MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF MECHANICAL ENGINEERING

ME 310 – NUMERICAL METHODS
SPRING 2022

HOMEWORK 2

Muhammed Rüstem ŞEVİK - 2446888

I understand that this is an individual assignment. I affirm that I have not given or received any unauthorized help on this assignment, and that this work is my own.

Table of Contents

1 Introduction

This report aims to provide a solution to the given engineering problems using linear algebraic equation systems. The report addresses questions related to a truss structure, a steady one-dimensional heat conduction and a steady two-dimensional heat conduction problems, which require solving linear equation systems. For the truss problem, the report discusses three methods, MATLAB linsolve() function, naive Gaussian elimination and partial row pivoting Gaussian elimination to solve the system. For the heat conduction problems, the report presents Thomas algorithm, Cholesky decomposition and Gauss-Seidel algorithm. The report aims to provide a detailed analysis of the solutions obtained from each method and discuss the feasibility of the results.

2 Body

2.1 Question 3

The question asks for the solution of a given system by using various methods stated in separate parts.

$$\begin{bmatrix} 0 & 0.9231 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -0.3846 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -0.8575 & 0 \\ -1 & 0 & -0.7809 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3846 & 0.7809 & 0 & 1 & -0.3846 & 0 & 0 \\ 0 & -0.9231 & -0.6247 & 0 & 0 & 0.9231 & 0 & 0 \\ 0 & 0 & 0.6247 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0.5145 & 1 \end{bmatrix} \begin{bmatrix} F_{AB} \\ F_{AC} \\ F_{BC} \\ F_{BD} \\ F_{CD} \\ F_{CE} \\ F_{DE} \\ F_{DF} \end{bmatrix} = \begin{bmatrix} 1690 \\ 3625 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

2.1.1 Part A

The solution of the system by using linsolve function of MATLAB:

$$\begin{bmatrix} F_{AB} \\ F_{AC} \\ F_{BC} \\ F_{BD} \\ F_{CD} \\ F_{CE} \\ F_{DE} \\ F_{DF} \end{bmatrix} = \begin{bmatrix} 4329.1 \\ 1830.8 \\ -5543.8 \\ 3463.2 \\ 2886.2 \\ -1920.9 \\ -3365.9 \\ 5194.9 \end{bmatrix} \tag{2}$$

2.1.2 Part B

The solution of the system by using Naive.m code:

$$\begin{bmatrix} F_{AB} \\ F_{AC} \\ F_{BC} \\ F_{BD} \\ F_{CD} \\ F_{CE} \\ F_{DE} \\ F_{DF} \end{bmatrix} = \begin{bmatrix} NaN \\ NaN \\ NaN \\ NaN \\ NaN \\ NaN \\ NaN \\ NaN \end{bmatrix} \tag{3}$$

The solution is not the same as the part a. Note that, NaN (Not-a-Number) is a special value in MATLAB that represents undefined or unrepresentable numerical results. It can be returned in scenarios such as arithmetic operations involving undefined values, functions with undefined or unrepresentable results, or functions that encounter errors. NaN can propagate through subsequent calculations, so it's important to handle NaN values appropriately. The *isnan* function can be used to check if a value is NaN.

2.1.3 Part C

Partial pivoting is a technique used in linear algebra when solving systems of equations. It involves swapping the rows of a matrix in order to ensure that the largest absolute value element of a column is placed in the pivot position. This helps to prevent numerical errors and instability when performing Gaussian elimination or LU decomposition. By using partial pivoting, the resulting solutions are more accurate and reliable.
The solution of the partial row pivoting Gaussian elimination code:

$$\begin{bmatrix} F_{AB} \\ F_{AC} \\ F_{BC} \\ F_{BD} \\ F_{CD} \\ F_{CE} \\ F_{DE} \\ F_{DF} \end{bmatrix} = \begin{bmatrix} 4329.1 \\ 1830.8 \\ -5543.8 \\ 3463.2 \\ 2886.2 \\ -1920.9 \\ -3365.9 \\ 5194.9 \end{bmatrix} \tag{4}$$

The solution is the same as the part a. The reason of that the naive gaussian elimination is failed is that the division by zero case is not considered.

2.1.4 Part D

Highest tensile force member:

$$F_{DF} = 5194.9 \tag{5}$$

Highest compressive force member:

$$F_{BC} = -5543.8 \tag{6}$$

Considering the assumptions these results makes sense.

