Multigrid Methods for Solving the "Good" Helmholtz Equation

D. Finn

May 6, 2021

1 Introduction

How do you solve a linear system of equations,

$$Ax = b \tag{1}$$

where $b \in \mathbb{R}^n$ is a known vector, $A \in \mathbb{R}^{n \times n}$ is a real, symmetric, positive definite matrix, and is very large and sparse? The traditional methods for solving this linear system can be extremely costly and typically converge slowly, if at all. For example, given a matrix A of size $n \times n$, Gaussian elimination has a complexity of $\mathcal{O}(n^3)$ which is not efficient when dealing with the large matrices that arise in numerical analysis [1]. A major step forward came in the 1950s when researchers realized that classical iteration methods all lead to solutions sequences that span the *Krylov Subspace*. Based on this idea, new fast iterative methods, such as Conjugate Gradients and Generalized Minimum Residual (GMRES), were derived. Then in the 1960s and 1970s, Nikolai Bakhvalov [2], Radii Petrovich Fedorenko [3, 4], Achi Brandt [5, 6], and Wolfgang Hackbusch [7], developed *Multigrid Methods* that further sped up iterative methods by applying them to multiple levels of discretizations.

For this project, we will explore using multigrid methods through the Portable, Extensible Toolkit for Scientific Computation (PETSc) to solve the "Good" Helmholtz equation. It is of interest to use the developed code to test and compare the multigrid solvers on CPUs and GPUs. GPUs present an interesting potential for code speedup in certain circumstances, however some literature [8] suggests that the GPU optimization will only become more efficient than CPU code at very large problem sizes. We hope to explore this idea more with this work.

2 Methods

Classical numerical methods for solving linear systems have revolved around various implementations of Gaussian elimination. These implementations work

to exploit the sparse structure of the coefficient matrix but often still have computational complexities near $\mathcal{O}(n^3)$ or convergence rates too slow to be used in a reasonable amount of time. Alternatively, numerous iterative methods have been developed that can significantly reduce computational complexity and speedup convergence.

In this section, we explore in more depth some of the concepts of iterative solvers that are used in this project. We start with Krylov and Multigrid methods and how PETSc handles both solvers. Lastly we describe the Helmholtz formulation used in this project.

2.1 Krylov Subspace Methods

Starting with some initial guess x_0, Krylov methods bootstraps up to more accurate approximations, x_k [1]. In a given iteration, k, Krylov methods produce an approximate solution, x_k, from a Krylov space generated by vector c,

$$\mathcal{K}_k(A, c) = span\{c, Ac, ..., A^{k-1}c\}. \tag{2}$$

2.1.1 Conjugate Gradients

To understand Krylov Methods further, we look at a specific Krylov algorithm, *Conjugate Gradients (CG)*. CG works by generating successive approximations (Krylov subspaces) to the solution, residuals corresponding to the approximations, and search directions used in updating the approximations and residuals. Figure 1 contains a *preconditioned conjugate gradients algorithm* where K is a preconditioner and z_i is the preconditioned residual. To obtain the *unpreconditioned conjugate gradients algorithm*, the preconditioner K can be set to the identity matrix I. In more detail, starting with some initial guess x_0, the first residual is calculated using,

$$r_0 = b - Ax_0. \tag{3}$$

In each successive CG iteration, the iterates x_i are updated by a multiple α_i of the search direction p_i,

$$x_i = x_{i-1} + \alpha_i p_i, \tag{4}$$

and the residual is updated with the equation,

$$r_i = r_{i-1} - \alpha_i q_i \;\; where, \tag{5}$$

$$q_i = Ap_i. \tag{6}$$

The constant,

$$\alpha_i = \frac{r_{i-1}^T r_{i-1}}{p_i^T q_i} \tag{7}$$

is chosen to minimize $r_i^T A^{-1} r_i$ over all possible α_i. The search directions p_i are updated using the residuals,

$$p_i = r_i + \beta_{i-1} p_{i-1}, \tag{8}$$

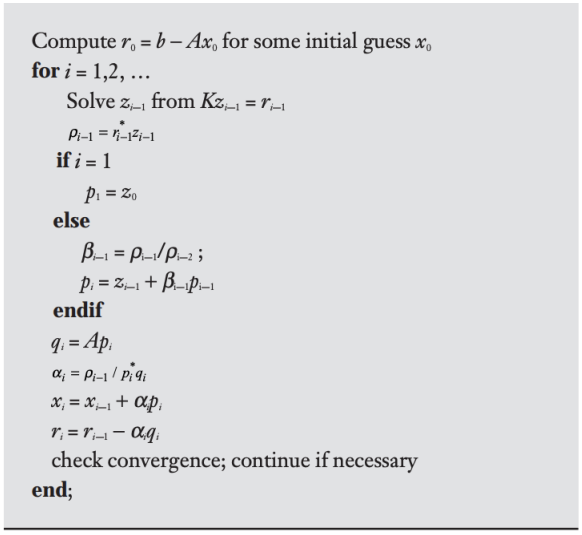

Figure 1: The conjugate gradients algorithm [9].

where,

$$\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}, \tag{9}$$

is chose to ensure p_i and Ap_{i-1} are orthogonal. Equivalently, this implies any two successive residuals, r_{i-1} and r_i, are also orthogonal. The process of orthogonalizing the search directions and residuals is called *Gram-Schmidt Process* [10]. Furthermore, it can be shown that the choice of β_i makes the search direction p_i and the residual r_i orthogonal to all previous iterations.

