Final Project: A Parallel Multigrid Poisson Solver

Thomas D. Economon, Juan J. Alonso
CME 342, Spring 2013-2014
Stanford University

1 Project Overview

The overall goals of this project are to parallelize an existing serial code (C/C++) for a multigrid poisson equation solver using MPI and to study the performance and scalability of the resulting implementation. This will require the parallelization of two key components in the solver:

1. classical iterative methods
2. geometric multigrid

The classical iterative methods considered are the Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR) methods. The parallelization of the geometric multigrid will require the partitioning of nested coarse mesh levels with the appropriate halo layers and communication schedules. With each stage of the implementation, the performance and scalability of the solver will be assessed.

2 Problem Description

We are interested in solving the Poisson equation, which can be expressed in a domain $\Omega \subset \mathbb{R}^n$ with a boundary $\partial\Omega$ (see Fig. 1) as

$$\begin{cases} -a\nabla^2\phi = f & \text{in } \Omega \\ \phi = g & \text{on } \partial\Omega \end{cases} \tag{1}$$

where a is a known constant, $\nabla^2 = \nabla \cdot \nabla(\cdot)$ is the Laplacian operator, $\phi = \phi(\vec{x})$ is our scalar variable that is a function of space, and $f = f(\vec{x})$ is a forcing function. The second line of (1) represents a Dirichlet condition on the boundary $\partial\Omega$, where $g = g(\vec{x})$ is a prescribed function along the boundary.

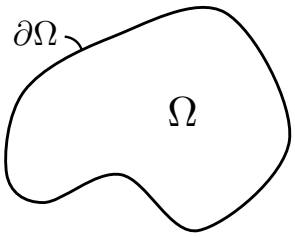

Figure 1: Schematic of the mathematical domain.

2.1 Discretization

Let Ω represent a 2D square domain given by $[0,1]$ x $[0,1]$. We choose to discretize (1) using a 2nd-order finite difference approximation on a cartesian mesh composed of a number N nodes in the x- and y-directions with uniform spacing h, as seen in Fig. 2. The value of ϕ on the 2D cartesian mesh can then be approximated for each node $\{i,j\}$ in the interior of the computational domain as

$$-\frac{a}{h^2}\left(\phi_{i,j-1} + \phi_{i-1,j} - 4\,\phi_{i,j} + \phi_{i+1,j} + \phi_{i,j+1}\right) = f_{i,j}, \quad i,j = 2,...,N-1, \tag{2}$$

where the subscripts i and j represent the indices of the current node in the computational domain for the x- and y-directions, respectively. Assuming a Dirichlet boundary condition specifies the values for ϕ on the boundary of the domain, such as that given in (1), we can consider the values for $\phi_{1,j}$, $\phi_{N,j}$, $\phi_{i,1}$, and $\phi_{i,N}$ known. As written in Eqn. 2, the result is a system of $(N-2)$ x $(N-2)$ linear equations for the unknown values of $\phi_{i,j}$ in the interior of the domain.

2.2 Forming the Linear System

We have freedom in deciding how to organize and ultimately solve the system of linear equations given by (2). Given a natural ordering of the unknowns and the 5-point stencil in Eqn. 2, we can express the full linear system:

$$A\,\Phi = \mathbf{f}, \tag{3}$$

where the matrix A is banded as a result of the structured mesh discretization and is given by

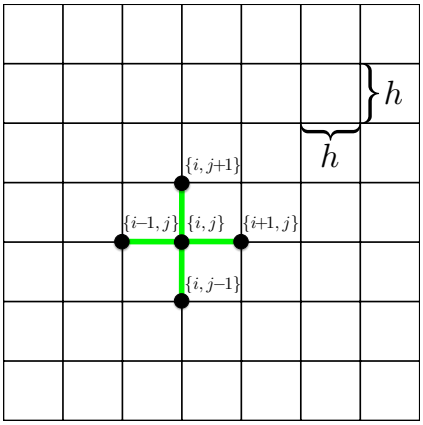

Figure 2: 2D uniform mesh featuring a discretization with a 5-point stencil.

