Assignment 2

學號: 111550129　姓名: 林彥亨

1. (20%) Use Gaussian elimination with partial pivoting to solve the following equations(given as the augmented matrix). Are any row interchanges needed?

$$\left[\begin{array}{ccc|c} 3 & 1 & -4 & 7 \\ -2 & 3 & 1 & -5 \\ 2 & 0 & 5 & 10 \end{array}\right]$$

$$\left[\begin{array}{ccc|c} 3 & 1 & -4 & 7 \\ -2 & 3 & 1 & -5 \\ 2 & 0 & 5 & 10 \end{array}\right]$$

No row interchanges needed.

$$= \left[\begin{array}{ccc|c} 3 & 1 & -4 & 7 \\ -2 & 3 & 1 & -5 \\ 0 & 3 & 6 & 5 \end{array}\right]$$

$$= \left[\begin{array}{ccc|c} 3 & 1 & -4 & 7 \\ 0 & \frac{11}{3} & \frac{-5}{3} & \frac{-1}{3} \\ 0 & 3 & 6 & 5 \end{array}\right]$$

$$= \left[\begin{array}{ccc|c} 3 & 1 & -4 & 7 \\ 0 & 11 & -5 & -1 \\ 0 & 0 & \frac{81}{11} & \frac{58}{11} \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 0 & 0 & \frac{8580}{2673} \\ 0 & 1 & 0 & \frac{209}{891} \\ 0 & 0 & 1 & \frac{58}{81} \end{array}\right]$$

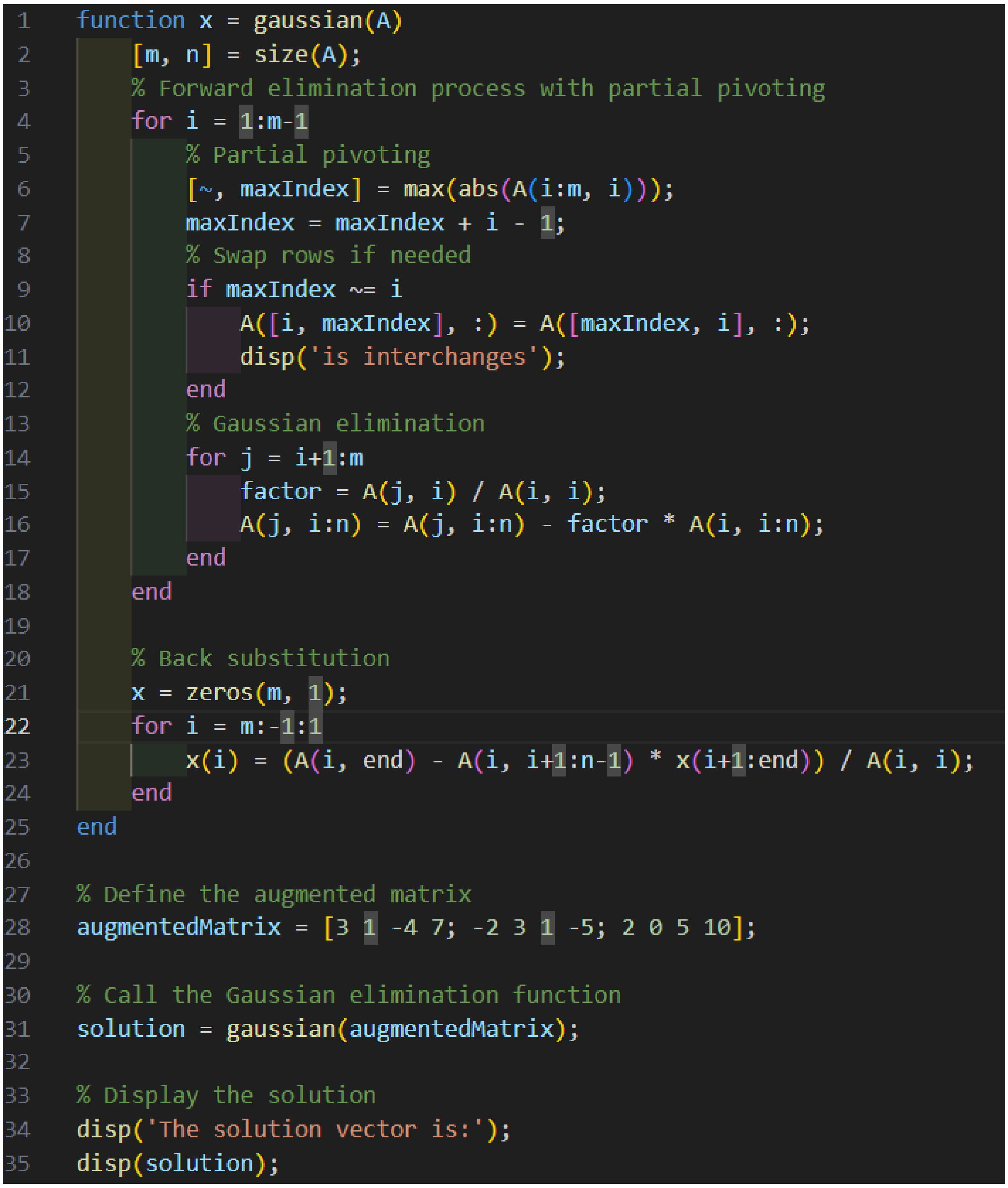
$$3x + \frac{209}{891} - 4 \times \frac{58}{81} = 7$$