One main draw of the CG method is that, while the size of the Krylov subspaces can be quite large, the number of vectors that are actually kept in the memory is small. Additionally the time complexity for CG, given a finite element formulation of a second-order elliptic boundary value problem, is $\mathcal{O}^{3/2}$ in 2D and $\mathcal{O}^{4/3}$ in 3D [11]. Recalling that the comlexity for Guassian Elimination is $\mathcal{O}^3$, this is a considerable improvement in efficiency.

2.2 Multigrid Methods

Since their first description in the 60s, multigrid methods have become well known for being the fastest numerical method for solving elliptic boundary value problems. A good derivation and explanation of multigrid methods is given in [12], but we will provide some of the basic methodologies here. The basic concept behind multigrid is to provide a fine grid solver with a good initial guess obtained from a (or a sequence of) coarse grid solve(s) [12]. Relaxation (or iterative) methods are typically represented by Jacobi or Gauss-Seidel iteration. To start, consider the simple Two-Grid Correction Scheme ($v^h \leftarrow MG(v^h, f^h)$):

- Relax ν_1 times on $A^h u^h = f^h$ on W^h with initial guess v_h.
- Compute the fine-grid residual $r^h = f^h - A^h v^h$ and restrict it to the coarse grid by $r^{2h} = I_h^{2h} r^h$.
- Solve $A^{2h} e^{2h} = r^{2h}$ on W^{2h}.
- Interpolate the coarse-grid error to the fine grid by $e^h = I_{2h}^h e^{2h}$ and correct the fine-grid approximation by $v^h \leftarrow v^h + e^h$.
- Relax ν_2 times on $A^h u^h = f^h$ on Ω^h with initial guess v^h.

Here, Ω^h is the finest discretization level and $\Omega^2 h$ is the coarse grid, u^h is the solution vector, v^h is the current iterate approximate solution, r^h is the residual, $e^h = u^h - v^h$ is the error and I_{2h}^h is the interpolation operator that takes coarse-grid vectors and produces fine-grid vectors. In practice ν_1 is usually 1,2 or 3. In the scheme, relaxation on the fine grid eliminates the oscillatory components of the error, leaving behind only the smooth error. The smooth error leads to a very good interpolation on the fine grid meaning the correction should be effective.

As described above, the two-grid correction scheme still leaves room for some improvements. Primarily, a more direct solution of the residual equation is possible by applying recursion to the coarse-grid correction. This leads to the more common multigrid scheme, the V-Cycle Scheme ($v^h \leftarrow V^h(v^h, f^h)$):

- Relax on $A^h u^h = f^h$ ν_1 times with initial guess v_h.
- Compute $f^{2h} = I_h^{2h} r^h$.
  - Relax on $A^{2h} u^{2h} = f^{2h}$ ν_1 times with initial guess $v^{2h} = 0$.
  - Compute $f^{4h} = I_{2h}^{4h} r^{2h}$.
    - Relax on $A^{4h} u^{4h} = f^{4h}$ ν_1 times with initial guess $v^{4h} = 0$.
    - Compute $f^{8h} = I_{4h}^{8h} r^{4h}$.

      $\vdots$

      $\cdot$ Solve $A^{Lh} u^{Lh} = f^{Lh}$.

      $\vdots$

    - Correct $v^{4h} \leftarrow v^{4h} + I_{8h}^{4h} v^{8h}$.
    - Relax on $A^{4h} u^{4h} = f^{4h}$ ν_2 times with initial guess v^{4h}.
  - Correct $v^{2h} \leftarrow v^{2h} + I_{4h}^{2h} v^{4h}$.
  - Relax on $A^{2h} u^{2h} = f^{2h}$ ν_2 times with initial guess v^{2h}.
- Correct $v^h \leftarrow v^h + I_{2h}^h v^{2h}$.
- Relax on $A^h u^h = f^h$ ν_2 times with initial guess v^h.

The V-Cycle is just one of many different multigrid schemes.

2.3 Helmholtz Equation

2.3.1 Good Cop vs. Bad Cop

Solving the general Helmholtz equation,

$$-\Delta u + k^2 u = f, \tag{10}$$

with numerical methods, and particularly multigrid methods, is an open problem [13]. We will briefly touch upon why this is in this section before moving forward with the formulation used in this project. For simplicity, we consider only the cases where $k^2 = +1$ and $k^2 = -1$, respectively referred to as "Good Cop" and "Bad Cop" Helmholtz equations. First, we introduce the concepts of *positive definite* and *negative definite* matrices.

Definition 2.1. A $n \times n$ Hermitian matrix, A, is said to be **positive definite** if the scalar z^*Az is *positive* for all real, non-zero column vectors z of size n. Here, z^* denotes the conjugate transpose of the vector z. If however z^*Az is *negative*, then A is called a **negative definite** matrix. Lastly, if z^*Az can be either *positive* or *zero*, then it is called **positive semi-definite**, and, similarly, if z^*Az can be either *negative* or *zero*, it is called **negative semi-definite** [10].