$$A = \frac{a}{h^2}\begin{pmatrix} 4 & -1 & & & -1 & & \\ -1 & 4 & -1 & & & -1 & \\ & \ddots & \ddots & & & & \ddots \\ & & & & & & \\ -1 & & & & 4 & -1 & \\ & -1 & & & -1 & 4 & -1 \\ & & \ddots & & & \ddots & \ddots \end{pmatrix}, \tag{4}$$

with Φ and $\mathbf{f}$ being the vectors of unknowns and source terms in a natural ordering, respectively. Note that the discretization of the Poisson equation on a general unstructured mesh would result in a sparse matrix for A when the system in Eqn. 4 is formed. However, for this project, we are taking advantage of the structure of the matrix and will not explicitly form the matrix A in our calculations.

2.3 Classical Iterative Methods

At this point, a number of methods can be employed for solving the linear system in Eqn. 3, but we will focus on the classical iterative methods in this project. The Gauss-Seidel will serve as an example in the discussion below. Starting with a linear system

$$A\,\mathbf{x} = \mathbf{b}, \tag{5}$$

where

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & & \\ \vdots & & \ddots & \\ & & & \\ a_{N,1} & & & a_{N,N} \end{pmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_N \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_N \end{bmatrix}, \tag{6}$$

we can formulate the classical iterative methods by first splitting the matrix A into lower triangular, diagonal, and upper triangular parts and then rearranging terms to form point updates for the unknowns. For example, we can write the update for an unknown using the Gauss-Seidel method as follows:

$$x_i^{n+1} = \frac{1}{a_{i,i}}\left(b_i - \sum_{j=1}^{i-1} a_{i,j}\,x_j^{n+1} - \sum_{k=i+1}^{N} a_{i,k}\,x_k^{n}\right), \tag{7}$$

where n represents the iteration index, i is the row index, and j is the column index. Therefore, as we sweep through the unknowns, the entries below the diagonal in A are multiplied by x values that have already been

updated. The point updates for the Jacobi and SOR methods are similar in nature and can be found in the lecture notes. The system is typically solved using a number of sweeps until a residual tolerance is reached.

Applying Eqn. 7 to our linear system for solving the discrete Poisson equation in Eqn. 3 with matrix A given by Eqn. 4, we find that the Gauss-Seidel update for ϕ at a node $\{i, j\}$ is

$$\phi_{i,j}^{n+1} = \frac{1}{4}\left(\phi_{i,j-1}^{n+1} + \phi_{i-1,j}^{n+1} + \phi_{i+1,j}^{n} + \phi_{i,j+1}^{n} + \frac{h^2}{a} f_{i,j}\right). \tag{8}$$

2.4 Geometric Multigrid

For typical iterative numerical solution methods, high-frequency (local) errors in the solution are well-damped, while lower frequency (global) errors are poorly damped. Therefore, the low-frequency errors are difficult to eliminate, which leads to slower solver convergence, especially on fine meshes. The key idea behind multigrid is that effective rates of convergence at all scales can be maintained in a solver by leveraging a sequence of grids at various resolutions. With geometric multigrid, multiple levels of physical grids with varying resolution are used to provide better approximations of the solution with each step of an iterative solution method (i.e., a *multigrid cycle*).

To illustrate the basic components of linear multigrid for elliptic problems, define the error in the solution to be the difference between the solution Φ and the approximation to the solution $\tilde{\Phi}$, or

$$\mathbf{e} = \Phi - \tilde{\Phi}, \tag{9}$$

where $\mathbf{e}$ is the error vector (one value per node in the computational mesh). We can also define a residual vector $\mathbf{r}$, which is a measure of how well the discretized governing equations are being satisfied by our numerical solution procedure, as

$$\mathbf{r} = \mathbf{f} - A\,\tilde{\Phi}. \tag{10}$$

Introducing Eqn. 9 into our original system in Eqn. 3 gives

$$A\left(\tilde{\Phi} + \mathbf{e}\right) = \mathbf{f}, \tag{11}$$

and by introducing Eqn. 10, we recover the following expression:

