CME211 Final Project

Yu Gu

December 14, 2018

1. **Introduction**
   In this project, I develop a program to solve the 2D heat equation on a simple geometry using a sparse matrix solver written in C++. First, I implement a sparse matrix solver in C++, which includes the sparse matrix class and functions for relevant operations of sparse matrices and vectors. Then I form the system of heat equations for a specified geometry, solving the system, and writing Python code to visualize the results.

2. **Pseudo-code for Conjugate Gradient (CG) algorithm**

Algorithm 1 CG algorithm

Initialize: u_0
$r_0 = b - Au_0$
$p_0 = r_0$
$niter = 0$
while $niter < nitermax$ **do**
  $niter = niter + 1$
  $\alpha_n = {r_n}^T r_n / {p_n}^T A p_n$
  $u_{n+1} = u_n + \alpha_n p_n$
  $r_{n+1} = r_n - \alpha_n A p_n$
  if $\|r_{n+1}\|_2 / \|r_0\|_2 < threshold$ **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

3. **Design for helper functions**
   Noticing that in this CG algorithm, we need to deal with some computations between constant, vectors and matrices. Thus, I design six methods below to help us solve these problems.

   - *matvecDot*: Dot product of matrix with CSR format matrix and vector.
   - *vecAdd*: Add two vectors.
   - *vecSubtract*: Subtract two vectors.
   - *vecMul*: Multiply constant to vector.
   - *vecDot*: Dot product of two vectors.
   - *vecNorm*: 2-norm of a vector.

4. **OOP design for sparse matrix class**
   The basic idea for the sparse matrix class is to store a sparse matrix in COO format, providing methods to resize the matrix dimensions and add new entries. After inputing all entries of the matrix, you can use a method to convert the COO format into CSR format. There are 5 private members which define the detail of a sparse matrix. *i_idx, j_idx, a* are the values of non-zero entries and their positions, which is stored in COO format first and can be converted to CSR format. *ncols, nrows* define the shape of the martix. There are 7 main methods:

- *Resize*: Method to modify sparse matrix dimensions.
- *AddEntry*: Method to add entry to matrix in COO format.
- *ConvertToCSR*: Method to convert COO matrix to CSR format using provided function.
- *vecMul*: Method to perform sparse matrix vector multiplication using CSR formatted matrix.
- *get_i_idx, get_j_idx, get_a*: Method to get the parameters of the sparse matrix.

5. **OOP design for heat equation class**
   The basic idea for the 2D heat equation class is to setup by inputing a filename which contains the basic information about the system, then solve the system and store the result solution into output file. There are 3 private members, the sparse matrix A, constant in system b and solution x.There are 5 main methods:

   - *Setup*: Method to setup as Ax=b system.
     The idea of setup is that each point in the geometry has one corresponding function in the system, and the function for each point is only related to its neighbor point in 4 directions. Here we treated points on the left and right boundaries as the periodic boundary, which means they always have the same temperature if they are in same row. Upside and downside boundaries are treated as given information, where the hot isothermal boundary (upside) is constant, and cold isothermal boundary (downside) is the inverted Gaussian temperature computed by x coordinate. Store the system in COO format and convert to CSR format later.
   - *Solver*: Method to solve system using CGsolver.
   - *T_x*: Method to compute the inverted Gaussian temperature.
   - *is_equal*: Method to judge whether two vectors are equal (whether Ax = b in this case).
   - *get_x*: Method to get current solution x.

6. **Users guide**

   - First use **make** to compile the whole program.
   - Then use **./main inputfile.txt solution**. It will show whether the CG solver converges.
   - If converged, **solution_converge.txt** will be the final result x and stored in the same directory with the program cpp file.
   - Next, use **python3 postprocess.py inputfile.txt solution_converge.txt** to implement postprocessing and visualization.

7. **Visualization for example data**

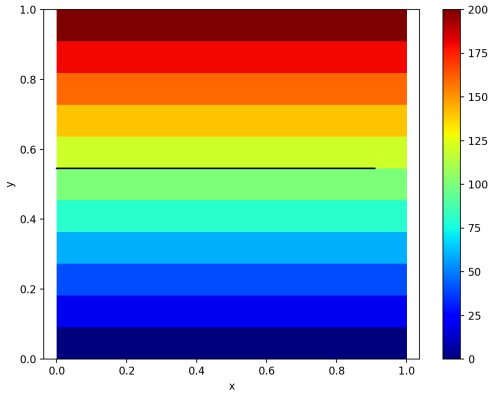

Figure 1: Temperature distribution of input0

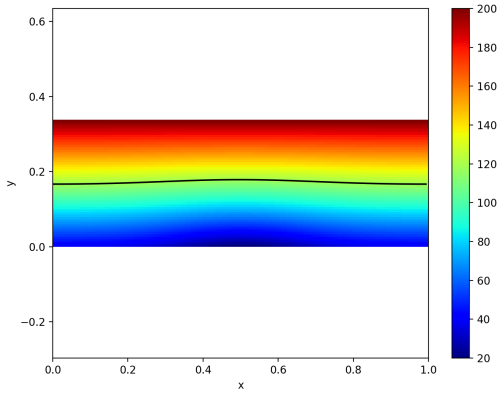

Figure 2: Temperature distribution of input1

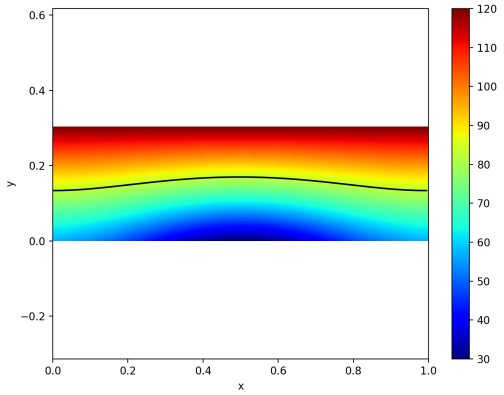

Figure 3: Temperature distribution of input2

References

[1] cme211-project-part-1.pdf

[2] cme211-project-part-2.pdf