$$= \frac{567+232}{81} - \frac{209}{891}$$

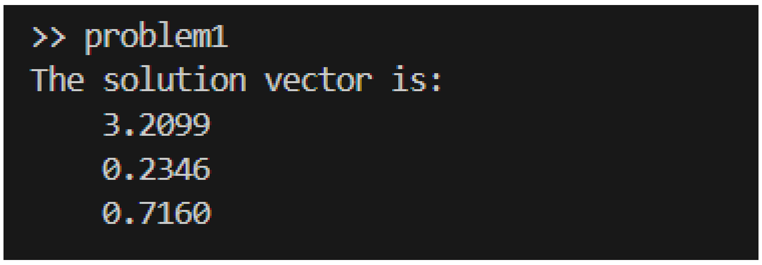
$$3x = \frac{8580}{891}$$

$$x = \frac{8580}{2673}$$

$$= \frac{799}{81} = \frac{8789-209}{891}$$



```matlab
function x = gaussian(A)
    [m, n] = size(A);
    % Forward elimination process with partial pivoting
    for i = 1:m-1
        % Partial pivoting
        [~, maxIndex] = max(abs(A(i:m, i)));
        maxIndex = maxIndex + i - 1;
        % Swap rows if needed
        if maxIndex ~= i
            A([i, maxIndex], :) = A([maxIndex, i], :);
            disp('is interchanges');
        end
        % Gaussian elimination
        for j = i+1:m
            factor = A(j, i) / A(i, i);
            A(j, i:n) = A(j, i:n) - factor * A(i, i:n);
        end
    end

    % Back substitution
    x = zeros(m, 1);
    for i = m:-1:1
        x(i) = (A(i, end) - A(i, i+1:n-1) * x(i+1:end)) / A(i, i);
    end
end

% Define the augmented matrix
augmentedMatrix = [3 1 -4 7; -2 3 1 -5; 2 0 5 10];

% Call the Gaussian elimination function
solution = gaussian(augmentedMatrix);

% Display the solution
disp('The solution vector is:');
disp(solution);
```



```
>> problem1
The solution vector is:
    3.2099
    0.2346
    0.7160
```

- I write code to check the result.

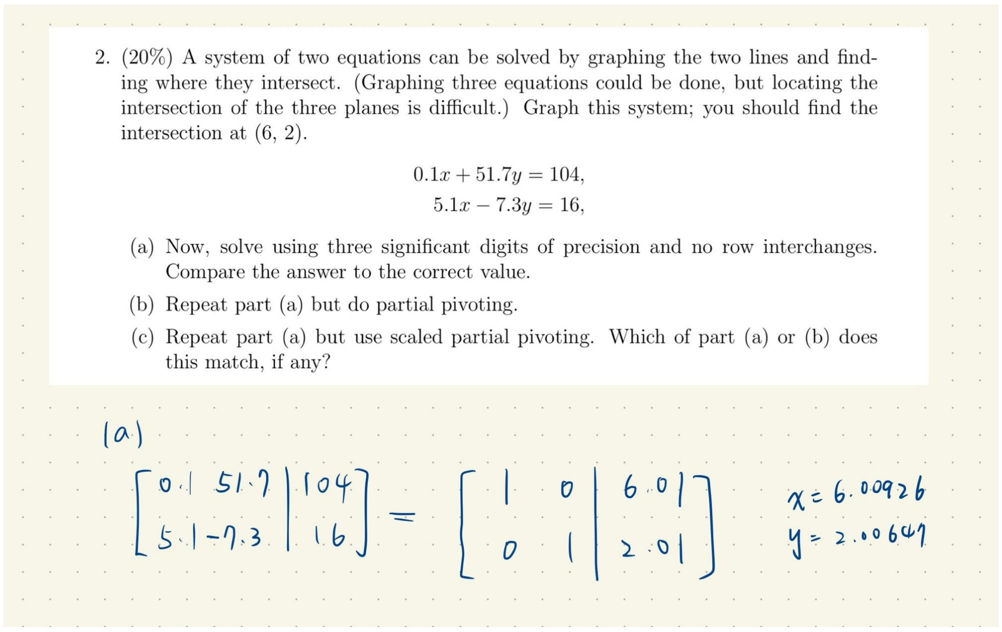
2. (20%) A system of two equations can be solved by graphing the two lines and finding where they intersect. (Graphing three equations could be done, but locating the intersection of the three planes is difficult.) Graph this system; you should find the intersection at (6, 2).

