Writeup: Short introduction / summary of the overall project

The goal of this part of the project was to set up and solve the steady-state heat equation for a certain geometry, in the context of fluid transfer. Specifically we studied a system where hot fluid is flowing within a pipe, and cool jets on the exterior of the pipe spray cold air to keep the outer surface cool. We were given a model to envision the geometry of a certain (periodic) section of the pipe wall and asked to setup and solve the discretized heat equation for the modeled, rectangular grid of points.

We used the Conjugate Gradient Descent method to solve the system of equations. The design for the matrix was a class called SparseMatrix, and the solver was implemented and called from another file. The following is a description of the methods in this class:

- AddEntry–Appends row, column, and value entries to a sparse matrix in COO format
- ConvertToCSR–In-place conversion of a COO-formatted matrix to CSR
- getRow–Returns the current state of the row pointer vector
- getCol–Returns the current state of the column pointer vector
- getVal–Returns the current state of the value pointer vector

The solver matrix was then setup and solved in another class called HeatEquation2D, whose methods are as follows:

- Setup–Takes a string containing the name of the file specifying key information mentioned in the handout and constructs the matrix in COO format
- Solve–Takes the name of a solution file prefix and outputs the solution given by conjugate gradient during the first and last iterations, as well as every 10 iterations in a specified format as given in the handout (solution000, solution010, etc.)

I implemented the algorithm in another file CGSolver.cpp, the psuedocode of which is below:

CGSolver.cpp invokes functions from matvecops.cpp, which contains a variety of common functions for working with matrix-vector products. They are outlined below:

- L2norm–returns the L2norm of a vector
- dot–returns the scalar product of two vectors
- scalMult–returns the scalar product αv of a double precision scalar α with a vector v.
- vecAdd–returns the vector addition of two vectors v_1 and v_2.

Algorithm 1 Conjugate Gradient Method

initialize u_0
$r_0 = b - Au_0$
L2normr0 = L2norm(r_0)
$p_0 = r_0$
niter = 0
while niter < nitermax **do**
  niter = niter +1
  alpha$_n$ = $(r_n^T r_n)/(p_n^T A p_n)$
  $u_{n+1} = u_n +$ alpha$_n p_n$
  $r_{n+1} = r_n -$ alpha$_n A p_n$
  L2normr = L2norm(r_{n+1})
  if L2normr/L2normr0 < threshold **then**
    break
  end if
  beta$_n$ = $(r_{n+1}^T r_{n+1})/(r_n^T r_n)$
  $p_{n+1} = r_{n+1} +$ beta$_n p_n$
end while

- MatMult–returns the matrix product of Av, where A is given in CSR format.

This eliminated redundant code because each operation above was used more than once. For matrix-vector multiplication for example, we first calculate Au_0, and then Ap_n a couple times. Each of these becomes one function call, as opposed to writing out the function twice, which results in cluttered code.

Users Guide:

The code is compiled in a makefile document. Simply run make and the executable "main" is generated from all the source code (in particular, heat.cpp, sparse.cpp, CGSolver.cpp, matvecops.cpp, and COO2CSR.cpp). To run main, the name of the input file (input1.txt, input2.txt. etc.) and prefix for the solution files needs to be provided as command line arguments:
See the following command :

```
$ ./main inputxx.txt soln
```

as an example. The postprocessing code is contained in the file postprocess.py, with the name of the input file and specific solution file to process and visualize specified as command line arguments. An example is below:

```
$ python3 postprocess.py inputxx.txt solnxx.py
```

This then reports to console the input file processed and the mean temperature– I chose not to include the boundary points for simplicity, and the mean temperature reported reflects that choice.

A heat map titled plot.png is generated in the same directory as the file: To view this simply secure copy the file to a home directory. An example heat map

for input2.txt is below, with the x axis the number of columns of matrix and the y the number of rows (so the color gives an idea for the value of $A[i][j]$, A the aforementioned heat matrix).

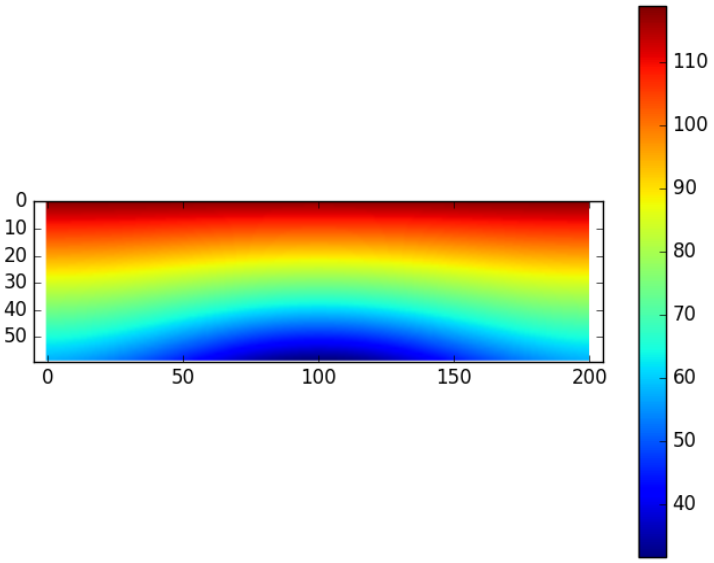

Figure 1: Example heat map using input2.txt

***Note: Using final 2 late days