# Project 1: 2-D Thermal Analysis

**Junjian Xie**
**junjianx@andrew.cmu.edu**

## Abstract
This report mainly discusses a 2-D thermal analysis. First, the problem is formulated mathematically into a system of linear equations based on the thermal PDE. Then, two methods, the Gaussian elimination and Cholesky factorization algorithm, are implemented to solve the problem with three different cases. In the next part, the experimental results are shown. Finally, the analysis is further discussed.

## 1. Mathematical Formulation
Heat equation is a 2nd-order linear PDE:

$$\rho \cdot C_P \cdot \frac{\partial T(x,y,t)}{\partial t} = \kappa \cdot \nabla^2 T(x,y,t) + f(x,y,t)$$

When reaching a steady state,

$$\frac{\partial T(x,y,t)}{\partial t} = 0$$

Then

$$\kappa \cdot \nabla^2 T(x,y,t) + f(x,y,t) = 0$$

For a 2-D space, it can be discretized into panels. Then for panel $(i,j)$, we have

$$\kappa \cdot \left[\frac{\partial^2 T(i,j)}{\partial x^2} + \frac{\partial^2 T(i,j)}{\partial y^2}\right] = -f(i,j)$$

$$(1 \le i \le N, 1 \le j \le M)$$

$$\begin{cases} \dfrac{\partial^2 T(i,j)}{\partial x^2} = \dfrac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{\Delta x^2} \\ \dfrac{\partial^2 T(i,j)}{\partial y^2} = \dfrac{T_{i,j+1} + T_{i,j-1} - 2T_{i,j}}{\Delta y^2} \end{cases}$$

$$\kappa \cdot \left[\frac{T_{i+1,j}+T_{i-1,j}-2T_{i,j}}{\Delta x^2} + \frac{T_{i,j+1}+T_{i,j-1}-2T_{i,j}}{\Delta y^2}\right] = -f(i,j)$$

$$\frac{2(\Delta x^2 + \Delta y^2)T_{i,j}}{\Delta x^2 \Delta y^2} - \frac{T_{i+1,j} + T_{i-1,j}}{\Delta x^2} - \frac{T_{i,j+1} + T_{i,j-1}}{\Delta y^2} = \frac{1}{\kappa} \cdot f(i,j)$$

So for $T_{i,j}$, its coefficients are equal to $2(\Delta x^2 + \Delta y^2)/(\Delta x^2 \Delta y^2)$. Its neighbors on the horizontal direction have coefficients equal to $1/\Delta x^2$. Its neighbors on the vertical direction have coefficients equal to $1/\Delta y^2$.

When the x-coordinate for $T_{i,j}$ is equal to zero or N, the point $(i,j)$ is a left or right boundary point. When the y-coordinate for $T_{i,j}$ is equal to zero or M, the point $(i,j)$ is a bottom or top boundary point.

As the temperatures on the boundary are not unknown variables, they can be moved to the right hand side. So we get a system of linear equations like

$$AX = B$$

$$X = \left[T_{1,1}, T_{1,2}, \dots, T_{N,M}\right]^T$$

$$B = \frac{1}{\kappa} \cdot F + \lambda \cdot T_c$$

$$F = \left[f_{1,1}, f_{1,2}, \dots, f_{N,M}\right]^T$$

$$\lambda = 1/\Delta x^2 \text{ or } 1/\Delta y^2$$

Because matrix F is changed from 2-D to 1-D, the index of $f(i,j)$ in the column matrix is

$$(index, 1)$$

$$index = mod(j, M = 1) + M \times [mod(i, N + 1) - 1]$$

For the first M equations, they have temperatures on the left boundary, while the last M equations have temperatures on the right boundary. For points

$$f(mod(index, M) == 0, 1)$$

they are in the top row with temperature of top boundaries. And For points

$$f(mod(index, M) == 1, 1)$$

they are in the bottom row with temperature of bottom boundaries. So the matrices can be formulated.

## 2. Linear System Solver
In the solving part, I have implemented two methods, the Gaussian elimination and Cholesky factorization algorithm.

(1) Gaussian elimination

In linear algebra, Gaussian elimination (also known as row reduction) is an algorithm for solving systems of linear equations. It is usually understood as a sequence of operations performed on the associated matrix of coefficients.[1] Let's take a simple example. Suppose that we have a set of linear equations:

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 1 \\ x_1 + 4x_2 + 9x_3 = 13 \\ x_1 + 8x_2 + 27x_3 = 31 \end{cases}$$

Then we can re-write the system in matrix forms.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 13 \\ 31 \end{bmatrix}$$

$$C = [A \quad B] = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 1 & 4 & 9 & 13 \\ 1 & 8 & 27 & 31 \end{bmatrix}$$

First, we treat $C_{11}$ as the pivot and eliminate the first element in the lower rows.

