ALGEBRAIC MULTIGRID METHOD TO SOLVE 1D POISSON EQUATION

Ajay Krishna P
NA18B011

Aniket Malviya
NA20B006

ABSTRACT

In this project,the focus is to solve the large linear sparse matrix system that arises out of discretizing the 1d poisson equation using Algebraic Multigrid Method (AMG). The multigrid method offers distinct advantages over direct solvers (LU decomposition) and basic iterative methods (Jacobi, Gauss-Seidel) for solving linear systems.The project focuses specifically on two variants: V-cycle and F-cycle.

Furthermore, in this project, the implementation of the multigrid method is parallelized using OpenMP, a widely-used API for shared-memory parallel programming in C. By distributing computational tasks across multiple cores, the aim is to exploit parallelism and accelerate the solution process.

INTRODUCTION

In this project, we delve into the efficient solution of the one-dimensional Poisson equation using Algebraic Multigrid (AMG) techniques. The 1D Poisson equation, given by

$$\frac{d^2u}{dx^2} = f(x)$$

where u represents the unknown function, $f(x)$ denotes the source term, and x denotes the spatial variable.

The Jacobi and Gauss-Seidel iterations yield smooth errors, with high-frequency components of the error vector e nearly eliminated after a few iterations, while low frequencies diminish slowly. The highly efficient multigrid approach involves transitioning to a coarser grid, where smoothness transforms into roughness, effectively treating low frequencies as high ones. On this coarser grid, a significant portion of the error can be eliminated. We iterate only briefly before transitioning between fine and coarse grids. Remarkably, multigrid can solve numerous sparse and realistic systems with high precision in a fixed number of iterations, independent of size.

In this report, we explore the principles behind AMG and its application to solve the 1D Poisson equation. We discuss the construction of the algebraic multigrid hierarchy, including the coarsening and interpolation strategies crucial for achieving rapid convergence. Through numerical experiments and performance benchmarks, we aim to showcase the superiority of AMG over traditional iterative solvers in tackling the 1D Poisson equation with sparse matrices.

NOMENCLATURE

- **u** - unknown function.
- **x** - spatial variable.
- **f(x)** - source term.
- $\Delta x =$ **h** - grid spacing.
- $\mathbf{M_{nh}}$ - M represents any matrix with grid spacing nh,where n is the ratio between current grid size and finest grid size.
- $\mathbf{I_{2nh}^{nh}}$ - Interpolation matrix which converts coarse grid (2nh) to fine grid (nh)
- $\mathbf{R_{nh}^{2nh}}$ - Restriction matrix which converts fine grid (nh) to coarse grid (2nh)
- $\mathbf{A_h}$ - Matrix that arises from discretizing the poisson equation.
- **B** - right hand side known vector.
- $\mathbf{E_{nh}}$ - The error vector at grid size nh.

PROBLEM STATEMENT

The 1D Poisson equation,

$$\frac{d^2u}{dx^2} = f(x)$$

is discretized as $\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} = f_i$

In our project we have assumed $f(x) = 1$ and $x \in [0, 2]$ with boundary condition $u(0) = 1,\ u(2) = 1$.
For example, $\Delta x = 0.25$. We will have 9 points in total and 7 interior points. By applying the boundary condition our linear system will be represented as,

$$\begin{bmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & 1 & -2 & 1 & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} \Delta^2 f_1 - u_0 \\ \Delta^2 f_2 \\ \Delta^2 f_3 \\ \Delta^2 f_4 \\ \Delta^2 f_5 \\ \Delta^2 f_6 \\ \Delta^2 f_7 - u_8 \end{bmatrix}$$

THEORY

Multigrid demonstrates remarkable efficacy, particularly with symmetric systems. Integral to its success are the novel components, denoted as rectangular matrices **R** and **I**, which facilitate grid transformations.

The restriction matrix **R** facilitates the transfer of vectors from the fine grid to the coarse grid.The return to the fine grid is executed via an interpolation matrix **I**= I_{2h}^{h}.The original matrix A_h on the fine grid is approximated by $\mathbf{A_{2h} = RA_{2h}I}$.

The Interpolation matrix **I** (7x3) is given by,

$$I = \frac{1}{2}\begin{bmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & 2 \\ & & 1 \end{bmatrix}$$

Transitioning from the fine grid h to the coarse grid 2h is facilitated by the restriction matrix $\mathbf{R} = \frac{1}{2}(I_{2h}^{h})^{\mathrm{T}}$.

$$R = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{bmatrix}$$

In the multigrid technique, the current residual $\mathbf{r_h = b - Au_h}$ is shifted to the coarse grid for processing. Iterations are conducted on this coarser **2h** grid to estimate the error $\mathbf{E_{2h}}$ at that level. Subsequently, the approximation is interpolated back to the fine grid as $\mathbf{E_h}$, where it is applied as a correction to $\mathbf{u_h + E_h}$, initiating the process again.

