dependent partial differential equations (PDEs), such as weather models and wave propagation. Historically, computer simulations for these problems have been time-dependent, marching forward in time step-by-step. However, this approach of sequential time-stepping is becoming a bottleneck as it cannot be further parallelized, thus it is unable to take advantage of increasing number of processors on modern (super)computers. To avoid this bottleneck, researchers have developed new parallel-in-time solution approaches which significantly speed up computer simulations. These approaches solve for the full space-time system all at once in parallel.

The multigrid reduction in time (MGRIT) method and open-source software library XBraid were developed at Lawrence Livermore National Laboratory (LLNL) [1] to take advantage of existing simulation codes and technologies as much as possible, allowing scientists to migrate to a parallel-in-time simulation paradigm more easily. Significant advances have already been made on the development of MGRIT and other parallel-in-time approaches, but many outstanding research questions remain. One of the main research issues is the development of algorithms that are non-intrusive, i.e. algorithms that are easily adaptable to the wide-array of problems. Although MGRIT algorithms are designed to take advantage of existing code in a potentially non-intrusive way, for applications such as advection-dominated fluid dynamics, only highly intrusive algorithms have been developed [2, 3, 4, 5].

In this project we will approach developing non-intrusive algorithms by constructing effective MGRIT coarse-grid operators. The work will begin with simple constant-coefficient diffusion and advection in 1D, then move through a sequence of increasingly more difficult problems as the new techniques are developed. Combinations of offline and online training techniques will be investigated. The project will use Python for algorithm development and parallel performance studies.

2 Background: Multigrid-in-time for PDEs

2.1 Finite Difference Discretization of an Example PDE

Finite difference methods (FDM) are numerical techniques used to approximate derivatives by using differences between function values at discrete points. An example application of this method is the *advection-diffusion equation*, a linear PDE that is used to describe the behaviour of a scalar field such as the concentration of a pollutant or a solvent in a liquid, that varies with both space and time.

The general form of the 1D advection-diffusion equation for $u\colon [0,1]_x \times [0,1]_t \to \mathbb{R}$ is given by

$$u_t - \varepsilon u_{xx} + a u_x = f, \tag{1}$$

where $\varepsilon \in [0, \infty)$, $a = a(x)$, $f = f(x,t)$.

To solve this differential equation, we employ discretization methods that help us convert continuous models that are difficult to solve analytically into a discrete counterpart that is feasible to solve numerically.

Domain Discretization: The continuous domain of the problem is divided into a finite set of points, forming a grid. We consider x and t to represent our space and time dimensions, respectively. In this case, as we are working with one dimension, we divide the interval into evenly-spaced grid

nodes. Let $M+1$ be the number of spatial divisions and $N+1$ is the total time divisions. A uniform discretization sets

$$x_i = hi, \quad h = \frac{1}{M}$$

and

$$t_j = kj, \quad k = \frac{1}{N}.$$

Additionally, We label u_i^j as our approximation to $u(x_i, t_j)$.

We can employ a finite difference scheme to obtain the following discretizations:

$$\begin{aligned} u_t &\approx \frac{u_i^{j+1} - u_i^j}{k} \\ u_{xx} &\approx \frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h^2} \\ u_x &\approx \frac{u_{i+1}^j - u_i^j}{h}. \end{aligned}$$

2.2 Relaxation Methods

Relaxation methods are iterative methods to solve linear systems of the form $Au = b$, given $A \in \mathrm{GL}_n(\mathbb{R})$ (invertible $n \times n$ square matrices) and $b \in \mathbb{R}^n$. The relaxation methods we consider are *splitting methods*, which work by splitting up A into $A = M + N$, where M is invertible. Then, define the recurrence $Mx_{k+1} + Nx_k = b$. The *Jacobi* and *Gauss-Seidel* methods are special cases of this general iterative method when M is chosen to be the diagonal or the lower triangle (including the diagonal) of A, respectively. Under certain assumptions on A, we can prove that given any initial guess x_0, we have that $x_k \to A^{-1}b$ as $k \to \infty$.

