Tetiana Parshakova

CME211 Project 2

1 Introduction

This project is related to solving the heat equation for a specific boundary conditions.

Namely, we are given a pipe with fixed bottom and top temperatures and are using the discretized heat differential equation to find the temperature of walls. To account for the pipe, when we approximate it using the rectangle, we introduce periodicity along the x axis.

Discretize the pipe using $m \times n$ grid: $m = \frac{l}{h}$, $n = \frac{w}{h}$. Working with equation

$$(-1, -1, 4, -1, -1)^T (u_{i,j-1}, u_{i-1,j}, u_{i,j}, u_{i+1,j}, u_{i,j+1}) = 0 \tag{1}$$

We flatten our solution vector u as follows:

$$u = [u_{0,0}, ..., u_{m-2,0}, u_{0,1}, ..., u_{m-2,n-3}]^T$$

Hence we can restructure equation **??** into a linear system with $(m-1)(n-2)$ unknowns, where matrix $A \in \mathbb{R}^{(m-1)(n-2)\times(m-1)(n-2)}$ contains -1's and 4's on the corresponding diagonals.

The system. has defined values at the boundaries and periodicity, hence additionally when creating $Au = b$, we have

$$\begin{aligned}
&\text{if } i = 0: \text{ replace } u_{i-1,j} \text{ in eq. ?? by } u_{m-2,j},\\
&\text{if } i = m-2: \text{ replace } u_{i+1,j} \text{ in eq. ?? by } u_{0,j},\\
&\text{if } j = 0: \text{ replace } u_{i,j-1} \text{ in eq. ?? by } T_h,\\
&\text{equivalent to setting } b[i + j * (m-2)] = T_h\\
&\text{if } j = n-3: \text{ replace } u_{i,j+1} \text{ in eq. ?? by } T(ih),\\
&\text{equivalent to setting } b[i + j * (m-2)] = T(ih)
\end{aligned}$$

Hence, we proceed by using Conjugate Gradient to find the solution of the linear system $Au = b$.

2 Implementation Details

CGSolver is a function for solving $Ax = b$, where A is in SparceMatrix object. The starting guess for the solution is provided in x, and the solver runs a maximum number of iterations equal to the size of the linear system.

Returns the number of iterations to converge the solution to the specified tolerance, or -1 if the solver did not converge.

Algorithm 1 Conjugate Gradient Pseudo-code

```
1: function CGSOLVER($A$, $u_0$, $b$, $tol$)
2:     $n\_iter = 0$
3:     $n\_maxiter = A.size(0)$                                  ▷ number of rows in $A$
4:     $r_0 = b - Au_0$
5:     $p_0 = r_0$
6:     $L2normr0 = L2norm(r_0)$
7:     while $n\_iter < n\_maxiter;\ n\_iter++$ do
8:         $\alpha = \frac{r_n^T r_n}{p_n^T A p_n}$
9:         $u_{n+1} = u_n + \alpha p_n$
10:        $r_{n+1} = r_n - \alpha A p_n$
11:        $L2normr = L2norm(r_{n+1})$
12:        if $\frac{L2normr}{L2normr0} < tol$ then
13:            break
14:        $\beta = \frac{r_{n+1}^T r_{n+1}}{r_n^T r_n}$
15:        $p_{n+1} = r_{n+1} + \beta p_n$
```

2.1 SparseMatrix

Let `SparseMatrix A` be a class instance. It contains data attributes of CSR format, i.e. `i_idx, j_idx, a` and methods for creating a matrix.

First, the matrix is stored in the COO format using the existing attributes. The population is achieved using its method `A.AddEntry(i, j, val)`.

Then we convert COO to CSR format using the method `A.ConvertToCSR()`.

The matrix multiplication that is using in the CGSolver on our matrix is achieved using another method `A.MulVec(vec)`

2.2 CGSolver

```
int CGSolver(SparseMatrix &A,
             std::vector<double> &b,
             std::vector<double> &u_n,
             double              tol,
             std::string soln_prefix)
```

implements the pseudocode ??.

2.3 HeatEquation2D

Let `HeatEquation2D sys` be a class instance. It contains data attributes of SparseMatrix and vectors x,b, i.e. `SparseMatrix A; std::vector<double> b, x` and methods for setting up the A, b, x and solving the heat equation.

