Project Writeup

Bowei Ma

December 9, 2015

Overall Description of the Project

The major objective of the final project is to implement a solver for simplified steady-state heat equation. The heat transportation system is described where you are transferring some hot fluid with temperature T_h within a pipe. Moreover, in order to keep the exterior of the pipe cool, a series of cold air jets with temperature T_c are equally distributed along the pipe and continuously impinge on the pipe surface.

The goal of our solver is to determine the mean temprature within the pipe wall using a periodic portion of the pipe.Since the matrix A formed from the geometry is symmetric positive definite and rather sparse, we choose to implement a solver using Conjuagate Gradiant (CG) method to solve the equation $Ax = b$ iteratively.

Description of the CG solver implementation

The problem could be summarized as solving $Ax = b$ for some sparse matrix A (which is the representation of the underlying heat equation) using CG method. In order to efficiently utilize the OOP design concept of C++ language, we encapsulate the equation solver into two classes:

- **SparseMatrix** : A class that contains the structure of a sparse matrix. The `SparseMatrix` class implement several data members and fundamental operations (such as set the dimension of the matrix, add an non-zero entry into the matrx, etc.).

- **HeatEquation2D** : A class that contains the setup and solving procedure of the heat equation system. The `HeatEquation2D` class utilize the instance of `SparseMatrix` class, setup the system using a file that describes the geometry of the pipe, and then solve the linear system $Ax = b$ using CG iteration.

The pseudocode of the CG algorithm is decribed below:

initialize u_0;
$r_0 = b - Au_0$;
2-$norm(r0)$ = 2-$norm(r_0)$;
$p_0 = r_0$;
$niter = 0$;
while $niter < max\#iteration$ **do**
    $niter = niter + 1$;
    $\alpha_n = (r_n^T r_n)/(p_n^T A p_n)$;
    $u_{n+1} = u_n + \alpha_n p_n$;
    $r_{n+1} = r_n - \alpha_n A p_n$;
    2-$norm(r)$ = 2-$norm(r_{n+1})$;
    if 2-$norm(r)$/2-$norm(r0) < threshold$ **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 of Conjugate Gradient (CG) Algorithm

User's Guide to Execute the Program

The program of the heat equation solver is divided into two parts:

- The computation of the solver is written in C++ using OOP design. With the convienence of `makefile` , the compilation procedure is pretty straight forward. Basic usage from command line is:

  ```
  make
  ./main [input filename] [solution_prefix]
  ```

  The solution files are written for the initial guess, the last convergence solution and every 10 iterations. The names of the solution files are in the following pattern:

  `solution_prefix + number of iterations.txt`

  After the execution of the program, a meeage with convergence information would be print on the console.

  A sample output of the C++ program is given below:

  ```
  $ ./main input2.txt solution
  SUCCESS: CG solver converged in 157 iterations.
  ```

- The postprocessing and visualization of the results are written in Python file `postprocessing.py`, which would compute the mean temperature within the pipe and visualize the thermal distribution using a pseudocolor plot.

  The basic usage of the postprocessing procedure from the command line is:

  `python postprocessing.py [input filename] [solution filename]`

  A sample output of the Python program is given below:

```
Input file processed:  input2.txt
Mean Temperature:  81.8317
```

Example of Visualization of the Heat Distribution

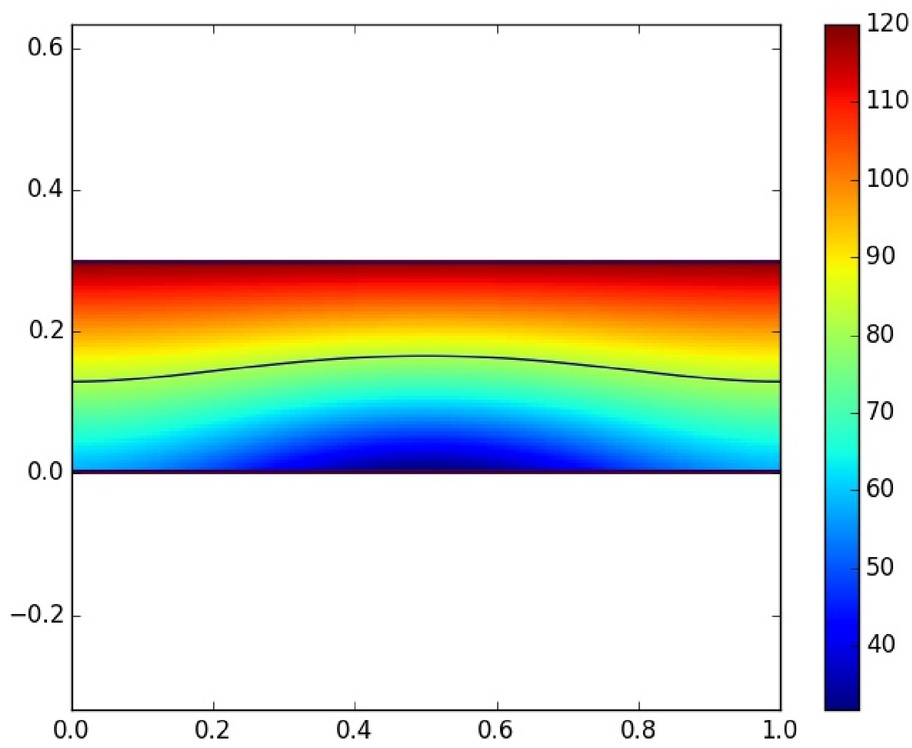

Figure 1: Example of a pseudocolor plot of input2.txt

Reference

Nick Henderson, *CME 211: Project Part 1*(2015)
Nick Henderson, *CME 211: Project Part 2*(2015)