$$0.1x + 51.7y = 104,$$
$$5.1x - 7.3y = 16,$$

(a) Now, solve using three significant digits of precision and no row interchanges. Compare the answer to the correct value.

(b) Repeat part (a) but do partial pivoting.

(c) Repeat part (a) but use scaled partial pivoting. Which of part (a) or (b) does this match, if any?



2. (20%) A system of two equations can be solved by graphing the two lines and finding where they intersect. (Graphing three equations could be done, but locating the intersection of the three planes is difficult.) Graph this system; you should find the intersection at (6, 2).

$$0.1x + 51.7y = 104,$$
$$5.1x - 7.3y = 16,$$

(a) Now, solve using three significant digits of precision and no row interchanges. Compare the answer to the correct value.

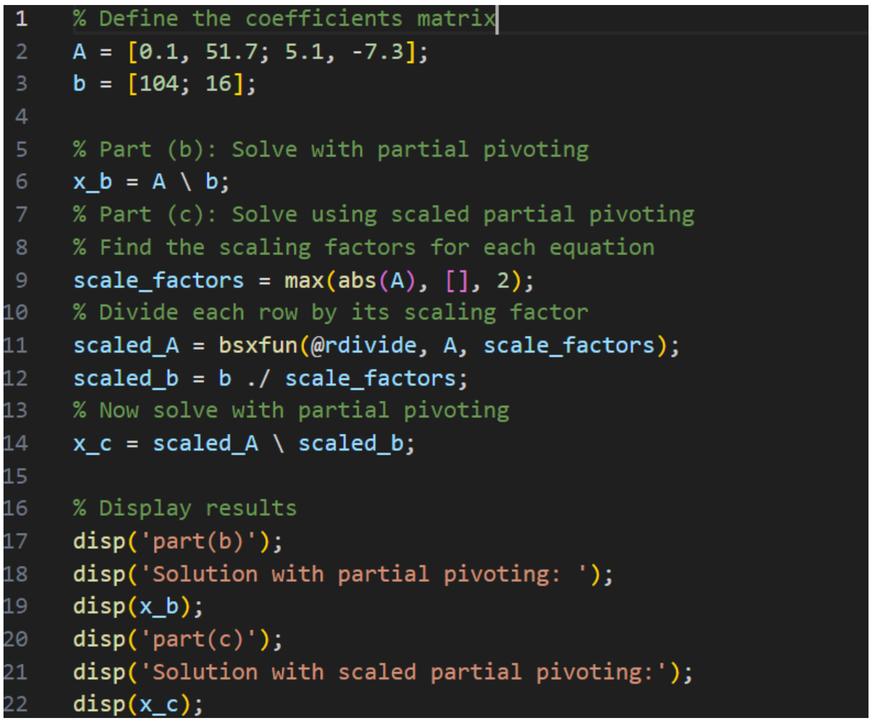
(b) Repeat part (a) but do partial pivoting.

(c) Repeat part (a) but use scaled partial pivoting. Which of part (a) or (b) does this match, if any?

