Project 1: 2-D Thermal Analysis

Name : Jiaxin Chen

Email: jiaxinc@andrew.cmu.edu

Abstract

I simulate the 2-D steady-state thermal problem by MATLAB. I build the heat equation and simplify the 2nd-order linear PDE according to the steday state. And I discretize PDE with boundary condition to construct the linear system. Finally I plot the temperature distribution by solving the linear system with Gaussian Elimination and Cholesky Factorization respectively.

1 Mathematical Formulation

Given the 2nd-order linear PDE(heat equation):

$$\rho \cdot C_p \cdot \frac{\partial T(x,y,t)}{\partial t} = \kappa \cdot \nabla^2 T(x,y,t) + f(x,y,t) \quad (1)$$

Assuming the heat conduction has reached a steady state, which means:

$$\frac{\partial T(x,y,t)}{\partial t} = 0 \quad (2)$$

We can simplify the PDE as:

$$\kappa \cdot \nabla^2 T(x,y) + f(x,y) = 0 \quad (3)$$

Then we can discretize $\nabla^2 T(x,y)$ as the linear algebra equation:

$$\begin{cases} \frac{\partial^2 T(i,j)}{\partial x^2} = \frac{T_{i+1,j}+T_{i-1,j}-2T_{i,j}}{\Delta x^2} \\ \frac{\partial^2 T(i,j)}{\partial y^2} = \frac{T_{i,j+1}+T_{i,j-1}-2T_{i,j}}{\Delta y^2} \end{cases} \quad (4)$$

Especially for the boundary condition of leftBound, rightBound, topBound and bottomBound, we consider:

$$\begin{aligned} T(0,j) &= T_c \qquad T(N+1,j) = T_c \\ T(i,M+1) &= T_c \qquad T(i,0) = T_c \end{aligned} \quad (5)$$

Therefore we can combine all the linear equations together:

$$\begin{aligned} &\kappa \cdot [\frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{\Delta x^2} + \frac{T_{i,j+1} + T_{i,j-1} - 2T_{i,j}}{\Delta y^2}] \\ &= -f(i,j) \qquad (1 \le i \le N \quad 1 \le j \le M) \end{aligned} \quad (6)$$

And we obtain the linear system equation:

$$\begin{aligned} &A \cdot X = B \\ &X = [T_{1,1}, T_{1,2}, \cdots, T_{N,M}]^T \end{aligned} \quad (7)$$

I solve this linear system equation in MATLAB by the following steps:

1) Because of the symmetric and diagonally dominant property of coefficient matrix X, I only calculate the coefficents for $T_{i,j}$ in the diagonal of matrix A and the coefficients for $T_{i-1,j}$, $T_{i+1,j}$ and $T_{i,j-1}$, $T_{i,j+1}$ in the 4 symetric lines of matrix A. Besides, I set the boundary condition in matrix A as 0. And then I combine them to construct matrix A.

$$\begin{aligned} &\frac{1}{\Delta x^2}T_{i-1,j} + \frac{1}{\Delta x^2}T_{i+1,j} + \frac{1}{\Delta y^2}T_{i,j-1} + \frac{1}{\Delta y^2}T_{i,j+1} \\ &- 2(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2})T_{i,j} = -\frac{f(i,j)}{k} \end{aligned} \quad (8)$$

2) Calculate vector B by the heat source p firstly and set the boundary condition.

3) Implement Gaussian Elimination and Cholesky Factorization to solve the linear system:

   a) Gaussian Elimination
   Convert A to an upper triangular matrix.
   Solve for X via backward substitution.

   b) Cholesky Factorization
   Decompose matrix $A = LL^T$, L is lower triangular:

$$\begin{cases} LV = B \\ L^T X = V \end{cases} \quad (9)$$

   Solve for V via forward substitution and X via backward substitution.

4) Plot the thermal distribution for each case.

2 Linear System Solver

1) Gaussian ELimination

   a) Calculate the factor

$$factor = A_i \cdot (Aj, i/Ai, i) \quad (10)$$

   b) Replace the old A_j with the new update A_j:

$$A_j = factor + A_j \quad (11)$$

   c) Repeat the step (a) and (b) iteratively until we convert the matrix A into the upper triangular matrix as well as updating the matrix B.

   d) Calculate the X from the last one:

$$X_i = B_i / A_{i,i} \quad (12)$$

e) Update matrix B:

$$B_{i-1} = B_{i-1} - A_{i-1} \cdot X \tag{13}$$

f) Repeat the step (d) and (e) above iteratively, we can solve for X via backward substitution X finally by Gaussian Elimination.