It is worth noting that the existence of positive, negative or zero eigenvalues of a matrix can also be used to define what type of matrix A is. Since we strive to convert our initial equation formulation $-\Delta u + k^2 u = f$, also called the *strong form*, to a linear system $Au = f$, we can extend (2.1) to the strong form of our equations where $A = -\Delta + k^2$. Furthermore, in numerical methods, a positive definite operators are highly desirably as they can be easily inverted and contain other useful properties.

The negative Laplacian operator, $-\Delta$, is well known for being a positive definite operator. Therefore, the positive definiteness of the Helmholtz equation is dependent on the k^2 term. If $k^2 \geq 0$, i.e. the "Good Cop" form, then the equation remains positive definite and is solvable with common methods. However, if $k^2 \leq 0$, i.e. the "Bad Cop" form, then the equation becomes indefinite and solution becomes much trickier. As stated at the beginning of this section, solving the indefinite form of the Helmholtz equation is still an open research topic of interest.

2.3.2 Variational Formulation

For the reasons given in the previous section, we consider only the case where $k^2 = +1$, or the "Good Cop" Helmholtz equation [14],

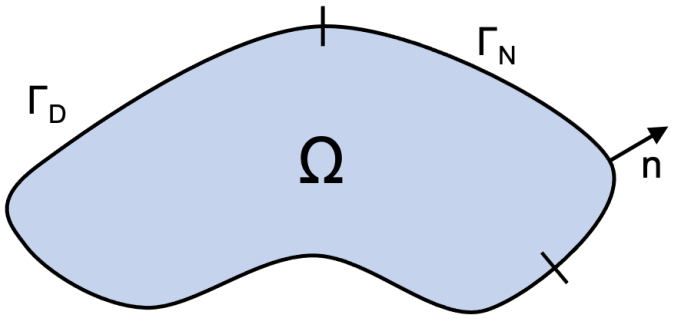

Figure 2: General domain for solving

$$-\Delta u + u = f \ \ in \ \ \Omega, \tag{11}$$

$$u = u_0 \ \ on \ \ \Gamma_D \subset \partial\Omega, \tag{12}$$

$$\nabla u \cdot n = g \ \ on \ \ \Gamma_N \subset \partial\Omega. \tag{13}$$

To discretize this equation, we use the finite elements method. The first step in discretization is to derive a weak (or variational) formulation. The process used to derive the weak form for the "Good Cop" Helmholtz equation in this project can be found in [15] and [16]. We start by multiplying by a test function v and integrating over the domain to obtain,

$$-\int_\Omega (\Delta u)\, v\, dx + \int_\Omega u\, v\, dx = \int_\Omega f\, v\, dx. \tag{14}$$

It is common when deriving weak formulations to try and reduce the order of the derivatives of u and v as much as possible. In the case of the Helmholtz equation, the laplacian term can be reduce to two first order gradients of u and v using integration by parts. The integration by parts formula reads,

$$-\int_\Omega (\Delta u)\, v\, dx = \int_\Omega \nabla u \cdot \nabla v - \int_\Gamma \frac{\partial u}{\partial n}\, v\, ds, \tag{15}$$

$$= \int_\Omega \nabla u \cdot \nabla v - \int_\Gamma \nabla u \cdot v\, ds, \tag{16}$$

where $\frac{\partial u}{\partial n} = \nabla u \cdot n$ is the derivative of u in the normal direction n on the boundary. We note that Gauss' Theorem is used to replace the volume integral with the surface integral. Using this formula on the Helmholtz equation gives,

$$\int_\Omega \nabla u \cdot \nabla v\, dx - \int_{\Gamma_N} v\, (\nabla u \cdot n)\, ds + \int_\Omega u\, v\, dx = \int_\Omega f\, v\, dx. \tag{17}$$

Another common rule when deriving weak formulations is that the test function v vanishes on the parts of the boundary where the solution u is known. This

requirement is explained in detail in [17]. Presently, this means $v = 0$ on the entire boundary $\partial\Omega$ simplifying the weak form to,

$$\int_\Omega \nabla u \cdot \nabla v \, dx + \int_\Omega u \, v \, dx = \int_\Omega f \, v \, dx. \tag{18}$$

We require that this formulation holds true for all test functions v in some *test space* $\hat{V}$. From this requirement, we can then determines a unique solution u which lies in some function space V, also called a *trial space*. For this problem, the test and trial spaces, V and $\hat{V}$, are defined as,

$$V = \{v \in H^1(\Omega) : v = u_0 \; on \; \partial\Omega\}, \tag{19}$$

$$\hat{V} = \{v \in H^1(\Omega) : v = 0 \; on \; \partial\Omega\}, \tag{20}$$

where $H^1(\Omega)$ is the well-known Sobolev space containing the functions v such that v^2 and $|\nabla v|^2$ have finite integrals over Ω. Therefore, the continuous solution of the Helmholtz PDE must lie in a function space where the derivatives are also continuous, however the Sobolev space $H^1(\Omega)$ allows functions with discontinuous derivatives. This weaker continuity condition of u in the weak formulation allows the use of piecewise polynomial function spaces, or simply put function spaces constructed by stitching together polynomial functions on simple domains such as intervals (1D), triangles (2D) or tetrahedrons (3D).