(a)

$$\left[\begin{array}{cc|c} 0.1 & 51.7 & 104 \\ 5.1 & -7.3 & 16 \end{array}\right] = \left[\begin{array}{cc|c} 1 & 0 & 6.01 \\ 0 & 1 & 2.01 \end{array}\right]$$

$x = 6.00926$
$y = 2.00647$

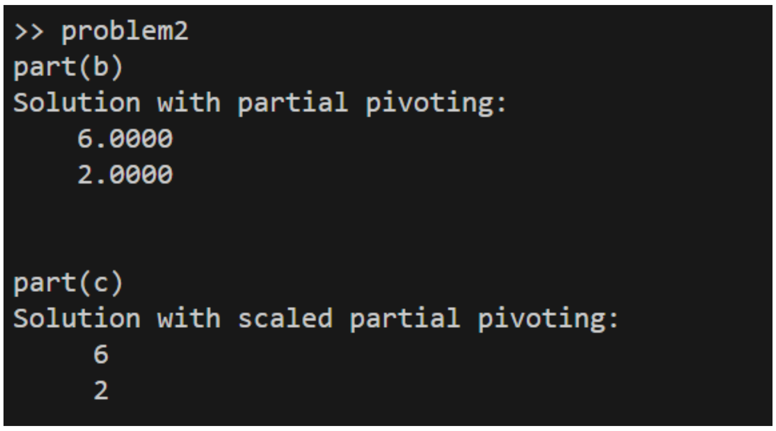


```
% Define the coefficients matrix
A = [0.1, 51.7; 5.1, -7.3];
b = [104; 16];

% Part (b): Solve with partial pivoting
x_b = A \ b;
% Part (c): Solve using scaled partial pivoting
% Find the scaling factors for each equation
scale_factors = max(abs(A), [], 2);
% Divide each row by its scaling factor
scaled_A = bsxfun(@rdivide, A, scale_factors);
scaled_b = b ./ scale_factors;
% Now solve with partial pivoting
x_c = scaled_A \ scaled_b;

% Display results
disp('part(b)');
disp('Solution with partial pivoting: ');
disp(x_b);
disp('part(c)');
disp('Solution with scaled partial pivoting:');
disp(x_c);
```

In part (b), i use the matlab fuction ‘ \ ’ to calculate the matrix with partial pivoting.

In part (c), first i find the scaleing factor, then divide the each row.

- Here is the result.



```
>> problem2
part(b)
Solution with partial pivoting:
    6.0000
    2.0000


part(c)
Solution with scaled partial pivoting:
     6
     2
```

As the result, the part(b) is match the result of scaling partial pivoting.

3. (20%) Given system A:

$$A = \begin{bmatrix} 2 & -1 & 3 & 2 \\ 2 & 2 & 0 & 4 \\ 1 & 1 & -2 & 2 \\ 1 & 3 & 4 & -1 \end{bmatrix}$$

Find the LU equivalent of matrix A that has 2's in each diagonal position of L rather than 1's.

$$A = \begin{bmatrix} 2 & -1 & 3 & 2 \\ 2 & 2 & 0 & 4 \\ 1 & 1 & -2 & 2 \\ 1 & 3 & 4 & -1 \end{bmatrix} \quad R_2 - (1)R_1$$

$$= \begin{bmatrix} 2 & -1 & 3 & 2 \\ 0 & 3 & -3 & 2 \\ 1 & 1 & -2 & 2 \\ 1 & 3 & 4 & -1 \end{bmatrix} \quad R_3 - (\tfrac{1}{2})R_1$$

$$= \begin{bmatrix} 2 & -1 & 3 & 2 \\ 0 & 3 & -3 & 2 \\ 0 & \frac{3}{2} & \frac{-7}{2} & 1 \\ 1 & 3 & 4 & -1 \end{bmatrix} \quad R_4 - (\tfrac{1}{2})R_1$$

$$= \begin{bmatrix} 2 & -1 & 3 & 2 \\ 0 & 3 & -3 & 2 \\ 0 & \frac{3}{2} & \frac{-7}{2} & 1 \\ 0 & \frac{7}{2} & \frac{5}{2} & -2 \end{bmatrix} \quad R_3 - \tfrac{1}{2}R_2$$

$$= \begin{bmatrix} 2 & -1 & 3 & 2 \\ 0 & 3 & -3 & 2 \\ 0 & 0 & -2 & 0 \\ 0 & \frac{7}{2} & \frac{5}{2} & -2 \end{bmatrix} \quad R_4 - \tfrac{7}{6}R_2$$

$$= \begin{bmatrix} 2 & -1 & 3 & 2 \\ 0 & 3 & -3 & 2 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 6 & \frac{-13}{3} \end{bmatrix} \quad R_4 - (-3)R_3$$

$$= \begin{bmatrix} 2 & -1 & 3 & 2 \\ 0 & 3 & -3 & 2 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & \frac{-13}{3} \end{bmatrix} \quad U$$

$$\Rightarrow L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & \frac{7}{6} & -3 & 1 \end{bmatrix}$$

$$\Rightarrow L' = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 1 & \frac{7}{3} & -6 & 2 \end{bmatrix} \#$$

$$U' = \begin{bmatrix} 1 & \frac{-1}{2} & \frac{3}{2} & 1 \\ 0 & \frac{3}{2} & \frac{-3}{2} & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & \frac{-13}{6} \end{bmatrix} \#$$

