

# Multigrid Methods for Solving the "Good" Helmholtz Equation

D. Finn

April 5, 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], Rii 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  $c$ ,

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

### 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,

$$r_0 = b - Ax_0. \quad (3)$$

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

$$x_i = x_{i-1} + \alpha_i p_i, \quad (4)$$

and the residual is updated with the equation,

$$r_i = r_{i-1} - \alpha_i q_i \text{ where,} \quad (5)$$

$$q_i = Ap_i. \quad (6)$$

The constant,

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

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

$$p_i = r_i + \beta_{i-1} p_{i-1}, \quad (8)$$

```

Compute  $r_0 = b - Ax_0$  for some initial guess  $x_0$ 
for  $i = 1, 2, \dots$ 
    Solve  $z_{i-1}$  from  $Kz_{i-1} = r_{i-1}$ 
     $\rho_{i-1} = r_{i-1}^T z_{i-1}$ 
    if  $i = 1$ 
         $p_1 = z_0$ 
    else
         $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$ ;
         $p_i = z_{i-1} + \beta_{i-1} p_{i-1}$ 
    endif
     $q_i = Ap_i$ 
     $\alpha_i = \rho_{i-1} / p_i^T q_i$ 
     $x_i = x_{i-1} + \alpha_i p_i$ 
     $r_i = r_{i-1} - \alpha_i q_i$ 
    check convergence; continue if necessary
end;

```

Figure 1: The conjugate gradients algorithm [9].

where,

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

is chosen 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  $\beta_i$  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 complexity for Gaussian 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  $\nu_1$  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  $\nu_2$  times on  $A^h u^h = f^h$  on  $\Omega^h$  with initial guess  $v^h$ .

Here,  $\Omega^h$  is the finest discretization level and  $\Omega^{2h}$  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  $\nu_1$  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  $A^h u^h = f^h$   $\nu_1$  times with initial guess  $v_h$ .
- Compute  $f^{2h} = I_h^{2h} r^h$ .
  - Relax on  $A^{2h} u^{2h} = f^{2h}$   $\nu_1$  times with initial guess  $v^{2h} = 0$ .
  - Compute  $f^{4h} = I_{2h}^{4h} r^{2h}$ .
    - Relax on  $A^{4h} u^{4h} = f^{4h}$   $\nu_1$  times with initial guess  $v^{4h} = 0$ .
    - Compute  $f^{8h} = I_{4h}^{8h} r^{4h}$ .
    - $\vdots$
    - Solve  $A^{Lh} u^{Lh} = f^{Lh}$ .
    - $\vdots$
    - Correct  $v^{4h} \leftarrow v^{4h} + I_{8h}^{4h} v^{8h}$ .
    - Relax on  $A^{4h} u^{4h} = f^{4h}$   $\nu_2$  times with initial guess  $v^{4h}$ .
    - Correct  $v^{2h} \leftarrow v^{2h} + I_{4h}^{2h} v^{4h}$ .
    - Relax on  $A^{2h} u^{2h} = f^{2h}$   $\nu_2$  times with initial guess  $v^{2h}$ .
- Correct  $v^h \leftarrow v^h + I_{2h}^h v^{2h}$ .
- Relax on  $A^h u^h = f^h$   $\nu_2$  times with initial guess  $v^h$ .

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

### 2.3 Helmholtz Equation

Solving the general Helmholtz equation  $-\Delta u + k^2 u = f$  with multigrid methods is an open problem in the field of numerical methods [13]. For that reason, we consider only the case where  $k^2 = 1$ , referred to as the "Good" Helmholtz equation [14],

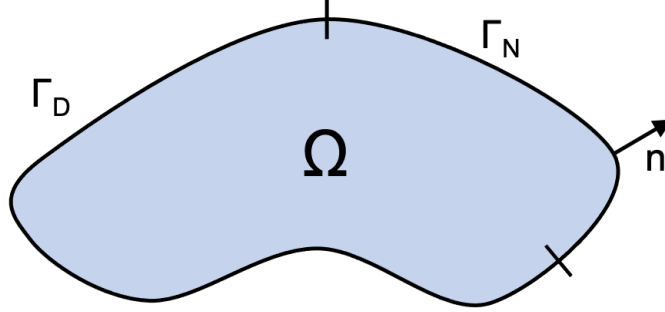


Figure 2: General domain for solving

$$-\Delta u + u = f \text{ in } \Omega, \quad (10)$$

$$u = u_0 \text{ on } \Gamma_D \subset \partial\Omega, \quad (11)$$

$$\nabla u \cdot n = g \text{ on } \Gamma_N \subset \partial\Omega. \quad (12)$$

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" 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,

$$-\int_{\Omega} (\Delta u) v \, dx + \int_{\Omega} u v \, dx = \int_{\Omega} f v \, dx. \quad (13)$$