$$A\,\mathbf{e} = \mathbf{r}, \tag{12}$$

which relates the error in the solution to the residual. Eqn. 12 allows us to compute a measure of the error on coarser mesh levels after transferring the values of the residual from the fine mesh level onto the coarse level (*restriction*). After calculating $\mathbf{e}$ on a coarse level, we can form a correction to the solution on the fine mesh as

$$\Phi = \tilde{\Phi} + \mathbf{e}, \tag{13}$$

upon transferring the error up to a fine mesh from the coarse mesh level below (*prolongation*). Furthermore, we can apply these ideas recursively over an entire set of grids of various resolutions to complete a full multigrid cycle, such as the V-cycle detailed in Alg. 1.

During a multigrid V-cycle, the solution is first approximated using several smoothing iterations with a method like Gauss-Seidel on the finest mesh (pre-smoothing), and then the residual is transferred to the first coarse level, where additional smoothing iterations occur. This restriction followed by smoothing continues recursively until the coarsest mesh level is reached (the downstroke of the cycle). After performing some smoothing iterations on the coarsest level, a correction for the solution values is transferred to the finer mesh level above. This upward stroke of the cycle with prolongation and smoothing continues until a correction is finally applied to the solution on the finest mesh. Typically, several final smoothing iterations (post-smoothing) are performed on the finest mesh before moving on to the next multigrid cycle. The downstroke and upstroke of the cycle form a V-shape when viewed graphically, as in Fig. 3. Other cycles are possible, and W-cycles are common, for instance.

Algorithm 1 Multigrid V-Cycle

procedure MULTIGRID_CYCLE($\Phi, \mathbf{f}, l$) ▷ l is the current mesh level
  $\Phi^l \leftarrow$ GAUSS_SEIDEL($\Phi^l, \mathbf{f}^l$) ▷ Pre-smoothing of the solution
  if $l < n\_levels$ **then**
    $\Phi^{l+1} \leftarrow 0$
    $\mathbf{f}^{l+1} \leftarrow$ RESTRICT($\Phi^l, \mathbf{f}^l$)
    MULTIGRID_CYCLE($\Phi, \mathbf{f}, l+1$) ▷ Recursive call
    $\Phi^l \leftarrow$ PROLONGATE($\Phi^{l+1}, \mathbf{f}^{l+1}$)
  $\Phi^l \leftarrow$ GAUSS_SEIDEL($\Phi^l, \mathbf{f}^l$) ▷ Post-smoothing of the solution

2.4.1 Mesh Coarsening

As we have chosen a simple, cartesian discretization of the domain Ω, the fine mesh can be coarsened repeatedly in a straightforward manner by removing every other node in the i and j dimensions. In 1D, this results in a reduction in the degrees of freedom by a factor of 2 on each coarser level. In 2D, the reduction is by a factor of 4 with each coarser level, and in 3D, the reduction factor is 8. Therefore, with every coarsening, the number of degrees of freedom reduces by a factor of 2^d, where d is the physical dimension of the problem. In our 2D example problem with a vertex-based scheme, the mesh will be automatically coarsened until 3 nodes remain in both the i and j dimensions (or until the mesh dimension is no longer evenly divisible), which will allow for our 5-point stencil to be computed for a single central node. An example of the nested mesh levels can be seen in Fig. 3.