2.1.5 Code

```
clc; clear;
A = [0  0.9231  0  0  0  0  0  0;
     1  -0.3846  0  0  0  0  0  0;
     0  0  0  0  -1  0  -0.8575  0;
     -1  0  -0.7809  0  0  0  0  0;
     0  0.3846  0.7809  0  1  -0.3846  0  0;
     0  -0.9231  -0.6247  0  0  0.9231  0  0;
     0  0  0.6247  1  0  0  0  0;
     0  0  0  -1  0  0  0.5145  1];
b = [1690; 3625; 0; 0; 0; 0; 0; 0];

PPGE(A, b)

function [x] = PPGE(A, b)
N = size(A,1);
x = zeros(N,1);    % Allocate memory for the solution vector

% Forward elimination
for k = 1:N-1                          % Eliminate unknown k from Eqn. k+1, k+2,      , N
    % Find the row with the largest absolute value in column k and swap with row k
    [~, maxRow] = max(abs(A(k:N, k))); % search for max in column k below row k
    maxRow = maxRow + k - 1; % adjust the row number to the original index
    if maxRow ~= k % if the row is different from the current one, swap them
        A([k, maxRow], k:N) = A([maxRow, k], k:N);
        b([k, maxRow]) = b([maxRow, k]);
    end
    for i = k+1:N                      % Modifying Eqn. i.
        factor = A(i,k) / A(k,k);
        for j = k+1:N                  % Only modify {b} and the upper triangular part of [A
            ].
                                       % Do not waste time to set the lower part of [A] to
                                          zero.
            A(i,j) = A(i,j) - A(k,j) * factor;
        end
        b(i) = b(i) - b(k) * factor;
    end
end


% Back substitution
x(N) = b(N)/A(i,i);      % Solve the last unknown using the last equation.
for i = N-1:-1:1         % This for loop runs backwards, from N-1 to 1 with increments of
     -1.
    sum = b(i);
    for j = i+1:N
        sum = sum - A(i,j) * x(j);
    end
    x(i) = sum / A(i,i);
end
end
```

2.2 Question 4

The question asks for the solution of the given system by using Thomas algorithm, Cholesky decomposition and iterative Gasuss-Seidel algorithm with relaxation.

$$\begin{bmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ \cdot \\ \cdot \\ \cdot \\ T_{N-1} \\ T_N \end{Bmatrix} = -\frac{\dot{q}(\Delta x)^2}{k} \begin{Bmatrix} 1 \\ 1 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \\ 1 \end{Bmatrix} + \begin{Bmatrix} -T_L \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ -T_R \end{Bmatrix} \tag{7}$$

2.2.1 Part A

The solution obtained by the Thomas algorithm as follows:

$$\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \\ T_{10} \\ T_{11} \\ T_{12} \\ T_{13} \\ T_{14} \\ T_{15} \end{Bmatrix} = \begin{Bmatrix} 58.9844 \\ 67.1875 \\ 74.6094 \\ 81.2500 \\ 87.1094 \\ 92.1875 \\ 96.4844 \\ 100.0000 \\ 102.7344 \\ 104.6875 \\ 105.8594 \\ 106.2500 \\ 105.8594 \\ 104.6875 \\ 102.7344 \end{Bmatrix} \tag{8}$$

2.2.2 Part B

The solution obtained by the Cholesky decomposition, forward and backward substitution the lower matrix L
Since the matrix is a banded matrix it is practical to write the matrix in two columns where for $D_i = L_{i,i}$ for $i = 1, .., 15$ and $F_i = L_{i+1,i}$ for $i = 1, .., 14$

$$D = \begin{bmatrix} 1.4142 \\ 1.2247 \\ 1.1547 \\ 1.1180 \\ 1.0954 \\ 1.0801 \\ 1.0690 \\ 1.0607 \\ 1.0541 \\ 1.0488 \\ 1.0445 \\ 1.0408 \\ 1.0377 \\ 1.0351 \\ 1.0328 \end{bmatrix}, F = \begin{bmatrix} -0.7071 \\ -0.8165 \\ -0.8660 \\ -0.8944 \\ -0.9129 \\ -0.9258 \\ -0.9354 \\ -0.9428 \\ -0.9487 \\ -0.9535 \\ -0.9574 \\ -0.9608 \\ -0.9636 \\ -0.9661 \end{bmatrix} \tag{9}$$