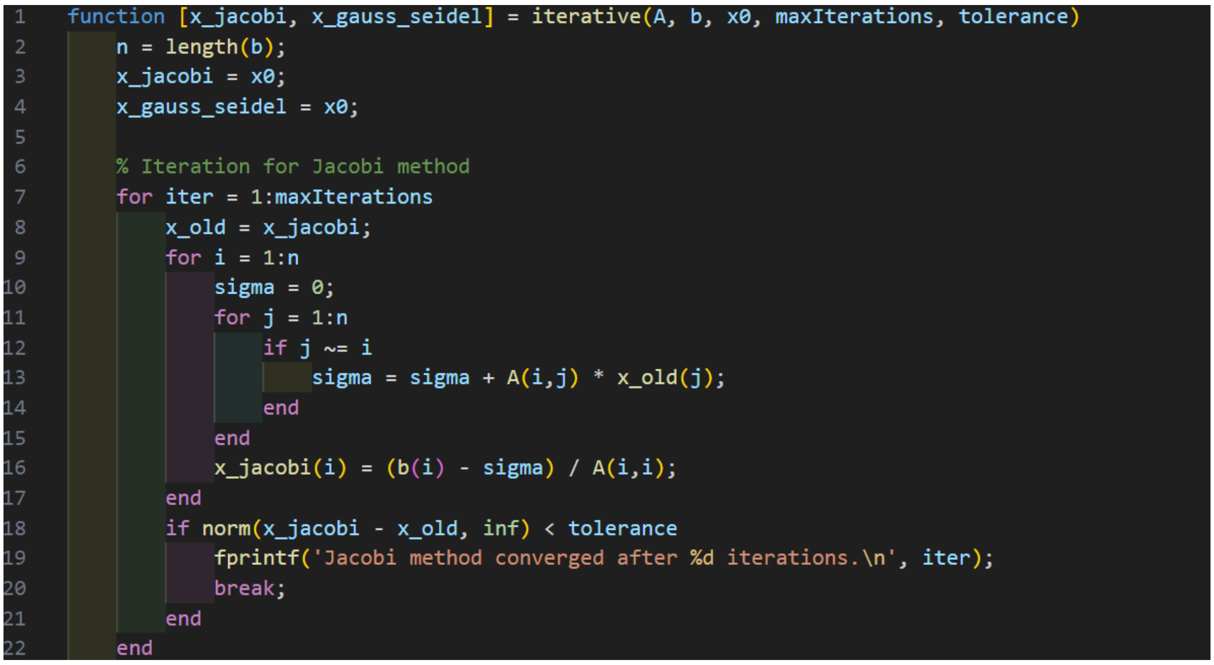
4. (20%) Solve this system with the Jacobi method. First rearrange to make it diagonally dominant if possible. Use [0, 0, 0] as the starting vector. How many iterations to get the solution accurate to five significant digits?

$$\begin{bmatrix} 7 & -3 & 4 & 6 \\ -3 & 2 & 6 & 2 \\ 2 & 5 & 3 & -5 \end{bmatrix}$$

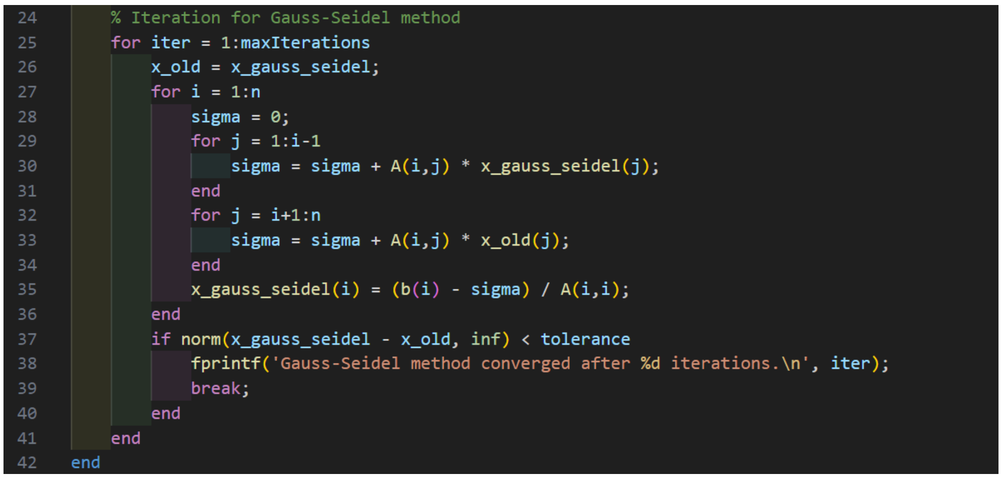
5. (20%) Repeat Problem 4 with the Gauss-Seidel method. Are fewer iterations required?