First, the matrix with vectors are created as described in the introduction using the class method `sys.Setup(inputfile)`. Unknown is initialized to 0, $x = u_0 = 0$.

Then we solve the linear system by calling the method `sys.Solve(sol_prefix)`, that uses `CGSolver(this->A, this->b, this->x, tol, soln_prefix)`.

3 User guide

Run the following to create executable files, and solve heat equation and create plots.

1. `make`
2. `./main input2.txt solution`
3. `python postprocess.py input2.txt sol154.txt`

4 Results

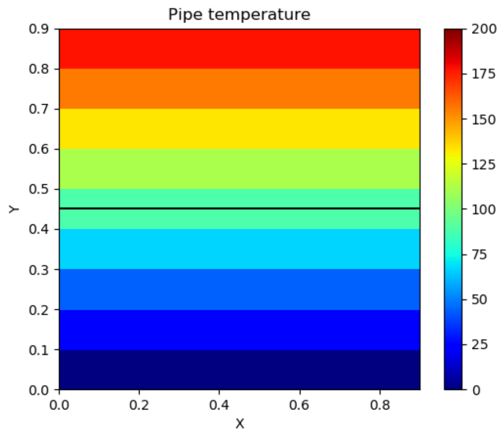

Figure 1: Temperature of pipe for `input0.txt`. Mean temperature: 100.00002

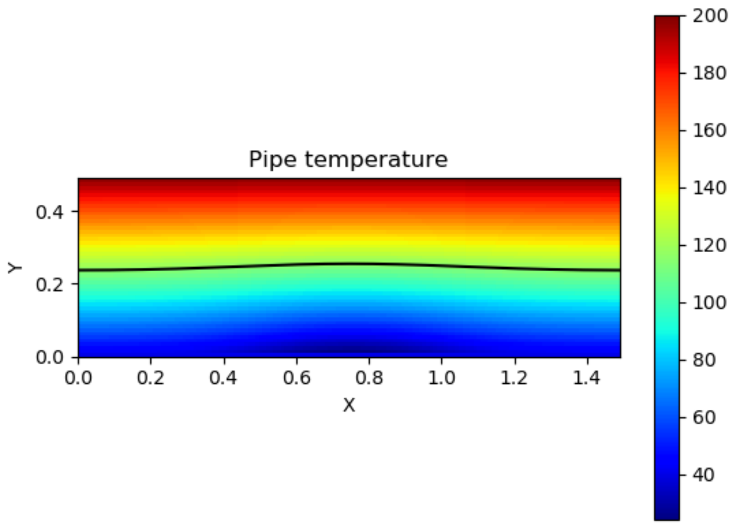

Figure 2: Temperature of pipe for `input1.txt`. Mean temperature: 116.38993

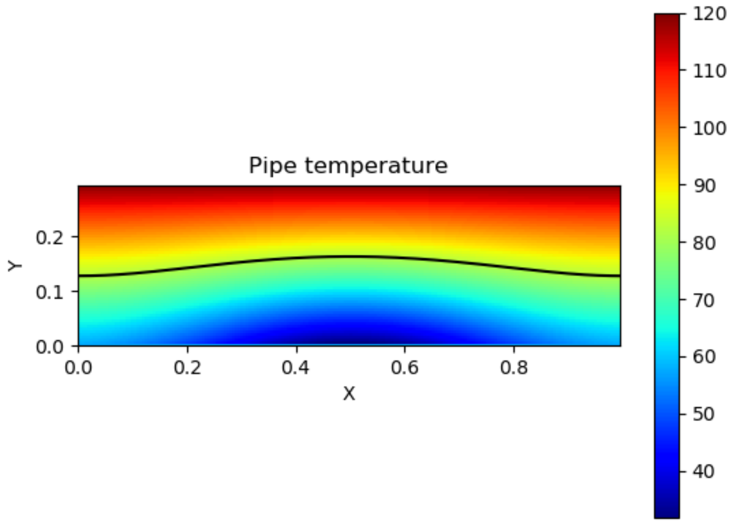

Figure 3: Temperature of pipe for `input2.txt`Mean temperature: 81.99997