This loop of transitioning between fine and coarse grids forms a two-grid V-cycle, commonly referred to as a v-cycle. The process involves several steps, bearing in mind that the error resolves to $\mathbf{A_h(u - u_h) = b_h - A_hu_h = r_h}$

ALGORITHM v CYCLE

1. Iterate on $\mathbf{A_hu = b_h}$ to obtain $\mathbf{u_h}$.
2. Restrict the residual $\mathbf{r_h = b_h - A_hu_h}$ to $\mathbf{r_{2h} = R_h^{2h} r_h}$ at the coarse grid level.
3. Approximate the coarse-grid error: $\mathbf{A_{2h}E_{2h} = r_{2h}}$, yielding $\mathbf{E_{2h}}$.
4. Interpolate back to the fine grid :$\mathbf{E_h = I_{2h}^{h} E_{2h}}$ Then, add $\mathbf{E_h}$ to $\mathbf{u_h}$.
5. Iterate $\mathbf{A_hu = b_h}$ with guess value as $\mathbf{u_h + E_h}$ until convergence, typically involving several Gauss-Seidel steps.

Repeat the process iteratively, transferring the residual to the coarse grid, solving or approaching the solution there, and then interpolating back to the fine grid for further refinement.

Notice that the $\mathbf{A_{2h}}$ matrix is the same tridiagonal matrix in but in 3x3 dimension and also has a factor 0.25 attached to it.

$$A_{2h} = \frac{1}{4}\begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$$

In general $\mathbf{A_{nh}}$ will have a factor $\frac{1}{n^2}$

V-CYCLE and F-CYCLE

Certainly, multigrid methodology isn't restricted to just two grids. Limiting it to two grids would overlook its remarkable potential. Even on the 2h grid, the lowest frequency remains relatively low, implying that certain error components persist until we transition to even coarser grids like 4h or 8h (or even a significantly coarser 256h grid). Expanding beyond the two-grid v-cycle, this approach naturally extends to incorporate more grids. This extended framework can descend to even coarser levels (2h, 4h, 8h) and ascend back up to finer resolutions (4h, 2h, h). This nested succession of v-cycles constitutes a V-cycle (capital V) (Figure 1). Importantly, coarse grid iterations are notably faster than fine grid iterations, and analysis demonstrates the efficacy of allocating time to coarse grid computations.

To construct F-Cycle (Figure 2) we perform a half V cycle upto the coarsest level and then interpolate one level and

perform a full V cycle from this level and interpolate back.Repeat the process until we reach the fine grid.

Given that each step within every cycle entails solely **O(n)** operations on sparse problems of size n, multigrid maintains its **O(n)** efficiency. This characteristic remains consistent even in higher dimensions.