2) Cholesky Factorization

a) Convert matrix A into positive definite matrix.

b) Compute the scaler

$$L_{i,i} = \sqrt{A_{i,i} - L_i L_i^T} \tag{14}$$

c) Update matrix L

$$L_{j,i} = \frac{A_{j,i} - L_j L_i^T}{L_{i,i}} \tag{15}$$

d) Repeat the step (b) and (c) iteratively to decompose matrix A into matrix L and L^T.

e) Solve the lower triangle matrix for V via forward substitution, which calculates the matrix V from the first one and update matrix B iteratively to obtain V.

$$V_i = B_i / L_{i,i} \qquad B_{i+1} = B_{i+1} - L_{i+1} \cdot V \tag{16}$$

f) Solve the upper triangle matrix for X via backward substitution, which calculates the matrix X from the last one and update matrix V iteratively to obtain X.

$$X_i = V_i / Ui, i \qquad V_{i-1} = V_{i-1} - U_{i-1} \cdot X \tag{17}$$

3 Experimental Results

1) Gaussian Elimination

- Case1
  Elapsed time is 4.790479 seconds.
  error = 1.9386e-15
- Case2
  Elapsed time is 26.141486 seconds.
  error = 8.1967e-14
- Case3
  Elapsed time is 216.004320 seconds.
  error = 6.9443e-14

2) Cholesky Factorization

- Case1
  Elapsed time is 3.746629 seconds.
  error = 8.2067e-14
- Case2
  Elapsed time is 24.413962 seconds.
  error = 4.5168e-13
- Case3
  Elapsed time is 190.280303 seconds.
  error = 1.2027e-13