$$C = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 0 & 2 & 6 & 12 \\ 0 & 6 & 24 & 30 \end{bmatrix}$$

Second, we treat $C_{22}$ as the pivot and eliminate the second element in the lower rows.

$$C = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 0 & 2 & 6 & 12 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Finally, we implement backward substitution.

$$X = \begin{bmatrix} -14 \\ 9 \\ -1 \end{bmatrix}$$

So the solution of the system is

$$\begin{cases} x_1 = -14 \\ x_2 = 9 \\ x_3 = -1 \end{cases}$$

(2) Cholesky elimination

In linear algebra, the Cholesky decomposition or Cholesky factorization is a decomposition of a positive-definite matrix into the product of a lower triangular matrix and its transpose, useful for efficient numerical solutions.[2]

First, we factorize matrix A into matrix L,

$$A = LL^T$$

which is a lower-triangular matrix and transposed L, based on

$$\begin{cases} L_{jj} = \sqrt{A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2}, i = j \\ L_{ij} = \dfrac{A_{ij} - \sum_{k=1}^{j-1} L_{ik} \cdot L_{jk}}{L_{jj}}, i \neq j \end{cases}$$

After the factorization, we have

$$LL^T X = B$$

We can make it two equations.

$$\begin{cases} LV = B \\ L^T X = V \end{cases}$$

As L is triangular, we can implement forward substitution and backward substitution to solve the system.

## 3. Experimental Results

(1) Case 1:

The average runtime using the Gaussian elimination is 12.8358 seconds. The average runtime using Cholesky factorization algorithm is 12.6067 seconds. The thermal plot is shown as following:

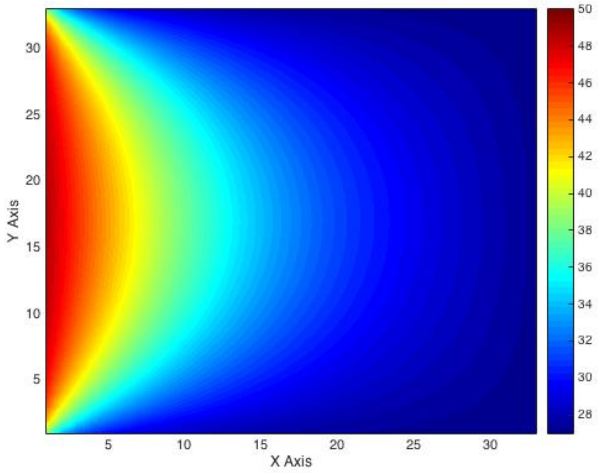


Figure 1 Thermal plot for case 1

(2) Case 2:

The average runtime using the Gaussian elimination is 125.8001 seconds. The average runtime using Cholesky factorization algorithm is 95.6242 seconds. The thermal plot is shown as following:

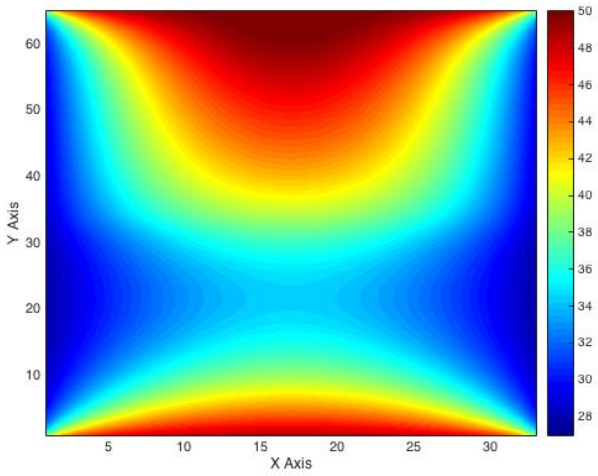


Figure 2 Thermal plot for case 2

(3) Case 3:
The average runtime using the Gaussian elimination is 1146.911967 seconds. The average runtime using Cholesky factorization algorithm is 774.5910 seconds. The thermal plot is shown as following:

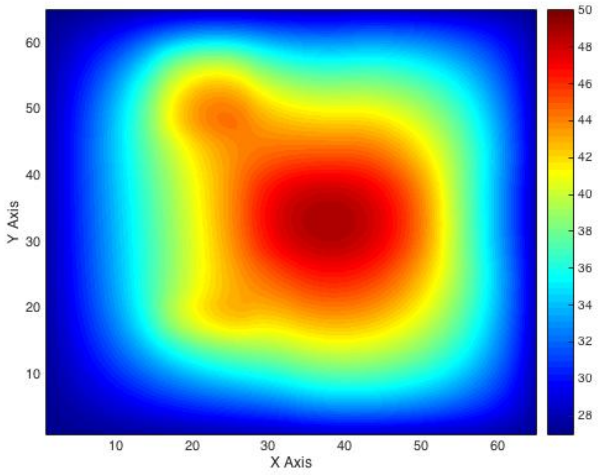


Figure 3 Thermal plot for case 3

## 4. Discussion

(1) Accuracy
Compared with the golden solution (X=A\B), the average error for each case is shown in Table 1.