2.3 Multigrid Methods

The weighted Jacobi and Gauss-Seidel methods are good at eliminating high-frequency components of error, but are significantly slower at eliminating low-frequency components. In particular, smooth error is characterized to have small eigenmodes. If e is some smooth error vector, then

$$e^\top Ae = \sum_{i<j} -a_{ij}(e_i - e_j)^2 \ll 1.$$

Multigrid methods employ a grid hierarchy in an attempt to convert low-frequency errors on the fine grid into high-frequency errors on coarser grids. This allows relaxation to be more effective than just operating on the fine grid.

For some linear system $Au = f$, and $u^* = A^{-1}f$, given a guess u_0, we may compute the *residual* $r = f - Au_0 = A(u^* - u_0) = Ae$, where $e_0 = u^* - u_0$ is the *error*. Then, we solve the equation $Ax = r$, where the true solution is e_0. Then, given some approximation x to e_0, then we can update our approximation to $u_* = u_0 + e_0$ by setting $u_1 = u_0 + x$.

Then, starting from the finest grid (the green grid), we relax then restrict our estimate to the next finest grid (the red grid). Then, we relax and restrict recursively until we hit the coarsest grid. At that point, we recursively relax and then interpolate our estimate to the next finest grid until we are back at the finest grid, at which point we have completed one multigrid V-cycle. The steps in the V-cycle can be summarized in Figure 2.

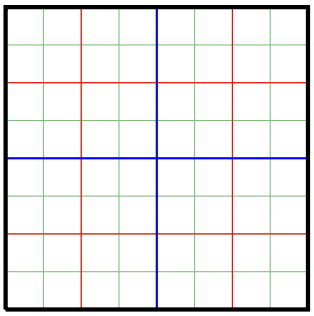

Figure 1: Multigrid Hierarchy

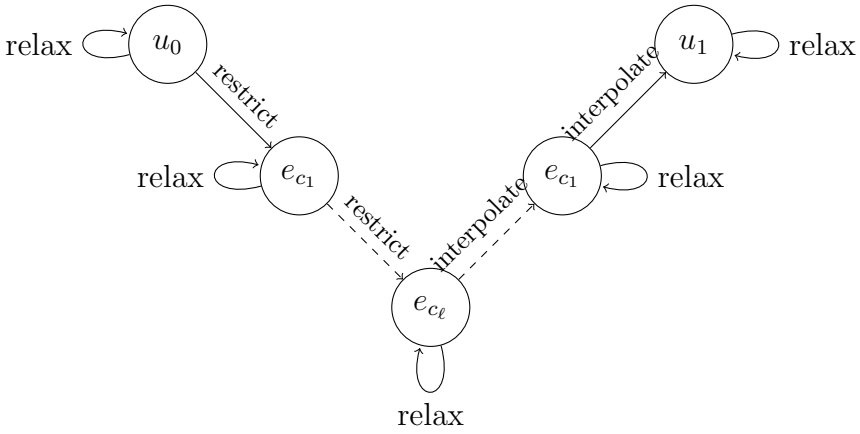

Figure 2: Multigrid V-Cycle

2.4 Multigrid-in-time

The multigrid reduction in time (MGRIT) algorithm is a parallel-in-time approach for solving time dependent problems that is designed to be as non-intrusive as possible on existing codes. For details, see [1].

Consider the update rule

$$u_i = \Phi_i(u_{i-1}) + g_i, \text{ for } i = 1, 2 \ldots, n \text{ and } u_0 = g_0$$

of some time-dependent process on the interval $[0, T]$. By using the partition as in Figure 3, we let $t_i = i\delta t$ for the fine time grid and $T_j = mj\delta t$ for the coarse time grid with coarsening factor m. Then, considering the linear case for simplicity, sequential time stepping is equivalent to a forward solve of the system $A\mathbf{u} = \mathbf{g}$

$$\begin{pmatrix} I & & & \\ -\Phi_1 & I & & \\ & \ddots & \ddots & \\ & & -\Phi_N & I \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_N \end{pmatrix}$$

where the solution $\mathbf{u}$ includes the state at all time points. This is an $O(N)$ sequential direct method so MGRIT replaces this with an $O(N)$ iterative method using parallel multigrid reduction to simultaneously solve at all time points. As such, MGRIT converges to the same solution that

serial time stepping produces. Additionally, the user must only define a routine that applies Φ, the map that advances the solution from one step to the next, potentially leading to a non-intrusive algorithm that simply reuses serial time-stepping code.