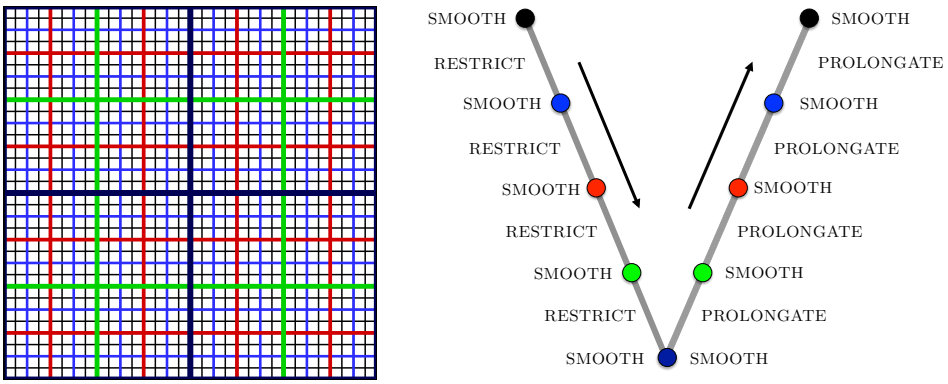

Figure 3: Representation of the nested mesh levels and corresponding multigrid V-cycle.

2.4.2 Restriction Operator

For our discretized Poisson problem, we can express the value of the residual $\mathbf{r}$ from Eqn. 10 at each node as

$$r_{i,j} = \frac{h^2}{a} f_{i,j} + \phi_{i,j-1} + \phi_{i-1,j} - 4\,\phi_{i,j} + \phi_{i+1,j} + \phi_{i,j+1}, \tag{14}$$

which can be seen as a rearrangement of Eqn. 8.

After computing $\mathbf{r}$, we restrict the values down to the next coarse level to form the right-hand side of Eqn. 12. For a weighted restriction, we will include information from all of the fine nodes that surround a particular coarse mesh node. To accomplish this, we will set the residual at a coarse mesh node to be the sum of a contribution from the coincident node (1/4 of the value), the nodes that are part of the stencil in

the north, south, east, and west directions (1/8 of the value), and the diagonal neighbors, i.e., north-east, north-west, south-east, and south-west (1/16 of the value).

2.4.3 Prolongation Operator

A prolongation operation is one that transfers the correction from a coarse mesh to a fine mesh. Similar to the weighted method for restriction, we will perform a weighted prolongation by setting the correction at a fine mesh node to be the value of the correction at a coincident coarse node, if applicable, or as the sum of a contribution from the coarse nodes that are nearest in the north, south, east, and west directions (1/2 of the value) and the nearest diagonal neighbors, i.e., north-east, north-west, south-east, and south-west (1/4 of the value).

2.5 Test Problem

A simple 2D test problem with a known solution has been chosen for exploring the performance and scalability of the multigrid Poisson solver. For the Poisson system in (1) on our square domain Ω given by $[0, 1]$ x $[0, 1]$, we select $a = 1$ and a forcing term of

$$f = -5.0 \exp(x) \exp(-2.0\, y). \tag{15}$$

The resulting boundary value problem has a solution in the domain Ω given by

$$\phi = \exp(x) \exp(-2.0\, y), \tag{16}$$

which is imposing as a Dirichlet condition on the boundaries $\partial\Omega$. The exact solution for ϕ is presented in Fig. 4.