Table 1 Average errors for the cases

| Case | 1 | 2 | 3 |
|---|---|---|---|
| Gaussian elimination | 2.4042e-16 | 2.5276e-16 | 3.3044e-16 |
| Cholesky factorization | 3.5605e-15 | 7.1030e-15 | 4.4680e-15 |

It can be seen that the errors are all much smaller than 0.01, which shows very high accuracy the programs have achieved. In addition, the results of Cholesky algorithm is about an order of magnitude larger than those of Gaussian elimination.

(2) Comparison
From the previous part of the experimental results, it can be seen that when the size of system is small, the difference between the runtimes of the two algorithms is very small. With the size increasing, the differences become larger and Cholesky factorization performs more efficiently with less time.

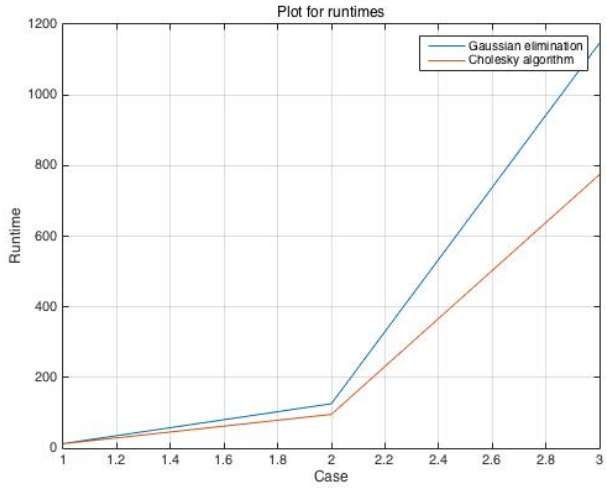


Figure 4 Plot for runtimes of two methods

(3) Uniqueness
Through observations on matrix A, I have found that it consists of three kinds of matrices as follows:

$$1:\begin{bmatrix} \frac{2(\Delta x^2+\Delta y^2)}{\Delta x^2\Delta y^2} & -\frac{1}{\Delta y^2} & 0 & 0 & 0 \\ -\frac{1}{\Delta y^2} & \frac{2(\Delta x^2+\Delta y^2)}{\Delta x^2\Delta y^2} & -\frac{1}{\Delta y^2} & 0 & 0 \\ 0 & -\frac{1}{\Delta y^2} & \frac{2(\Delta x^2+\Delta y^2)}{\Delta x^2\Delta y^2} & -\frac{1}{\Delta y^2} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & -\frac{1}{\Delta y^2} & \frac{2(\Delta x^2+\Delta y^2)}{\Delta x^2\Delta y^2} \end{bmatrix}$$

$$2:\begin{bmatrix} -\frac{1}{\Delta x^2} & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{\Delta x^2} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\Delta x^2} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & -\frac{1}{\Delta x^2} \end{bmatrix}$$

$$3:\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Therefore, the big matrix A can be divided into a number of small matrices above. The length of each small matrix is equal to M, which means points with the same x-coordinate (the same "i") share the same small matrix. In addition, matrix 1 occupies

the diagonal of the big matrix A. And matrix 2 stands on both sides of matrix 1. The remaining space is occupied by matrix 3. With this observation, I can easily generate matrix A with different N and M.
One more uniqueness is about matrix B. As matrix F is unfolded from 2-D to 1-D, the index for $f(i,j)$ can be derived from the following formula:

$$index = mod(j, M = 1) + M \times [mod(i, N + 1) - 1]$$

This index formula plays an important role in determining which point in matrix B should add the boundary temperature. The first M points add the left ones, while the last M points add the right ones. If

$$mod(index, M) == 0$$

this kind of points sit on the top boundary. If

$$mod(index, M) == 1$$

this kind of points sit on the bottom boundary. This index formula brings efficiency in determining boundaries.

(4) Improvement
The algorithms inside the Gaussian elimination can be improved. For example, the simple result of the Gaussian elimination is an upper-triangular matrix. But further elimination can be implemented on this matrix. It can be changed into a diagonal matrix, which makes the solution more efficiently without the use of backward substitution.

## References


[1] Gaussian elimination, "Gaussian elimination - Wikipedia" [Online] Available: https://en.wikipedia.org/wiki/Gaussian_elimination
[2] Cholesky decomposition, "Cholesky decomposition - Wikipedia" [Online] Available: https://en.wikipedia.org/wiki/Cholesky_decomposition