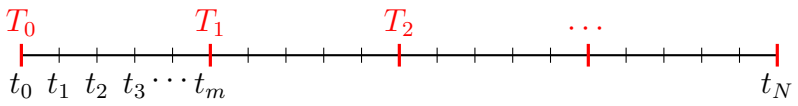

Figure 3: Coarse and fine time grids

3 Idea to Explore: Machine Learning Techniques to Approximate Coarse-Grid Operator

When executing the MGRIT method, we want to coarsen the grid so instead of using $\mathbf{u}$ and $\mathbf{g}$, we will use $\mathbf{u}_\Delta$ and $\mathbf{g}_\Delta$ that contain the value of u_{T_j} and g_{T_j}. Then we define the matrix A_Δ such that

$$A_\Delta \mathbf{u}_\Delta = \mathbf{g}_\Delta = R_\Psi \mathbf{g} \tag{2}$$

where R_Ψ is the restriction operator. Then we can write A_Δ as

$$A_\Delta = \begin{pmatrix} I & & & \\ -\Phi^m & I & & \\ & \ddots & \ddots & \\ & & -\Phi^m & I \end{pmatrix}$$

where we let $\Phi = \Phi_i$ and Φ^m be the ideal coarse time-stepping operator. However, using Φ^m offers no parallel speedup. Therefore, we want to utilize machine learning techniques to find a practical (sparse) coarse operator Ψ that approximates the action of the ideal operator [6]. In other words, we want to obtain a matrix

$$A_\Psi = \begin{pmatrix} I & & & \\ -\Psi & I & & \\ & \ddots & \ddots & \\ & & -\Psi & I \end{pmatrix}$$

which minimizes

$$f(\Psi) = \|I - A_\Psi^{-1} A_\Delta\|. \tag{3}$$

However, minimizing (3) directly is difficult, so one heuristic is to let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t$ be random vectors and minimize

$$\text{Opt}(\Psi) = \sum_{k=1}^{t} \frac{\|\Phi^m \mathbf{v}_k - \Psi \mathbf{v}_k\|}{\|I - \Phi^m \mathbf{v}_k\|},$$

which can be simplified further by letting Φ_r^m, Ψ_r be row r of Φ^m and Ψ respectively to the following

$$\sum_{k=1}^{t} \frac{(\Phi_r^m \mathbf{v}_k - \Psi_r \mathbf{v}_k)^2}{(1 - \Phi_r^m \mathbf{v}_k)^2}.$$

4 Project Objectives

4.1 Mathematical Background

Mathematical concepts we believe will be necessary for the projects includes: linear algebra, iterative methods, finite difference methods, multigrid methods, and interpolation.

4.2 Computing Background

For implementation of algorithms, we will need to familiarize ourselves with the method of MGRIT and tools such as PyMGRIT, PyTorch and Machine Learning techniques. We will start with implementation of Multigrid in time for constant-coefficient advection and diffusion problems in 1D.

4.3 Develop integration between PyMGRIT and PyTorch

Building on the idea of using Machine Learning approaches to approximate the coarse-grid operator, we will explore ways to integrate existing tools and libraries on solving space-time problems using multigrid and training ML models. We will be starting with applying machine learning techniques to 1D diffusion problem as we already know what is ideal.

4.4 Evaluate online and offline training techniques

If we receive good results from applying machine learning model to the 1D diffusion and advection problems, we will look into both online and offline training techniques and attempt to generalize the model to other differential equations as well.

4.5 Further Testing, Performance Study, and Optimization

To gain further understanding of new methods, we will look at the performance and efficiency of our models. We will do so by comparing the new non-intrusive methods developed to known algorithms (for cases where this is possible) and evaluate the parallel speedup over sequential time stepping.