The solution obtained by the Cholesky decomposition, forward and backward substitution as follows:

$$\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \\ T_{10} \\ T_{11} \\ T_{12} \\ T_{13} \\ T_{14} \\ T_{15} \end{Bmatrix} = \begin{Bmatrix} 58.9844 \\ 67.1875 \\ 74.6094 \\ 81.2500 \\ 87.1094 \\ 92.1875 \\ 96.4844 \\ 100.0000 \\ 102.7344 \\ 104.6875 \\ 105.8594 \\ 106.2500 \\ 105.8594 \\ 104.6875 \\ 102.7344 \end{Bmatrix} \tag{10}$$

The result is the same as the part a. This actively demonstrates that one may use a Cholesky decomposition to solve systems of linear equations.

2.2.3 Part C

Yes, the matrix with -2 on the main diagonal and 1 on the adjacent diagonals is diagonally dominant.

Diagonal dominance is a property of matrices where the absolute value of the diagonal elements is greater than or equal to the sum of the absolute values of the other elements in the same row. In this case, the absolute value of the diagonal element (-2) is greater than the sum of the absolute values of the other elements in the same row (which is equal to 1). This property ensures that the matrix is well-conditioned and invertible, and it can also help to ensure that certain numerical algorithms converge quickly and accurately.

Therefore, the matrix with -2 on the main diagonal and 1 on the adjacent diagonals is diagonally dominant.

2.2.4 Part D

The gauss seidel algorithm converged to the same solution with the given tolerance in 447 iterations.

$$\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \\ T_{10} \\ T_{11} \\ T_{12} \\ T_{13} \\ T_{14} \\ T_{15} \end{Bmatrix} = \begin{Bmatrix} 58.9844 \\ 67.1875 \\ 74.6094 \\ 81.2500 \\ 87.1094 \\ 92.1875 \\ 96.4844 \\ 100.0000 \\ 102.7344 \\ 104.6875 \\ 105.8594 \\ 106.2500 \\ 105.8594 \\ 104.6875 \\ 102.7344 \end{Bmatrix} \tag{11}$$

The relaxation parameter λ affects convergence in Gauss-Seidel iteration by controlling the degree of relaxation between successive iterations. When λ is closer to 1, the algorithm takes smaller steps towards the solution at each iteration, resulting in slower but more stable convergence. On the other hand, when λ is further from 1, the algorithm takes larger steps towards the solution, resulting in faster but less stable convergence. The choice of λ is therefore a trade-off between convergence speed and stability.

In the case of our problem, I observed that smaller λ values lead to decrease in convergence rate and increasing λ from 1 to 1.6 leads to faster convergence, as the number of iterations required to reach the desired tolerance decreases. This is because increasing λ allows the algorithm to take larger steps towards the solution at each iteration, accelerating the convergence process.However, choosing λ too large can lead to instability or divergence of the algorithm, as we have observed in our problem. Increasing λ from 1.6 to 2 lead to slow down the convergence rate. Therefore, it is important to choose λ carefully, balancing the desire for faster convergence with the need for stability and accuracy.