The next step is to replace the infinite-dimensional function spaces V and $\hat{V}$ with discrete (finite-dimensional) trial and test spaces $V_h \subset V$ and $\hat{V}_h \subset \hat{V}$. We define the discrete form of u, $u_h \in V_h \subset V$ and plug it back into the variational formulation which gives the discretretized weak form,

$$\int_\Omega \nabla u_h \cdot \nabla v \, dx + \int_\Omega u_h \, v \, dx = \int_\Omega f \, v \, dx \;\; \forall v \in \hat{V}_h \subset \hat{V}. \tag{21}$$

This discretized weak form along with suitable definitions of the discrete trial and test spaces V and $\hat{V}$ define the approximate numerical solution for the "Good Cop" Helmholtz equation. We assume that we have a basis $\{\phi_j\}_{j=1}^N$ for V_h and a basis $\{\hat{\phi}_i\}_{i=1}^N$ for $\hat{V}_h$, where N is the dimension of the space V_h. We can then make an ansatz for u_h in terms of the basis functions of the trail space,

$$u_h(x) = \sum_{j=1}^{N} U_j \phi_j(x), \tag{22}$$

where $U \in \mathbb{R}^N$ is the vector of defrees of freedom to be computed [15]. Plugging the bases into the variational form gives,

$$\sum_{j=1}^{N} U_j \int_\Omega \nabla \phi_j \cdot \nabla \phi_i \, dx + \sum_{j=1}^{N} U_j \int_\Omega \phi_j \, \phi_i \, dx = \int_\Omega f \, \phi_i \, dx \quad i = 1, 2, ..., N. \tag{23}$$

Therefore,a finite element solution $u_h(x) = \sum_{j=1}^N U_j \phi_j(x)$, can be computed for this system by solving the linear system,

$$AU = b \tag{24}$$

where,

$$A_{ij} = \int_\Omega \nabla\phi_j \cdot \nabla\phi_i \, dx + \int_\Omega \phi_j \, \phi_i \, dx \tag{25}$$

$$b_i = \int_\Omega f \, \phi_i \, dx. \tag{26}$$

2.3.3 Method of Manufactured Solutions

Thus far, we have used a generalized form of the "Good Cop" Helmholtz equation, including some function f and domain Ω, to derive the necessary formulations needed for our code. For the implementation, however, we require some specific choices for f, Ω and u_0. Furthermore, it is wise to first choose an exact solution for u and derive f, Ω and u_0 based on the exact solution. This process is referred to as the *Method of Manufactured Solutions*. Therefore, we choose our exact solution to be,

$$u_e(x, y) = 1 + x^2 + 2y^2. \tag{27}$$

Inserting u_e into equation (13) we find that u_e is a solution if,

$$f(x, y) = -5 + x^2 + 2y^2, \tag{28}$$

$$u_0(x, y) = 1 + x^2 + 2y^2. \tag{29}$$

We also choose the domain to be,

$$\Omega = [0, 1] \times [0, 1]. \tag{30}$$

Finally, plugging f into the discretized variational form gives,

$$\sum_{j=1}^{N} U_j \int_\Omega \nabla\phi_j \cdot \nabla\phi_i \, dx + \sum_{j=1}^{N} U_j \int_\Omega \phi_j \, \phi_i \, dx = \int_\Omega (-5 + x^2 + 2y^2) \, \phi_i \, dx \quad i = 1, 2, ..., N. \tag{31}$$

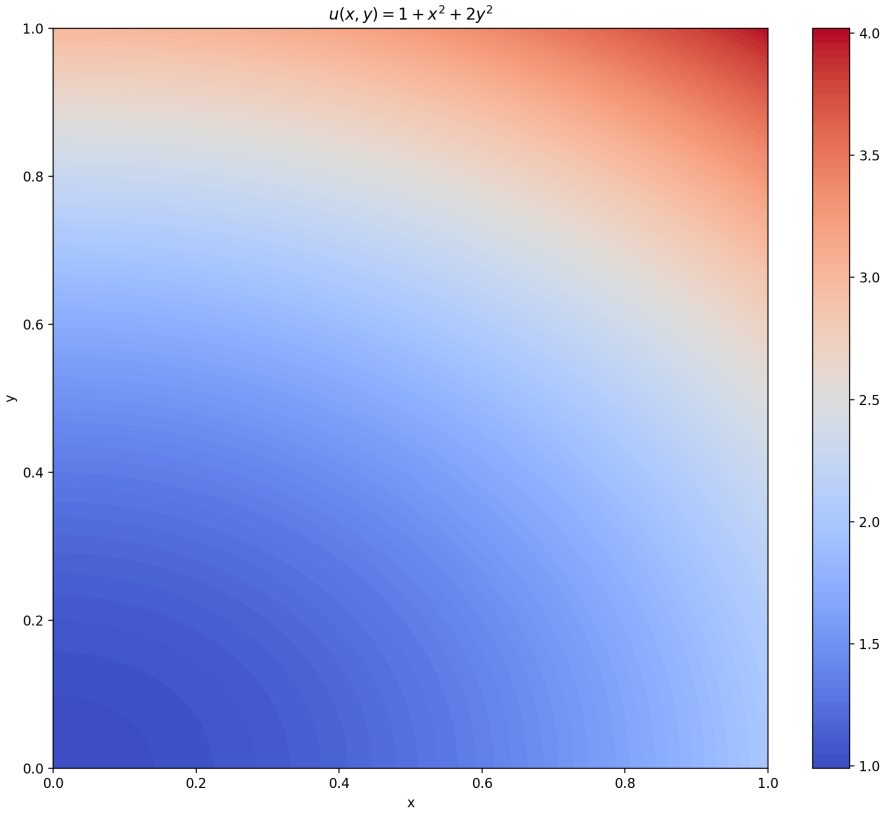

Figure 3: Multigrid V-Cycle (top) and F-Cycle (bottom).

