APMA 2822B Homework 3 Report

Yash Agrawal

October 31, 2025

Math Setup

Assume that $\Delta x = \Delta y = h$.

Using the second derivative approximations we were given, we have

$$f(x,y) = \frac{u[i-1,j] + u[i+1,j] + u[i,j-1] + u[i,j+1] - 4u[i,j]}{h^2}$$

Substituting in the given $f(x,y)$, we get

$$\begin{aligned} -8\pi^2 \sin(2\pi x)\cos(2\pi y) &= \frac{u[i-1,j] + u[i+1,j] + u[i,j-1] + u[i,j+1] - 4u[i,j]}{h^2} \\ -8\pi^2 h^2 \sin(2\pi x)\cos(2\pi y) &= u[i-1,j] + u[i+1,j] + u[i,j-1] + u[i,j+1] - 4u[i,j] \end{aligned}$$

Solving for $u[i,j]$, we have

$$\begin{aligned} 4u[i,j] &= u[i-1,j] + u[i+1,j] + u[i,j-1] + u[i,j+1] + 8\pi^2 h^2 \sin(2\pi x)\cos(2\pi y) \\ u[i,j] &= \frac{1}{4}\left(u[i-1,j] + u[i+1,j] + u[i,j-1] + u[i,j+1] + 8\pi^2 h^2 \sin(2\pi x)\cos(2\pi y)\right) \end{aligned}$$

We can use this equation to iteratively solve for u with the given $f(x,y)$ using the Jacobi method.

We will consider our solution to have converged when the maximum residual in u across all grid points is less than a specified tolerance ϵ.

Results

I implemented three versions of the Jacobi iterative solver. One runs sequentially, one uses OpenMP to parallelize using shared memory, and one uses OpenMPI to parallelize using distributed memory. The OpenMP implementation uses 8 threads, and the OpenMPI implementation uses 8 processes.

All versions produce correct and also exactly identical results at every iteration. They converge when the max residual is less than 0.001. The performance results are summarized below, where N is the size of the array used.

We can clearly observe that both of parallelism provided massive performance boosts. The OpenMP version performed better than the OpenMPI version by a significant margin.

Version	N	Time (s)	GB/s
Sequential	256	14.792	5.834
Sequential	512	245.958	5.641
OpenMP	256	1.706	50.619
OpenMP	512	23.936	57.967
OpenMPI	256	2.280	37.877
OpenMPI	512	37.588	36.913

Roofline Model

Memory Calculations

To generate a roofline model, we first need to calculate the memory usage for our separate operations.

We use double precision floating point numbers for all calculations, which are each 8 bytes. Let the size of the matrix be $N \times N$.

For the loop in which we update our solution, we must read the u matrix and write the new u matrix. This means the number of bytes we access for updating out matrix is

$$2 * 8 * N^2 = 16N^2$$

For the loop in which we calculate our max residual or convergence error, we must read the u matrix. This gives the number of bytes accessed as

$$8 * N^2 = 8N^2$$

Note we are ignoring memory accesses from precomputing values of f since this is not strictly required, however including them may bring out observed results closer to the theoretical roofline.