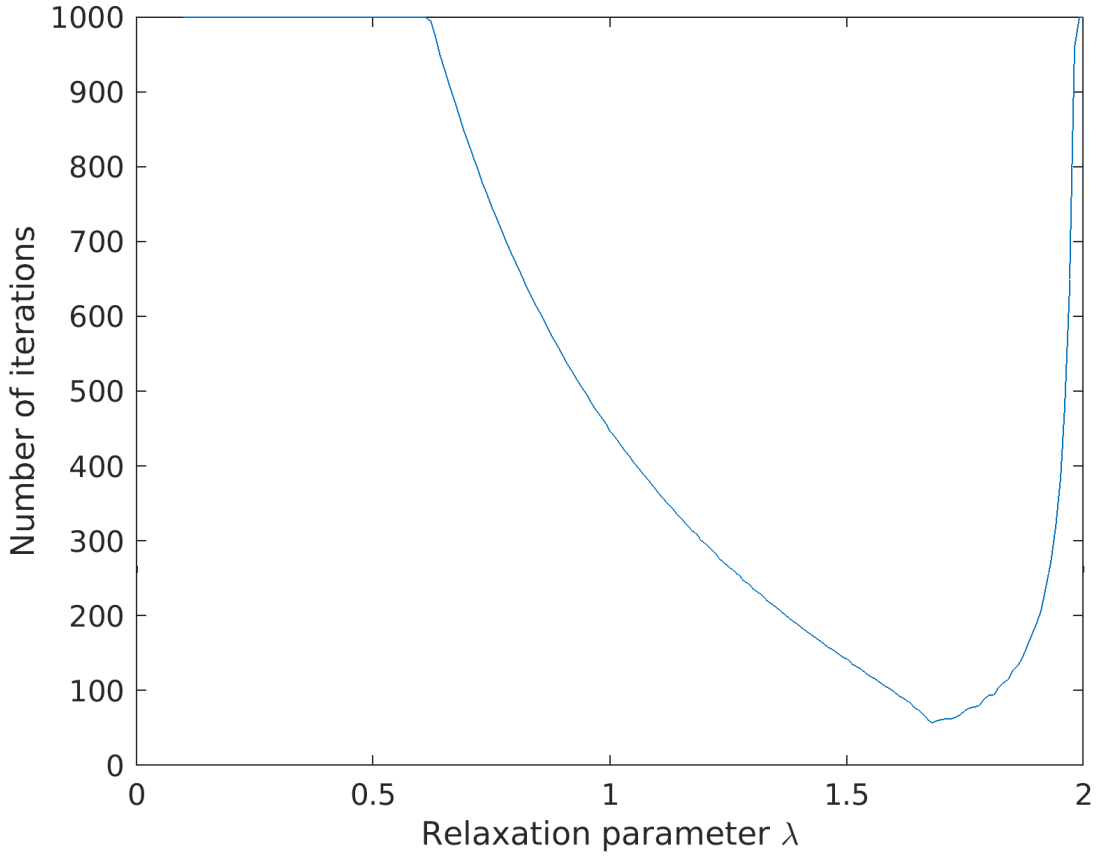

Figure 1: Relaxation Parameter λ versus Number of Iterations

2.2.5 Code

```
clc; clear;
L = 0.1; Tl = 50; Tr = 100; k = 80; q = 4e+5; N = 15;

% Create tridiagonal matrix columns
f = -2 * ones(N,1);
e = ones(N,1); e(1) = 0;
g = ones(N,1); g(N) = 0;

% Create right-hand side vector
dx = 2* L / (N +1);
r = - q * dx^2 / k * ones(N,1);
r(1) = r(1) - Tl; r(N) = r(N) - Tr;

% Create a tridiagonal full matrix
A = create_array(N);

i_guess = (Tr + Tl) / 2 *ones(N, 1);

sol = linsolve(A, r)
thom = thomas_algorithm(f, e, g, r)
L = choleskyDecomposition(-A)
chol = cholesky_solution(L, -r)
seid = gauss_seidel(A, r, i_guess, 1000, 1e-7, 1)

lambda_vals = 0.1:0.01:2;

num_iters = zeros(size(lambda_vals));
for i = 1:length(lambda_vals)
    [x, iter] = gauss_seidel(A, r, i_guess, 1000, 1e-7, lambda_vals(i));
    num_iters(i) = iter;
end

```

```
plot(lambda_vals, num_iters);
xlabel('Relaxation parameter \lambda');
ylabel('Number of iterations');
exportgraphics(gca ,"lambda_vs_noi.png", 'Resolution', 300)

function x = thomas_algorithm(f, e, g, r)
    N = length(r);

    % Forward substitution
    for i = 2:N
        factor = e(i) / f(i-1);
        f(i) = f(i) - factor * g(i-1);
        r(i) = r(i) - factor * r(i-1);
    end

    % Back substitution
    x = zeros(N,1);
    x(N) = r(N) / f(N);
    for i = N-1:-1:1
        x(i) = (r(i) - g(i) * x(i+1)) / f(i);
    end

end

function A = create_array(n)
    A = -2*eye(n);
    for i = 1:n-1
        A(i,i+1) = 1;
        A(i+1,i) = 1;
    end
end

function L = choleskyDecomposition(A)
    n = size(A, 1); L = zeros(n, n);

    for k = 1:n
        L(k,k) = sqrt(A(k,k) - sum(L(k,1:k-1).^2));
        for l = k+1:n
            L(l,k) = (A(l,k) - sum(L(l,1:k-1).*L(k,1:k-1))) / L(k,k);
        end
    end
end

function x = cholesky_solution(L, b)
    n = size(L, 1); y = zeros(n, 1); x = zeros(n, 1);

    %Forward
    for i = 1:n
        y(i) = (b(i) - L(i,1:i-1)*y(1:i-1)) / L(i,i);
    end

    %Backward

    for i = n:-1:1
        x(i) = (y(i) - L(i+1:n,i)'*x(i+1:n)) / L(i,i);
    end

end



function [x, iter] = gauss_seidel(A, b, x ,max_iter, e, lambda)
    n = length(b);
    xi_sum = 0;
    for iter = 1:max_iter
        for i = 1:n
            for j = 1:n
                if j ~= i
                    xi_sum = xi_sum + A(i,j)*x(j);
                end
            end
            x(i) = lambda*((b(i) - xi_sum)/A(i,i)) + (1 - lambda)*x(i);
            xi_sum = 0;
```

```
        end
        residual_norm = norm(A*x - b);
        if residual_norm < e
            converged = true;
        else
            converged = false;
        end

        if iter ~= 1 && converged
            break;
        end
    end

end
```