Trigonometric Solution A second exact solution is needed if bilinear finite elements $(Q_1, Q_2, ...)$ are to be used. This exact solution also works for linear finite elements $(P_1, P_2, ...)$ which will primarily be used here. The exact solution is given by,

$$u_e(x, y) = Sin(2\pi x) + Sin(2\pi y). \tag{32}$$

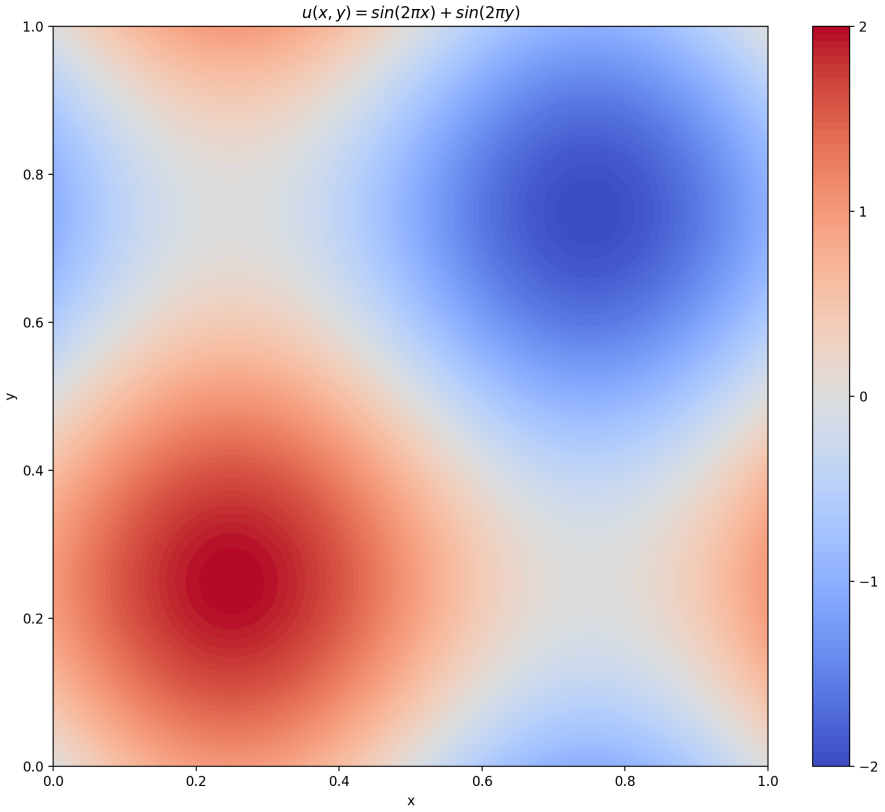

Figure 4: Multigrid V-Cycle (top) and F-Cycle (bottom).

2.3.4 PETSc Formulation

In PETSc, the *primal problem* is set by the functions PetscDSSetExactSolution, PetscDSSetResidual and PetscDSSetJacobian. Each function requires a specific equation form as the inputs. The residual function is requires the form,

$$\int_{\Omega} \phi_i f_0(u, u_t, \nabla u, x, t) + \nabla \phi_i \cdot \vec{f_1}(u, u_t, \nabla u, x, t), \tag{33}$$

and the jacobian function takes the form,

$$\begin{aligned}\int_\Omega & \phi_i g_0(u, u_t, \nabla u, x, t)\phi_j + \\ & \phi_i \vec{g}_1(u, u_t, \nabla u, x, t)\nabla\phi_j + \\ & \nabla\phi_i \cdot \vec{g}_2(u, u_t, \nabla u, x, t)\phi_j + \\ & \nabla\phi_i \cdot \overleftrightarrow{g}_3(u, u_t, \nabla u, x, t) \cdot \nabla\phi_k.\end{aligned} \tag{34}$$