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,

$$-\int_{\Omega} (\Delta u) v \, dx = \int_{\Omega} \nabla u \cdot \nabla v - \int_{\Gamma} \frac{\partial u}{\partial n} v \, ds, \quad (14)$$

$$= \int_{\Omega} \nabla u \cdot \nabla v - \int_{\Gamma} \nabla u \cdot v \, ds, \quad (15)$$

where  $\frac{\partial u}{\partial n} = \nabla u \cdot n$  is the derivative of  $u$  in the normal direction  $n$  on the boundary. Using this formula on the Helmholtz equation gives,

$$\int_{\Omega} \nabla u \cdot \nabla v \, dx - \int_{\Gamma_N} v (\nabla u \cdot n) \, ds + \int_{\Omega} u v \, dx = \int_{\Omega} f v \, dx. \quad (16)$$

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,

$$\int_{\Omega} \nabla u \cdot \nabla v \, dx + \int_{\Omega} u v \, dx = \int_{\Omega} f v \, dx. \quad (17)$$

We require that this formulation holds true for all test functions  $v$  in some *test space*  $\hat{V}$ . From this requirement, we can then determine 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,

$$V = \{v \in H^1(\Omega) : v = u_0 \text{ on } \partial\Omega\}, \quad (18)$$

$$\hat{V} = \{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\}, \quad (19)$$

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  $\Omega$ . 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 discretized weak form,

$$\int_{\Omega} \nabla u_h \cdot \nabla v \, dx + \int_{\Omega} u_h v \, dx = \int_{\Omega} f v \, dx \quad \forall v \in \hat{V}_h \subset \hat{V}. \quad (20)$$

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" 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 trial space,

$$u_h(x) = \sum_{j=1}^N U_j \phi_j(x), \quad (21)$$

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

$$\sum_{j=1}^N U_j \int_{\Omega} \nabla \phi_j \cdot \nabla \phi_i dx + \sum_{j=1}^N U_j \int_{\Omega} \phi_j \phi_i dx = \int_{\Omega} f \phi_i dx \quad i = 1, 2, \dots, N. \quad (22)$$

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,

$$AU = b \quad (23)$$

where,

$$A_{ij} = \int_{\Omega} \nabla \phi_j \cdot \nabla \phi_i dx + \int_{\Omega} \phi_j \phi_i dx \quad (24)$$

$$b_i = \int_{\Omega} f \phi_i dx. \quad (25)$$

### 2.3.1 Method of Manufactured Solutions

Thus far, we have used a generalized form of the "Good" Helmholtz equation, including some function  $f$  and domain  $\Omega$ , to derive the necessary formulations needed for our code. For the implementation, however, we require some specific choices for  $f$ ,  $\Omega$  and  $u_0$ . Furthermore, it is wise to first choose an exact solution for  $u$  and derive  $f$ ,  $\Omega$  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,

$$u_e(x, y) = 1 + x^2 + 2y^2. \quad (26)$$

Inserting  $u_e$  into equation (12) we find that  $u_e$  is a solution if,

$$f(x, y) = -5 + x^2 + 2y^2, \quad (27)$$

$$u_0(x, y) = 1 + x^2 + 2y^2. \quad (28)$$

We also choose the domain to be,

$$\Omega = [0, 1] \times [0, 1]. \quad (29)$$

Finally, plugging  $f$  into the discretized variational form gives,

$$\sum_{j=1}^N U_j \int_{\Omega} \nabla \phi_j \cdot \nabla \phi_i dx + \sum_{j=1}^N U_j \int_{\Omega} \phi_j \phi_i dx = \int_{\Omega} (-5 + x^2 + 2y^2) \phi_i dx \quad i = 1, 2, \dots, N. \quad (30)$$

### 2.3.2 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,

$$\int_{\Omega} \phi_i f_0(u, u_t, \nabla u, x, t) + \nabla \phi_i \cdot \vec{f}_1(u, u_t, \nabla u, x, t), \quad (31)$$

and the jacobian function takes the form,

$$\begin{aligned} & \int_{\Omega} \phi_i g_0(u, u_t, \nabla u, x, t) \phi_j + \\ & \phi_i \vec{g}_1(u, u_t, \nabla u, x, t) \nabla \phi_j + \\ & \nabla \phi_i \cdot \vec{g}_2(u, u_t, \nabla u, x, t) \phi_j + \\ & \nabla \phi_i \cdot \overleftrightarrow{g}_3(u, u_t, \nabla u, x, t) \cdot \nabla \phi_k. \end{aligned} \quad (32)$$

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

$$f_0 = u - (-5 + x^2 + 2y^2), \quad (33)$$

$$f_1 = \nabla u, \quad (34)$$

$$g_0 = 1.0, \quad (35)$$

$$g_1 = 0.0, \quad (36)$$

$$g_2 = 0.0, \text{ and } \quad (37)$$

$$g_3 = 1.0. \quad (38)$$

## 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](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](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](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](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.