5 Deliverables

5.1 From RIPS to LLNL

Summary Progress Report will be provided around Midterms (end of fifth week) along with a presentation.
Formal Presentation at the end of RIPS to document our findings throughout the project via presentation.
Code on trained model for approximation of coarse-grid operator depending on progress along the way as we explore the possibility of training Machine Learning models.
Error Analysis and results from looking at performance of new model compares to old MGRIT methods
Final Report with summary of our findings through following the objective outlines.

5.2 From LLNL to RIPS

Initial Code for a simple 1D diffusion problem with some machine learning integrated and additional resources such as notes and published papers needed to explain its conceptual basis will be provided.
Weekly Meetings (or as needed, decided by both RIPS and LLNL) in person or via zoom. This will be used to check in about progress on the project as well as to discuss new information that may help its development.
Site Visit to Lawrence Livermore National Laboratory in Livermore, California.

6 Project Timeline

We provide the following timeline as an estimation of our weekly progress throughout the project to ultimately achieve stated objectives. However, obstacles may arise which can lead to delays to this timeline.

Week 1: Start on first objective – learn mathematical background in numerical analysis, ODEs/PDEs, and linear algebra.

Week 2: Finish Work Statement; continue on first objective, start second objective as time permits

Week 3: Finish first objective and start on second objective after receiving initial code

Week 4: Start on Midterm presentation and summary report, start on third objective

Week 5: Practice and give midterm report; finish summary progress report; continue with third objective

Week 6: Wrap up third objective and depending on results, start on fourth objective

Week 7: Continue on fourth objective; start on final report

Week 8: Site visit; wrap up fourth objective and start on fifth objective; start preparation for final presentation and continue on final report

Week 9: Finishing up, final presentation.

References

[1] R. D. Falgout, S. Friedhoff, T. Kolev, S. MacLachlan, and J. Schroder, "Parallel time integration with multigrid," *SIAM Journal on Scientific Computing*, vol. 36, Dec 2014.

[2] H. D. Sterck, R. D. Falgout, S. Friedhoff, O. A. Krzysik, and S. P. MacLachlan, "Optimizing multigrid reduction-in-time and parareal coarse-grid operators for linear advection," *Numerical Linear Algebra with Applications*, vol. 28, no. e2367, 2021.

[3] H. D. Sterck, R. D. Falgout, O. A. Krzysik, and J. B. Schroder, "Efficient multigrid reduction-in-time for method-of-lines discretizations of linear advection," *Journal of Scientific Computing*, vol. 96, 2023.

[4] H. D. Sterck, R. D. Falgout, and O. A. Krzysik, "Fast multigrid reduction-in-time for advection via modified semi-lagrangian coarse-grid operators," *J. Sci. Comput.*, no. 45, pp. A1890–A1916, (2023).

[5] H. De Sterck, R. D. Falgout, O. A. Krzysik, and J. B. Schroder, "Parallel-in-time solution of scalar nonlinear conservation laws,"

[6] R. Huang, K. Chang, H. He, R. Li, and Y. Xi, "Reducing operator complexity in algebraic multigrid with machine learning approaches," *arXic preprint arXiv:2307.07695*, (2023).

IPAM Software Disclaimer for RIPS Sponsors

July 5, 2024

We want our RIPS sponsors to be aware of the nature of software developed by RIPS project teams. IPAM does not regard RIPS software as anything more than a prototype developed as a proof-of-concept only, and it is never developed for commercial use nor is it warranted by IPAM in any way. Here are some points to remember:

1. Software developed by a RIPS project team that appears to have been created wholly by a project team, may in fact contain proprietary codes borrowed from other sources; the sponsor must assume all risk for using such software.

2. IPAM makes every effort to discourage misuse of proprietary software by RIPS project participants; IPAM cannot be held responsible for such misuse.

3. As participants in an academic program, RIPS students will at times be permitted to use software that cannot be used by sponsors without a license.

4. Any restriction required by the sponsor on the use of special software, or platform needed to run the software, should be declared by the sponsor at the time of negotiating the project Work Statement. Otherwise the project team is free to choose software solutions as they see fit.