Therefore, in the multigrid code we use the function inputs,

$$f_0 = u - (-5 + x^2 + 2y^2), \tag{35}$$

$$f_1 = \nabla u, \tag{36}$$

$$g_0 = 1.0, \tag{37}$$

$$g_1 = 0.0, \tag{38}$$

$$g_2 = 0.0, \ and \tag{39}$$

$$g_3 = 1.0. \tag{40}$$

3 Numerical Results

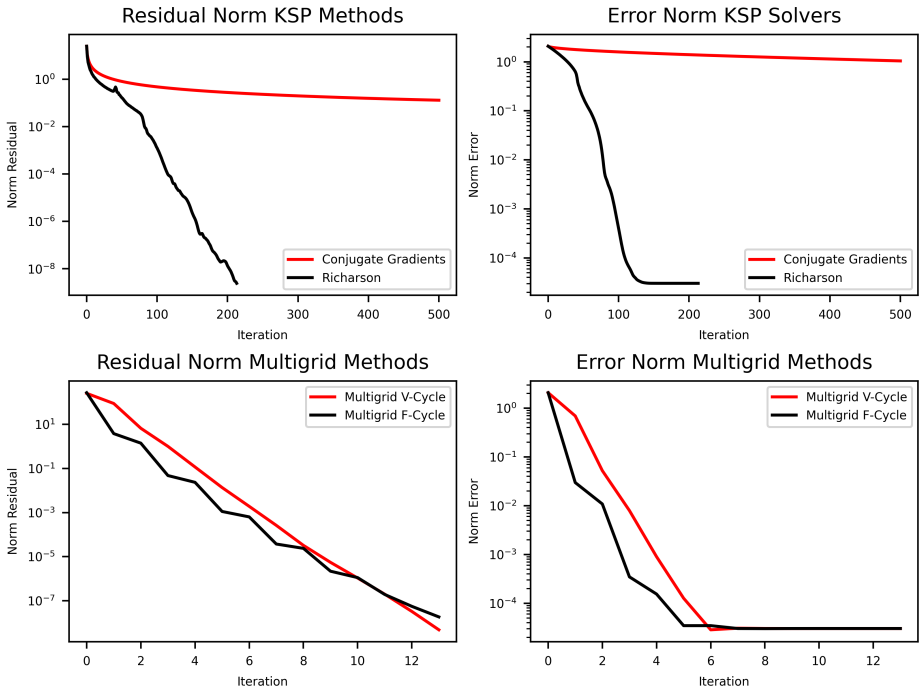

Figure 5: Convergence results for quadratic MMS. The top plots are residuals and errors for an SOR solver unaugmented (Richardson) and augmented with Krylov methods (Conjugate Gradients). The bottom plots are residuals and errors for the Multigrid Method V-Cycle and F-Cycle.

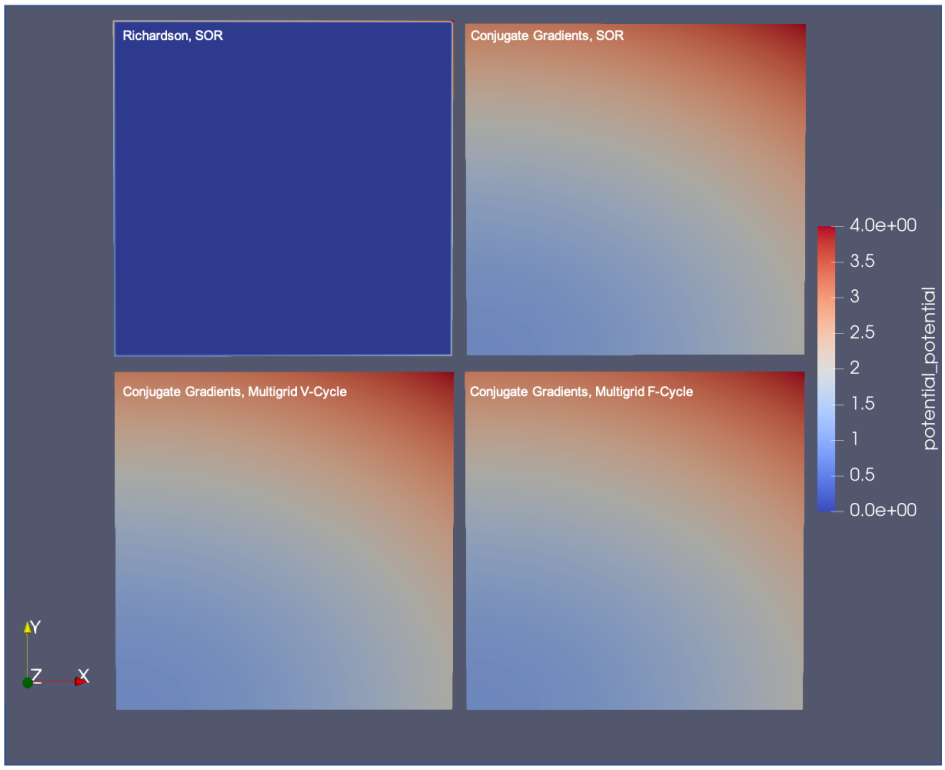

Figure 6: Solution visualization for quadratic MMS.

