CME 211 Final Project

December 10, 2019

1 Introduction

In this project [1][2], I have constructed a set of C++ objects and files which solve the steady state heat equation across a 2D pipe wall, given an input file containing the [(length, width, spacing); (Tc, Th)] values for the discretized grid and the temperature boundary conditions. The executable "main" loads the system of equations as a "Sparse" matrix object, solves the temperature profile along this grid using the conjugate gradient method, and outputs solution files every 10 iterations. The python file "postprocess.py" analyzes the final solution file, calculating the mean temperature in the pipe and plotting a pseudocolor plot of the temperature distribution.

2 CGSolver Implementation

The CG solver is called via the "Solve" method of the "HeatEquation2D" class. The "CGSolver" method takes in the matrix A (from the grid data), vector b (boundary conditions), vector x (initial guess of all ones), a tolerance of 0.00001, and the solution prefix for output files. Because A is negative definite, all values in the A and b used in the solver are multiplied by (-1). The pseudocode for the conjugate gradient method used is found below.

Initialize u as x;
r = b − A u;
L2normr0 = L2norm(r);
p = r;
niter = 0;
while *niter* < *1000* **do**
  increment niter;
  alpha = $(r_n^T r_n)$ / $(p_n^T A p_n)$;
  $u_{n+1} = u_n + alpha_n p_n$;
  $r_{n+1} = r_n - alpha_n A p_n$;
  L2normr = L2norm(r_{n+1});
  if *L2normr/L2normr0* < *threshold* **then**
    break;
  end
  $beta_n = (r_{n+1}^T r_{n+1})/(r_n^T r_n)$;
  $p_{n+1} = r_{n+1} + beta_n p_n$;
end

Algorithm 1: Pseudocode for the CG algorithm. [1]

Within this function, it utilizes multiple functions stored in "matvecops.cpp" and "matvecops.hpp" to reduce redundancy in terms of matrix vector operations. These functions include vector subtraction and addition, vector dot products, two norms, and scalar-vector multiplication. The function to multiply a matrix and a vector, "MulVec", is encapsulated as a method in the "Sparse" object which contains the matrix in CSR format. Solution files are output every ten iterations, plus the final converged solution, and if convergence is successful a value of 0 is returned.

3 User Guide

To run the makefile which compiles the program, type in the command line "make ./main" in the directory containing all of the files. You will see "g++ -std=c++11 -O3 -Wall -Wextra -Wconversion -Wpedantic -o main main.cpp CGSolver.cpp COO2CSR.cpp heat.cpp matvecops.cpp sparse.cpp" output. "make clean" also removes the object and executable files, if desired. To solve input files, write "./main input$< numin >$.txt solution" for each. It will output "SUCCESS: CG solver converged in $< numconv >$ iterations." and print numerous solution .txt files. To run the postprocessing code on the final solution file, write "python3 postprocess.py input$< numin >$.txt solution$< numconv >$.txt", where $< numconv >$ is formatted with leading zeros if not three digits in length. This will output "Input file processed: input$< numin >$.txt" and "Mean Temperature: $< meantemp >$", and save a color plot to a .png containing the temperature distribution and isotherm line.

4 Example Figures

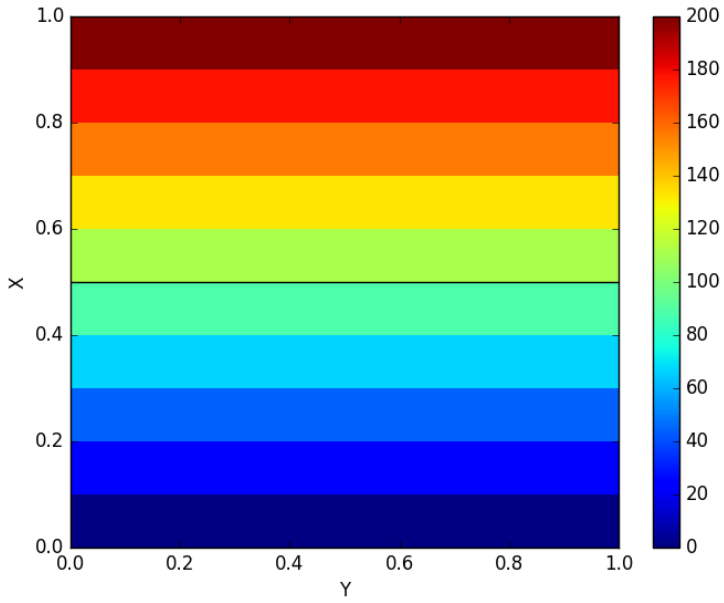

Figure 1: Temperature profile from "input0.txt", with the largest mesh size.

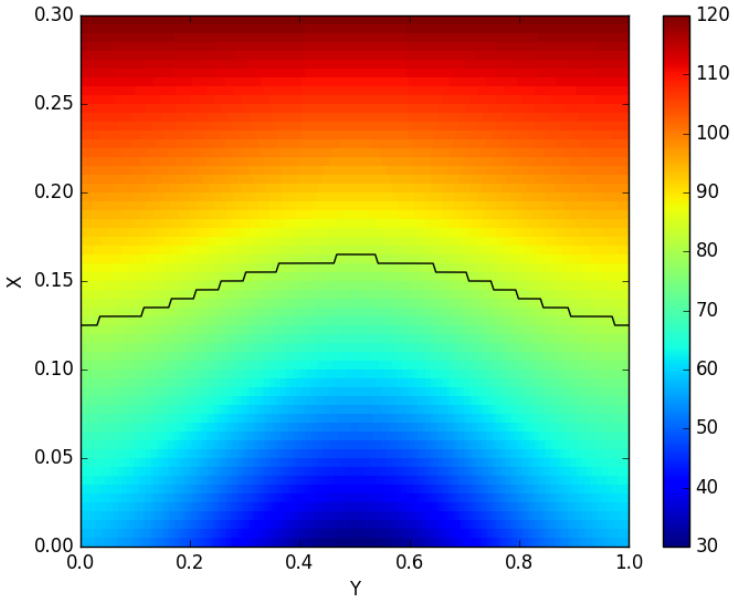

Figure 2: Temperature profile from "input2.txt", with the smallest mesh size.

References

[1] P. LeGresley. "CME 211 Final Project, Part 1", 2019

[2] P. LeGresley. "CME 211 Final Project, Part 2", 2019