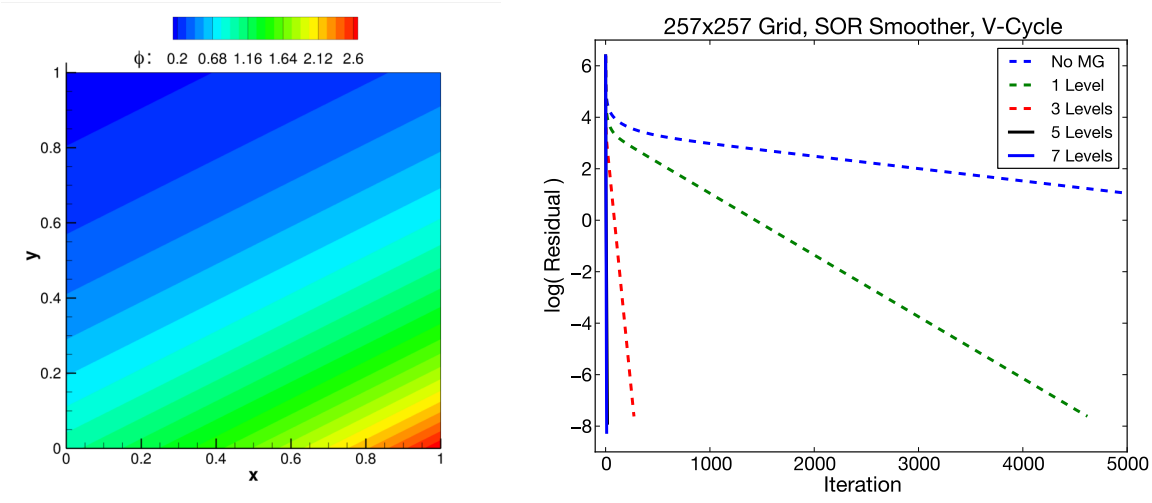

Figure 4: The known solution for ϕ for the chosen test problem (left) and examples of multigrid performance (right).

In the provided serial code, the values of the solution ϕ are initialized to the known solution on the boundaries of the domain $\partial\Omega$, while the value of the solution is taken as zero for all interior nodes. The iterative solution process continues until a specified level of convergence is obtained. An $\mathcal{L}^2$-norm of the residual $\mathbf{r}$ is taken as a convenient measure, and the solver iterates until the initial residual value is reduced by a prescribed number of orders of magnitude. Examples of the solver convergence for different numbers of multigrid levels can be seen in Fig. 4.

3 Project Deliverables

The deliverables for the final project are divided into three main categories: analyzing the performance of the serial solver, parallelization and assessment of the iterative linear solvers, and the parallelization and assessment of the geometric multigrid. Assume a convergence criteria of 14 orders reduction in the magnitude of the residual $\mathcal{L}^2$-norm with the given test problem for all studies below, unless specifically stated otherwise. Submit your code and a writeup that addresses all of the questions posed below.

3.1 Serial Performance

For a small problem ($N = 33$), answer the following questions for each of the serial iterative solvers (Jacobi, Gauss-Seidel, and SOR):

1. How many iterations are required to converge the test problem without multigrid? What is the serial execution time?
2. How many iterations are required to converge the test problem with multigrid? What is the serial execution time?

As the size of the problem increases ($N = 65, 129, 257$, etc.), what happens to the convergence behavior for the different iterative solvers both with and without multigrid?

3.2 Parallel Iterative Methods

For this portion of the project, disable the multigrid feature in the solver and assume that the grid is always partitioned among p processors in both the x- and y-directions, resulting in a total of $p * p$ processors in a given calculation. Complete the following:

1. Implement the Jacobi method in parallel with MPI. Verify the correctness of your implementation by matching the residual values reported in the serial version, and provide this information.
2. Implement parallel Gauss-Seidel and SOR methods using a red-black ordering of the unknowns with MPI. Verify the correctness of your implementation by matching the residual values reported in the serial version, and provide this information.
3. For $N = 257$ and a convergence criteria of 4 orders reduction in the magnitude of the residual $\mathcal{L}^2$-norm, compare and discuss the overall performance (in terms of iterations to convergence and wall-clock time) and scalability of the parallelized Jacobi, Gauss-Seidel, and SOR methods.

3.3 Parallel Multigrid

For simplicity, assume again that the domain will be partitioned among p processors in both the x- and y-directions and also that the number of nodes in the mesh will remain evenly divisible for multigrid (i.e., $N = 2^m + 1$ with a choice of m). Choose one of the three iterative smoothers and complete the following:

1. Construct the necessary halo layers and communication schedules for all coarse levels and reactivate multigrid. Verify the correctness of your implementation by matching the residual values reported in the serial version, and provide this information.
2. For $N = 1025$, assess and discuss the performance and scalability of the parallel multigrid Poisson solver as compared to the serial version for your chosen smoother.
3. (**10 % extra credit**) Assess the impact of reducing the number of communications on the coarser mesh levels on the overall performance of the parallel multigrid solver. Report your findings.