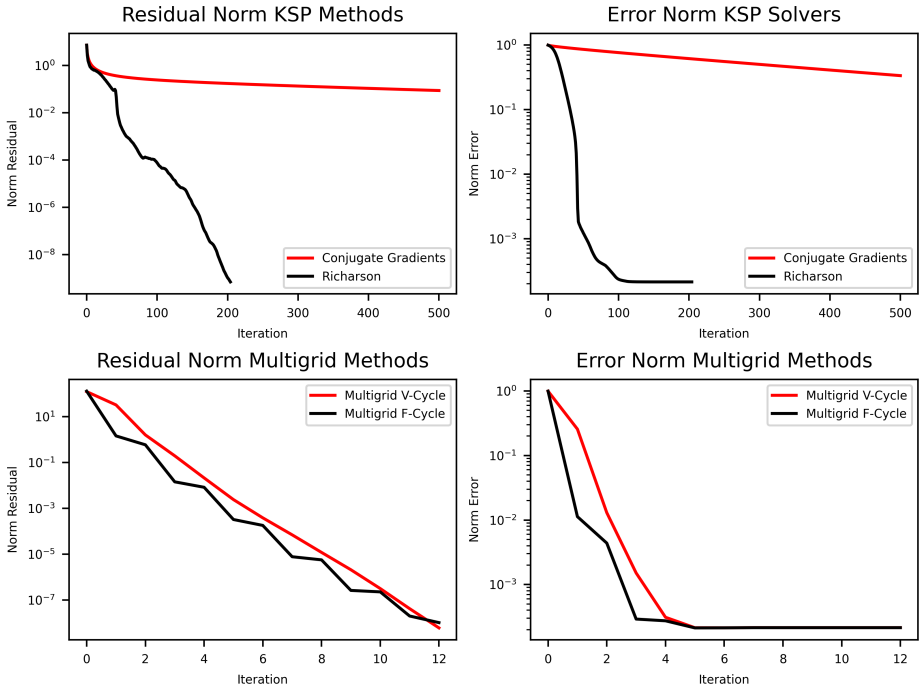

Figure 7: Convergence results for trigonometric MMS. The top plots are residuals and errors for an SOR solver unaugmented (Richardson) and augmented with Krylov methods (Conjugate Gradients). The bottom plots are residuals and errors for the Multigrid Method V-Cycle and F-Cycle.

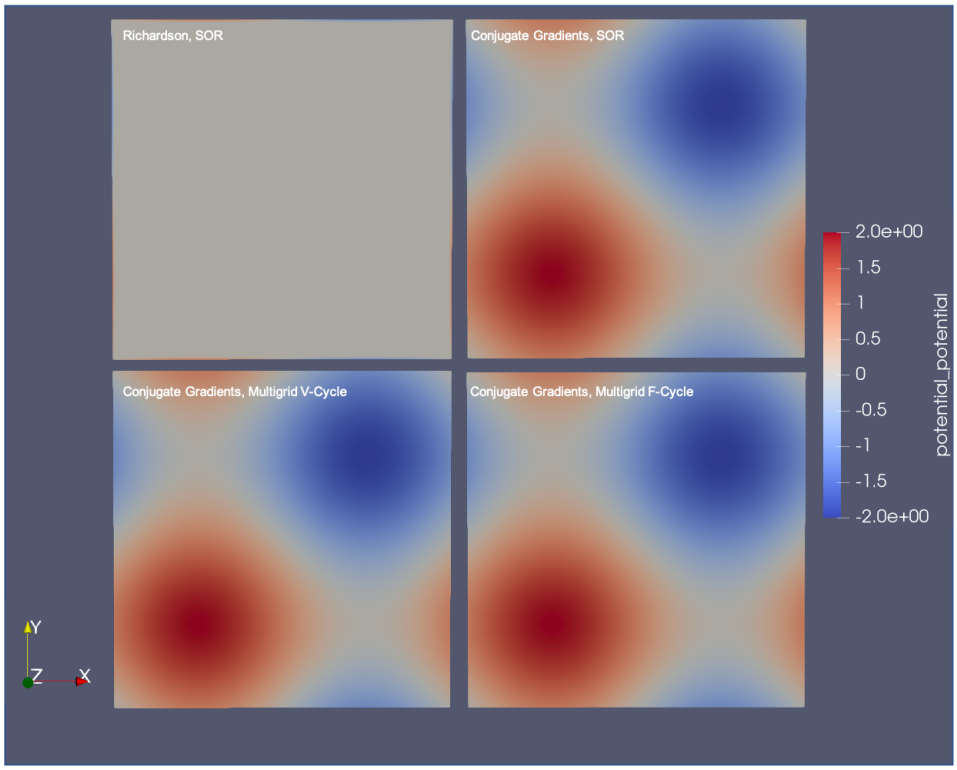

Figure 8: Solution visualization for trigonometric MMS.

Eight numerical tests were run to determine the efficiency and accuracy of the multigrid methods in comparison to traditional krylov solvers. The tests consisted of four tests with varying solvers for two different exact solutions, described in the Method of Manufactured Solutions section. The solvers chosen for the tests were:

1. Successive Over-Relaxation (SOR),
2. SOR with Conjugate Gradients Krylov Solver,
3. 2-level Multigrid V-Cycle using Conjugate Gradients and Jacobi Relaxation, and
4. Multigrid F-Cycle using Conjugate Gradients and Jacobi Relaxation.

The Richardson iterator is the most basic form of an iterator. Therefore, SOR is the primary solver in the Richardson test.

3.1 Single Level Solves

If a purely local solver is used, such as SOR, the residual and error initially decrease as we accurately solve local problems. However, to make more progress, we need increasingly more distant data which the local solver is not able to handle in any reasonable number of iterations. Thus, as shown in figures 5 and 7, there is a rapid slowdown and stagnation of the solve. The solution is visualized for each solver in figure 6, which also shows that SOR does not achieve any semblance of the correct solution. If SOR is augmented with a Krylov solver, say Conjugate Gradients, for problems of smaller sizes, we can achieve convergence. However, if the problem size is increased (which can be achieved by increasing -dm_refine_hierarchy to say 6), we once again see a stagnation in the solve. This is because Krylov solvers with local preconditioners asymptotically make no progress on a problem like this.

3.2 Multi Level Solves

To more efficiently solve this problem, the multigrid solver must use a coarse solver to eliminate error components not eliminated by the smoother. Two multigrid cycles were chosen for testing, shown in figure 9; the V-Cycle and F-Cycle (also known as Full Multigrid). The results for the two multigrid solves, shown in figures 5 and 7, clearly display convergence rates far faster than that of the single-level solves. The obtained solutions in figures 6 and 8 also show the accuracy of the multigrid solves, in comparison to the exact solutions in figures 3 and 4.