2.3 Question 5

2.3.1 Part A

The temperature distribution inside the plate is given by the following Laplace equation

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{12}$$

To discretize the Laplace equation for any inner node (i, j) use the following approximate derivatives:

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{i,j} \cong \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{(\Delta x)^2}, \quad \left.\frac{\partial^2 T}{\partial y^2}\right|_{i,j} \cong \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{(\Delta y)^2} \tag{13}$$

Plug the approximations into the Laplace equation

$$\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{(\Delta x)^2} + \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{(\Delta y)^2} = 0 \tag{14}$$

For $\Delta x = \Delta y$ the equation simplifies to following form:

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = 0 \tag{15}$$

To obtain the equations for $i = 1, 2, 3$ and $j = 1, 2, 3$, we substitute the corresponding values of i and j into the equation and simplify.

$$\begin{aligned}
T_{2,1} + T_{0,1} + T_{1,2} + T_{1,0} - 4T_{1,1} &= 0 \\
T_{2,2} + T_{0,2} + T_{1,3} + T_{1,1} - 4T_{1,2} &= 0 \\
T_{2,3} + T_{0,3} + T_{1,4} + T_{1,2} - 4T_{1,3} &= 0 \\
T_{3,1} + T_{1,1} + T_{2,2} + T_{2,0} - 4T_{2,1} &= 0 \\
T_{3,2} + T_{1,2} + T_{2,3} + T_{2,1} - 4T_{2,2} &= 0 \\
T_{3,3} + T_{1,3} + T_{2,4} + T_{2,2} - 4T_{2,3} &= 0 \\
T_{4,1} + T_{2,1} + T_{3,2} + T_{3,0} - 4T_{3,1} &= 0 \\
T_{4,2} + T_{2,2} + T_{3,3} + T_{3,1} - 4T_{3,2} &= 0 \\
T_{4,3} + T_{2,3} + T_{3,4} + T_{3,2} - 4T_{3,3} &= 0
\end{aligned} \tag{16}$$

Construct the matrix A and vector b of the form $AT_{i,j} = b$:

$$\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} T_{1,1} \\ T_{1,2} \\ T_{1,3} \\ T_{2,1} \\ T_{2,2} \\ T_{2,3} \\ T_{3,1} \\ T_{3,2} \\ T_{3,3} \end{bmatrix} = \begin{bmatrix} -T_{0,1} - T_{1,0} \\ -T_{0,2} \\ -T_{0,3} - T_{1,4} \\ -T_{2,0} \\ 0 \\ -T_{2,4} \\ -T_{4,1} - T_{3,0} \\ -T_{4,2} \\ -T_{4,3} - T_{3,4} \end{bmatrix} \tag{17}$$

2.3.2 Part B

By using the linsolve function of the MATLAB the solution of the system as follows:

$$\begin{bmatrix} T_{1,1} \\ T_{1,2} \\ T_{1,3} \\ T_{2,1} \\ T_{2,2} \\ T_{2,3} \\ T_{3,1} \\ T_{3,2} \\ T_{3,3} \end{bmatrix} = \begin{bmatrix} 57.8571 \\ 56.5625 \\ 47.1429 \\ 49.8661 \\ 46.2500 \\ 37.0089 \\ 45.3571 \\ 41.5625 \\ 34.6429 \end{bmatrix} \tag{18}$$

The result makes sense because the distribution of the temperatures on the grid is as expected when the boundary temperatures are considered.