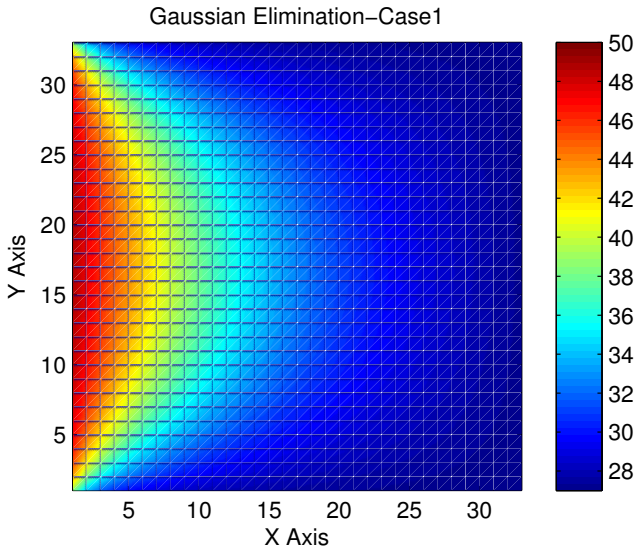

Figure 1: **Gaussian Thermalplot - case1**

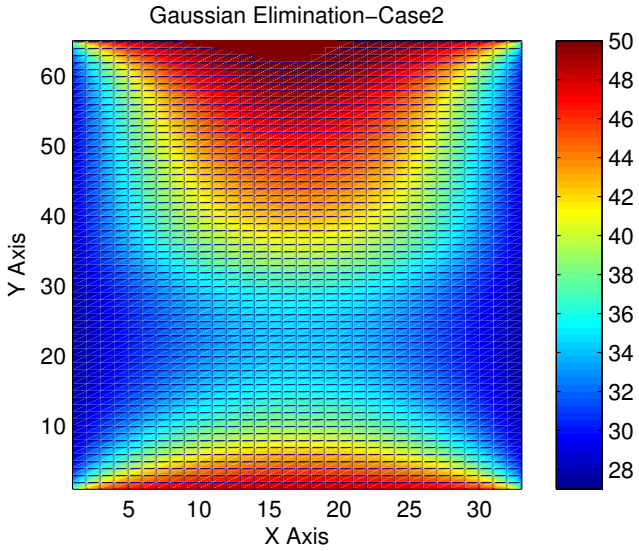

Figure 2: **Gaussian Thermalplot - case2**

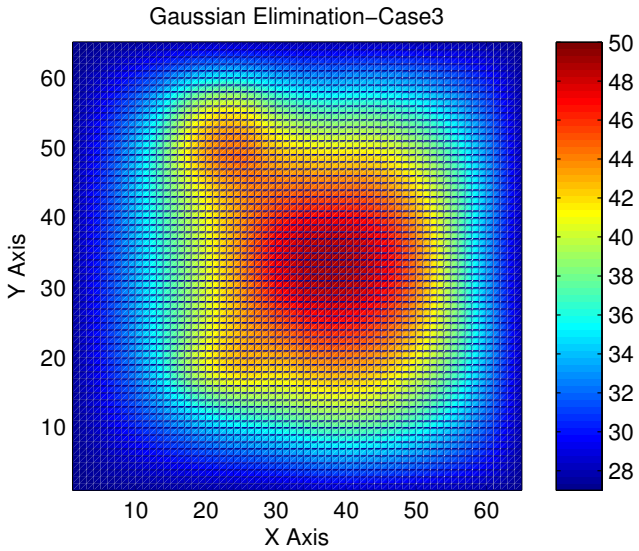

Figure 3: **Gaussian Thermalplot - case3**

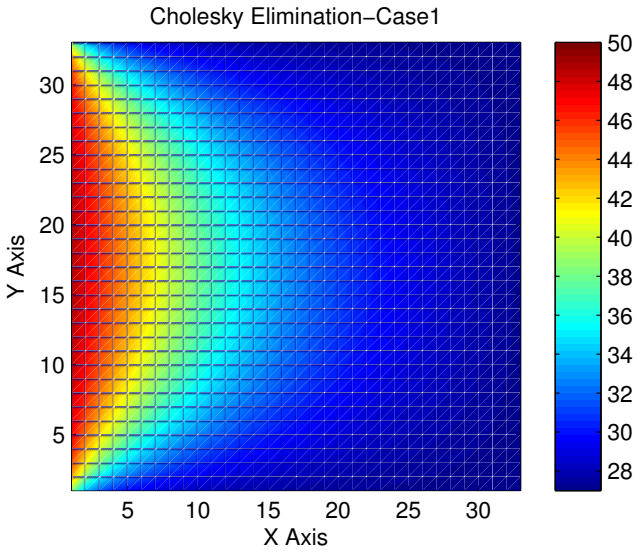

Figure 4: **Cholesky Thermalplot - case1**

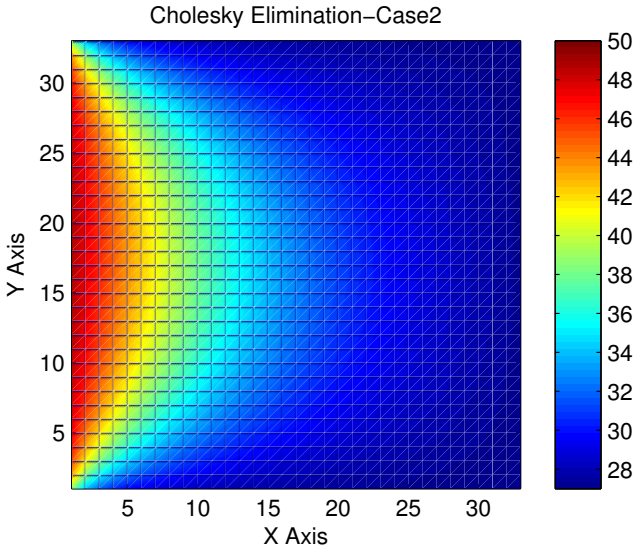

Figure 5: **Cholesky Thermalplot - case2**

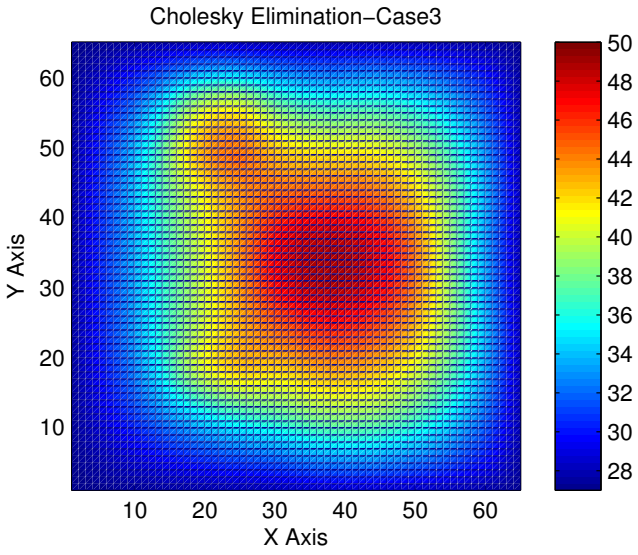

Figure 6: **Cholesky Thermalplot - case3**

		Error	Elapsed Time(s)
Gaussian	case1	1.9386e-15	4.790479
	case2	8.1967e-14	26.141486
	case3	6.9443e-14	216.004320
Cholesky	case1	8.2067e-14	3.746629
	case2	4.5168e-13	24.413962
	case3	1.2027e-13	190.280303

Table 1: **Accuracy and Efficiency**

4 Discussion

- Accuracy and efficiency my program can achieve

  I calculate the golden solution by MATLAB:

  $$T^* = A \backslash B; \tag{18}$$

  And I compare my results to the golden solution for accuracy by:

  $$Error = \sqrt{\frac{\sum_i \sum_j [T^*(i,j) - T(i,j)]^2}{\sum_i \sum_J [T^*(i,j)]^2}} \tag{19}$$

  Therefore, I can compute the error for every case and record the elapsed time as Table 1:

  According to the Table 1, I have some thoughts by comparing the accuracy and efficiency between Gaussian Elimination and Cholesky Factorization.

  a) Gaussian Elimination achieves better accuracy.
  Gaussian Elimination calculates the upper triangular matrix only once while cholesky factorization need twice, which means Gaussian Elimination can avoid more errors than Cholesky Factorization when computing.

  b) Cholesky Factorization achieves higher efficiency.
  The matrix A can be iteratively replaced by L and L^T when using Cholesky Factorization to compute. So no additional memory is required, which means it's more memory efficient and Cholesky Factorization computes faster than Gaussian Elimination.

- Size of problem my program can handle

  Currently my program can handle the 65×65 size input less than 200 seconds with high accuracy and efficiency without any bugs. Thereofore, I have confidence on my program that it can handle much bigger size.

- Possible improvements that can be done