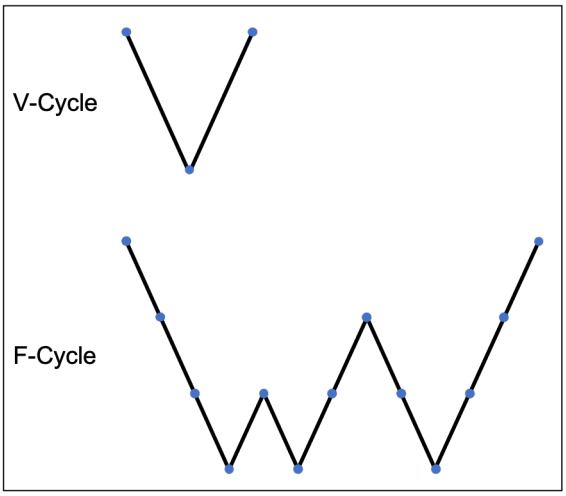

Figure 9: Multigrid V-Cycle (top) and F-Cycle (bottom).

3.3 GPU Code

As stated in the original project description, it is of interest to test the performance using GPUs. PETSc is configured with either MPICH or OpenMPI for parallelization of the codes. Additionally, OpenMPI can be configured with CUDA support which can be used to run PETSc codes on GPUs. MPICH is not a "GPU-aware" implementation of MPI, therefore reconfiguring PETSc with MPICH and CUDA can cause errors. The cleanest way to use CUDA in PETSc is to use OpenMPI through either a preinstalled library or by including "–download-openmpi –with-cuda" in the PETSc configure options.

References

[1] Ilse Ipsen and Carl Meyer. "The Idea Behind Krylov Methods". In: *The American Mathematical Monthly* 105 (Nov. 1997). DOI: 10.2307/2589281.

[2] N.S. Bakhvalov. "On the convergence of a relaxation method with natural constraints on the elliptic operator". In: *USSR Computational Mathematics and Mathematical Physics* 6.5 (1966), pp. 101–135. ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(66)90118-2. URL: https://www.sciencedirect.com/science/article/pii/0041555366901182.

[3] R.P. Fedorenko. "A relaxation method for solving elliptic difference equations". In: *USSR Computational Mathematics and Mathematical Physics* 1.4 (1962), pp. 1092–1096. ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(62)90031-9. URL: https://www.sciencedirect.com/science/article/pii/0041555362900319.

[4] R.P. Fedorenko. "The speed of convergence of one iterative process". In: *USSR Computational Mathematics and Mathematical Physics* 4.3 (1964), pp. 227–235. ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(64)90253-8. URL: https://www.sciencedirect.com/science/article/pii/0041555364902538.

[5] Achi Brandt. "Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems". In: *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*. Ed. by Henri Cabannes and Roger Temam. Berlin, Heidelberg: Springer Berlin Heidelberg, 1973, pp. 82–89. ISBN: 978-3-540-38377-2.

[6] Achi Brandt. "Multi-Level Adaptive Solutions to Boundary-Value Problems". In: *Mathematics of Computation* 31.138 (1977), pp. 333–390. ISSN: 00255718, 10886842. URL: http://www.jstor.org/stable/2006422.

[7] Wolfgang Hackbusch. "A fast numerical method for elliptic boundary value problems with variable coefficients". In: Oct. 1977, pp. 50–.

[8] Dave A. May et al. "Extreme-scale Multigrid Components within PETSc". In: *CoRR* abs/1604.07163 (2016). arXiv: 1604.07163. URL: http://arxiv.org/abs/1604.07163.

[9] Henk A. van der Vorst. "Krylov Subspace Iteration". In: *Computing in Science and Engineering* (Jan. 2000), pp. 32 –37.

[10] James P. Keener. *Principles of Applied Mathematics*. 2000. ISBN: 0-7382-0129-4.

[11] Jonathan R Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep. USA, 1994.

[12] William Briggs, Van Henson, and Steve McCormick. *A Multigrid Tutorial, 2nd Edition*. Jan. 2000. ISBN: 978-0-89871-462-3.

[13] O. G. Ernst and M. J. Gander. "Why it is Difficult to Solve Helmholtz Problems with Classical Iterative Methods". In: *Numerical Analysis of Multiscale Problems*. Ed. by Ivan G. Graham et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 325–363. ISBN: 978-3-642-22061-6. DOI: 10.1007/978-3-642-22061-6_10. URL: https://doi.org/10.1007/978-3-642-22061-6_10.

[14] Patrick E. Farrell. *Finite Element Methods for PDEs: C6.4 Hillary Term 2018*. 2018.

[15] Anders Logg, Garth Wells, and Kent-Andre Mardal. *Automated solution of differential equations by the finite element method. The FEniCS book*. Vol. 84. Apr. 2011. ISBN: 978-3-642-23098-1. DOI: 10.1007/978-3-642-23099-8.

[16] Hans Petter Langtangen and Anders Logg. *Solving PDEs in Python: The FEniCS Tutorial I*. 1st. Springer Publishing Company, Incorporated, 2017. ISBN: 3319524615.

[17] Hans Petter Langtangen and Kent-Andre Mardal. *Introduction to Numerical Methods for Variational Problems*. Jan. 2019. ISBN: 978-3-030-23787-5. DOI: 10.1007/978-3-030-23788-2.