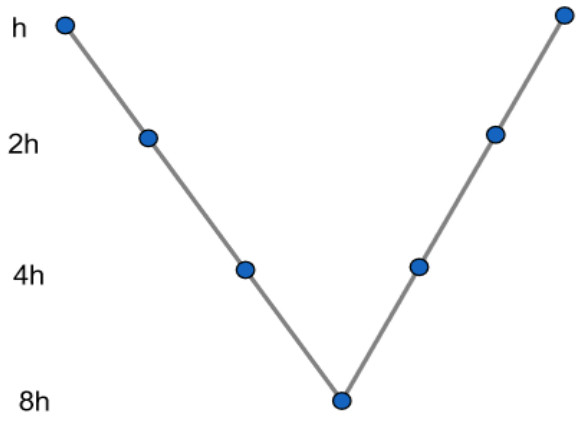


Figure 1: V-Cycle

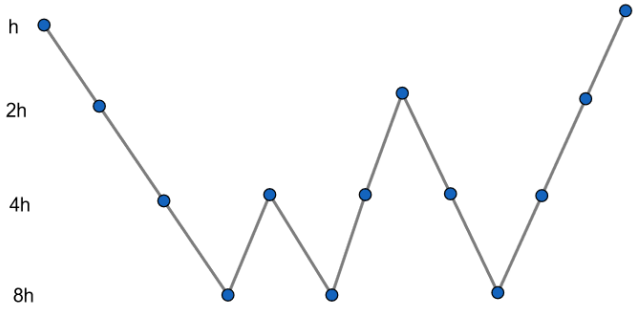


Figure 2: F-Cycle

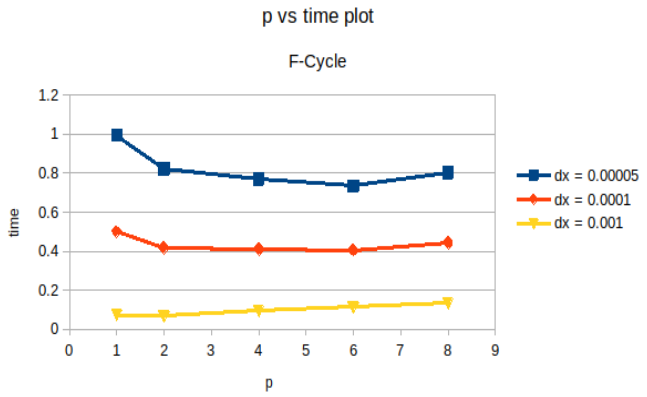
PARALLELIZATION

OpenMP directives are used to parallelize for loops across multiple threads.To calculate the norm we are using the reduction clause which involves combining values from multiple threads to single value.

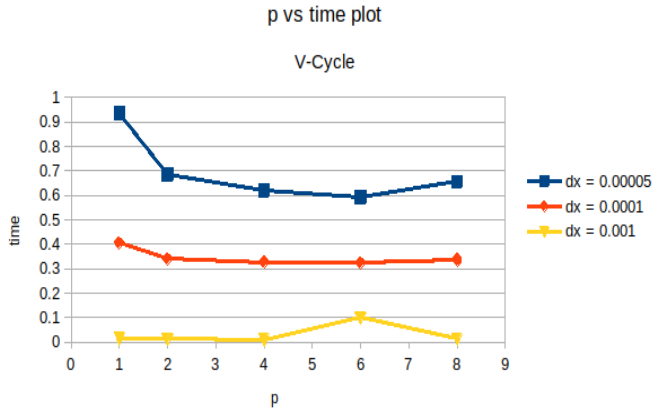
RESULTS

In this section, we present the findings from the analysis of the various AMG approaches, F-cycle and V-cycle. To compare the results of the serial version and the OpenMP parallel version, we used Number of threads vs time taken as the evaluate criteria.

- F-Cycle:

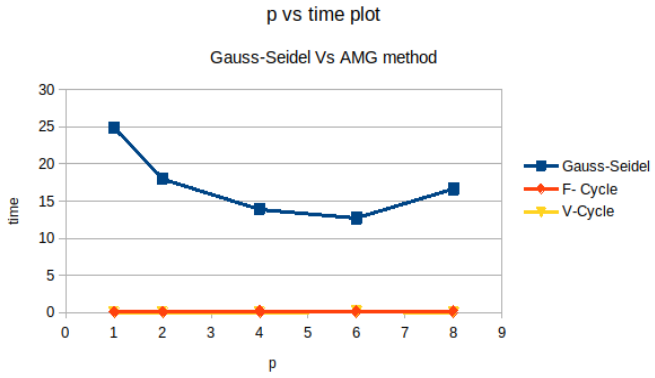


- V-Cycle:



From the graphs it can be seen that, as time taken for p = 1 (Serial Code) is comparatively larger than the values at p = (2, 4, 6, 8). In the graph we can also see that usually p = 8 is taking more time than p = 6, that is because the laptop used for computation is having only 6 cores.

- Comparison with Gauss-Seidel Method: From the graph below we can see that Gauss-Seidel method is taking much longer time to converge in comparison with the AMG method. Gauss seidel takes more time to converge because more iterations are spent on to reduce low frequency(smooth) errors. Whereas multigrid methods efficiently solve this problem by restricting the grid to a coarser thereby getting a high frequency error (rough) which can be easily reduced and interpolate back.



CONCLUSION

In this course assignment, we explored how to solve Poisson's Equation using the Algebraic Multigrid (AMG) approach and compared the results with the Gauss-Seidel (GS) method. The primary focus was to assess the effectiveness of parallelizing the code using OpenMP.

Our findings clearly demonstrate the benefits of parallelization. By increasing the number of threads (p = 2, 4, 6, 8), we observed a consistent decrease in the time taken to run the code. This indicates that the OpenMP parallelization significantly improves computational efficiency compared to the serial version of the code. Additionally, the comparison between the AMG and GS methods revealed that the AMG method consistently outperforms the GS method in terms of execution time.

In conclusion, the project successfully showed that parallelizing numerical methods using OpenMP can lead to substantial performance gains. The AMG method, in particular, proves to be highly efficient for solving Poisson's Equation, especially when implemented in a parallel computing environment. These results highlight the importance and effectiveness of parallel computing in accelerating complex numerical computations.

ACKNOWLEDGMENTS

Thanks to Prof. Kameswararao Anupindi.

REFERENCES

1. https://ocw.mit.edu/courses/18-086-mathematical-methods-for-engineers-ii-spring-2006/c4a6bca78524cb0d0678e0a0030f75e5_am63.pdf
2. Algebraic multigrid methods, Jinchao Xu and Ludmil Zikatanov