Computer Science 335
Parallel Processing and HPC
Siena College
Fall 2024

Programming Project 2: Jacobi Iteration in C

Due: 4:00 PM, Monday, September 23, 2024

In this programming project, you will repeat the first programming project, but this time you will write the code in C.

You may work alone or with a partner or two on this programming project. AI assistance is permitted, but must be documented and intermediate steps in the process must be committed to your repository.

Learning goals:

1. To gain experience using C for a scientific computation.
2. To use a plotting program to visualize simulation results.

Getting Set Up

In Canvas, you will find a link to follow to set up your GitHub repository, which will be named `jacobi-c-proj-yourgitname`, for this programming project. Only one member of the group should follow the link to set up the repository on GitHub, then others should request a link to be granted write access.

All GitHub repositories must be created with all group members having write access and all group member names specified in the `README.md` file by 4:00 PM, Thursday, September 19, 2024. This applies to those who choose to work alone as well!

Jacobi Iteration in C

The problem is simple: convert the Java version of Jacobi iteration into C. Same requirements otherwise, with the exception of a different output format described below.

Please make use of the provided `Makefile` as you develop your program, and be sure it works to compile your code when you submit. Note that you might need to change the `CC=` line to point to the compiler you are using.

You are welcome to use or ignore the `timer.h` and `timer.c` files. If you want to use the function defined in `timer.c`, you should include `timer.h` in the C source file that uses it. A skeleton of the `jacobi.c` file is also provided.

If you end up using any functions from `math.h`, such as `fabs`, which returns the absolute value of a floating-point number, you will have to add the `-lm` flag to the end of your link line in the `Makefile` to bring in the standard math library.

Solution Output for Visualization

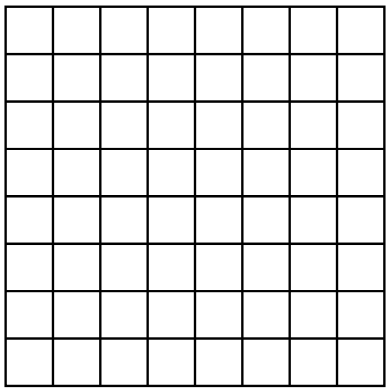
For this version, we will also change the solution output format to make it easier to plot a visualization of the results.

Recall that we are computing solution approximations at points on a regular grid in the computational domain. For an 8×8 grid, it looks like this:

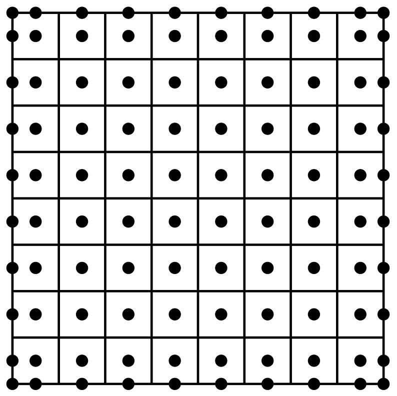
```
* * * * * * * * * *
* . . . . . . . . *
* . . . . . . . . *
* . . . . . . . . *
* . . . . . . . . *
* . . . . . . . . *
* . . . . . . . . *
* . . . . . . . . *
* . . . . . . . . *
* * * * * * * * * *
```

where the $\star$ characters represent boundary condition points and `.` characters represent solution points. Recall our computational domain goes from 0 to 1 in both dimensions.

Really what we are doing here is covering the computational domain with a mesh structure made up of 64 squares. We approximate the solution by assuming that there is a constant temperature throughout each square. If we have more squares (*i.e.*, a larger mesh made of up smaller squares), we can obtain a more accurate approximation of the solution.



None of this changes how our computation proceeds. There was nothing in the computational loop that knows or cares about actual coordinates. However, we can use this information to output our solutions in a format more suitable for plotting. To do this, we will output the solution values for each square, along with the coordinates of the center of that square. Also, we will want to plot points along the boundary. So in this case, we would output 100 points representing values at these points:



Assuming a computational domain of (0,0) to (1,1), you will need to compute the coordinates from the row/column values when you output. You can then plot the resulting solution data using (for example) Gnuplot. Gnuplot scripts are available in the

`SienaCSISParallelProcessing/JacobiPlotting`

repository. Include screen shots or other graphics files showing your visualizations for at least 3 different size grids in your repository.

Here's a sample output file generated with this command line from my reference solution:

```
./jacobi 50 1000 .00001 solution.dat
```

Bonus Opportunity

Gnuplot is a fine tool, but it's also a bit old fashioned. For up to 5 bonus points, create and demonstrate a different, ideally better, visualization of the output file format here using mode modern tools.

Submission

Commit and push!

Grading

This assignment will be graded out of 50 points.

Feature	Value	Score
Command-line parameters	3	
Grid allocation and deallocation	5	
Jacobi iterations	5	
Stop based on iteration count	2	
Stop based on error tolerance	3	
Print simulation stats	3	
Output solution to terminal	2	
Output solution in new format to file	12	
Solution plots	5	
Documentation	5	
Style	5	
Bonus (up to 5)	0	
Total	50	