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.

I also precompute the values of $f(x,y)$ and store them in an array to avoid recomputing them at every iteration. All my calculations will account for this precomputation in terms of the memory accesses and FLOPs.

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.

Version	N	Time (s)	GB/s	Iterations
Sequential	256	14.792	5.834	54,917
Sequential	512	245.958	5.641	220,536
Sequential	1024	-	-	-
OpenMP	256	1.706	50.619	54,917
OpenMP	512	23.936	57.967	220,536
OpenMP	1024	372.371	59.734	883,877
OpenMPI	256	2.280	37.877	54,917
OpenMPI	512	37.588	36.913	220,536
OpenMPI	1024	573.528	38.784	883,877

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.

Roofline Model

FLOPs

To generate a roofline model, we first need to calculate the number of floating point operations (FLOPs) for our separate operations.

For the loop in which we update our solution, we have the following operations for each grid point:

- 4 additions to sum the neighboring u values
- 1 multiplication to compute $8\pi^2 h^2$
- 2 multiplications to compute $\sin(2\pi x)$ and $\cos(2\pi y)$
- 1 multiplication to multiply $8\pi^2 h^2$ with $\sin(2\pi x)\cos(2\pi y)$
- 1 addition to add the result to the sum of neighboring u values
- 1 division to divide by 4

Memory Accesses

We also 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, write the new u matrix, and read the f values matrix. This means the number of bytes we access for updating out matrix is

$$2 * 8 * N^2 + 8 * N^2 = 24N^2$$

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

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