2.3.3 Part C

Yes there is an analytical solution of the Laplace equation.
The partial differential equation is:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{19}$$

Assuming the solution is separable, we write:

$$T(x, y) = X(x)Y(y) \tag{20}$$

Substituting this into the Laplace equation and dividing by $X(x)Y(y)$, we get:

$$\frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)} = 0 \tag{21}$$

Since the left-hand side depends only on x and y separately, each side must be a constant, which we call $-\lambda^2$:

$$\frac{X''(x)}{X(x)} = \frac{Y''(y)}{Y(y)} = -\lambda^2 \tag{22}$$

This leads to two ordinary differential equations:

$$X''(x) + \lambda^2 X(x) = 0 \tag{23}$$

$$Y''(y) + \lambda^2 Y(y) = 0 \tag{24}$$

The general solutions to these equations are:

$$X(x) = A\cos(\lambda x) + B\sin(\lambda x) \tag{25}$$

$$Y(y) = C\cos(\lambda y) + D\sin(\lambda y) \tag{26}$$

where A, B, C, and D are constants that depend on the boundary conditions. The general solution to the Laplace equation is then:

$$T(x,y) = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty}(A_{nm}\cos(\lambda_n x) + B_{nm}\sin(\lambda_n x))(C_{nm}\cos(\lambda_m y) + D_{nm}\sin(\lambda_m y)) \tag{27}$$

where λ_n and λ_m are the nth and mth positive roots of the equation $X''(x)+\lambda^2 X(x) = 0$, and A_{nm}, B_{nm}, C_{nm}, and D_{nm} are constants determined by the boundary conditions.

2.3.4 Code

```
T01 = 75; T02 = 75; T03 = 75;
T41 = 40; T42 = 40; T43 = 40;
T10 = 50; T20 = 50; T30 = 50;
T14 = 20; T24 = 20; T34 = 20;

A = [-4, 1, 0, 1, 0, 0, 0, 0, 0;
     1, -4, 1, 0, 1, 0, 0, 0, 0;
     0, 1, -4, 0, 0, 1, 0, 0, 0;
     1, 0, 0, -4, 1, 0, 1, 0, 0;
     0, 1, 0, 1, -4, 1, 0, 1, 0;
     0, 0, 1, 0, 1, -4, 0, 0, 1;
     0, 0, 0, 1, 0, 0, -4, 1, 0;
     0, 0, 0, 0, 1, 0, 1, -4, 1;
     0, 0, 0, 0, 0, 1, 0, 1, -4];

b = -[T01 + T10;
     T02;
     T03 + T14;
     T20;
     0;
     T24;
     T41 + T30;
     T42;
     T43 + T34];

linsolve(A, b)
```

3 Discussion

The first question in this report is related to a truss problem with six joints and eight members. The task is to solve an 8x8 linear system using the linsolve function of MATLAB and the provided NaiveGE.m code. Additionally, the question requires implementing partial row pivoting to the NaiveGE.m code and solving the system again. Naive Gaussian elimination is failed due to division by zero case and in order to prevent it partial row pivotting is applied and the problem is solved.

The second question in this report involves solving a steady one-dimensional heat conduction problem using a tridiagonal system. The Thomas algorithm is applied to the tridiagonal system. The solution is obtained. Then Cholesky decomposition is applied and the solution is obtained. The solutions are the same. Then Gauss-Seidel algorithm is applied to the system again. In desired accuracy the solution is obtained in 447 iterations. The relaxation parameter is applied in the range of 0.1 to 2 and the results are plotted. The relaxation parameter effects positively when selected properly. If not it may cause the converging system to diverge.

The third question is about the two-dimensional version of the previous problem. The question asks to construct the given system and solve it. The question also askes for the analytical solution of the system.

4 Conclusion

The homework is beneficial in terms of learning various analytical and iterative algorithms. I have also learned how these algorithms work and how parameters effect the accuracy of an algorithm. I have enhanced my ability to use MATLAB. This homework was very helpful to learn solution methods of systems of linear equations. Also in my case i have used latex to write the report. The report enhanced my ability to use tex environment.