  a) We can write the first part of creating matrix A and matrix B as the coefficients generation function. Therefore we can invoke this function and load its results for the same case without generating the matrix again, which can improve performance better.

b) We can exploit the built-in function in MATLAB as much as possible to simplify the code. For example, beacause the for loop takes lots of time to compute in MATLAB, sometimes we can use other function such as bsxfun to make our code more efficient.

- Anything unique I have done to improve/validate my program's accuracy/efficiency

a) In order to save time and improve performance, I utilize the symmetric and diagonally positive property to calculate matrix A. I only calculate the coefficents for $T(i,j)$ in the diagonal of matrix A and the coefficients for $T_{i-1,j}$, $T_{i+1,j}$ and $T_{i,j-1}$, $T_{i,j+1}$ in the 4 symetric lines of matrix A and set the boundary condition in matrix A as 0. Compared the time by using for loop to calculate matrix A and matrix B, my algorithm runs about four times faster.

b) For validating the accuracy of my code and record the elapsed time, I write the script `ErrorEstimation_Gaussian.m` and `ErrorEstimation_Cholesky.m` to fulfill this job. I write the equation (10) in the script to validate my results.

References

[1] Shapiro, L. G. ; Stockman, G. C: *"Computer Vision"*, page 137, 150. Prentence Hall, 2001

[2] R.A. Haddad and A.N. Akansu, *"A Class of Fast Gaussian Binomial Filters for Speech and Image Processing,"* IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 39, pp 723-727, March 1991.

[3] Mark S. Nixon and Alberto S. Aguado. *"Feature Extraction and Image Processing"*. Academic Press, 2008, p. 88.

[4] Gentle, J. E. *"Cholesky Factorization."* §3.2.2 in Numerical Linear Algebra for Applications in Statistics. Berlin: Springer-Verlag, pp. 93-95, 1998.

[5] Nash, J. C. *"The Choleski Decomposition."* Ch. 7 in Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation, 2nd ed. Bristol, England: Adam Hilger, pp. 84-93, 1990.

[6] Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; and Vetterling, W. T. *"Cholesky Decomposition."* §2.9 in Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed. Cambridge, England: Cambridge University Press, pp. 89-91, 1992.