I want to answer the question 4 and question 5 together.

- Here is the code.
- We need to interchange the row 2 and row3.
- In this function, i devide to two parts, one is the jacobi methods another is gauss-seidel methods.

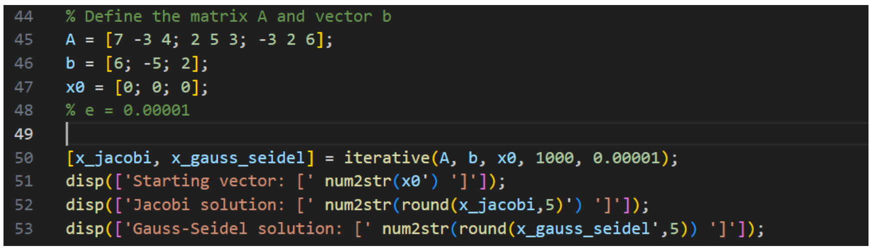


```
function [x_jacobi, x_gauss_seidel] = iterative(A, b, x0, maxIterations, tolerance)
    n = length(b);
    x_jacobi = x0;
    x_gauss_seidel = x0;

    % Iteration for Jacobi method
    for iter = 1:maxIterations
        x_old = x_jacobi;
        for i = 1:n
            sigma = 0;
            for j = 1:n
                if j ~= i
                    sigma = sigma + A(i,j) * x_old(j);
                end
            end
            x_jacobi(i) = (b(i) - sigma) / A(i,i);
        end
        if norm(x_jacobi - x_old, inf) < tolerance
            fprintf('Jacobi method converged after %d iterations.\n', iter);
            break;
        end
    end
```



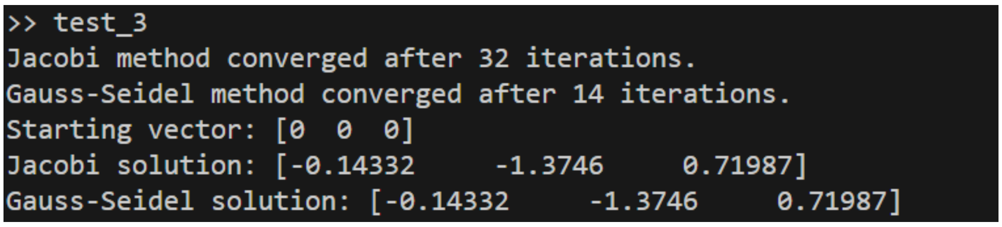
```matlab
    % Iteration for Gauss-Seidel method
    for iter = 1:maxIterations
        x_old = x_gauss_seidel;
        for i = 1:n
            sigma = 0;
            for j = 1:i-1
                sigma = sigma + A(i,j) * x_gauss_seidel(j);
            end
            for j = i+1:n
                sigma = sigma + A(i,j) * x_old(j);
            end
            x_gauss_seidel(i) = (b(i) - sigma) / A(i,i);
        end
        if norm(x_gauss_seidel - x_old, inf) < tolerance
            fprintf('Gauss-Seidel method converged after %d iterations.\n', iter);
            break;
        end
    end
end
```

- Finally, print the output of the interation numbers and solution in jacobi method and gauss-seidel methods.



```matlab
% Define the matrix A and vector b
A = [7 -3 4; 2 5 3; -3 2 6];
b = [6; -5; 2];
x0 = [0; 0; 0];
% e = 0.00001

[x_jacobi, x_gauss_seidel] = iterative(A, b, x0, 1000, 0.00001);
disp(['Starting vector: [' num2str(x0') ']']);
disp(['Jacobi solution: [' num2str(round(x_jacobi,5)') ']']);
disp(['Gauss-Seidel solution: [' num2str(round(x_gauss_seidel',5)) ']']);
```

- Here is the result of question 4 and question 5.



```
>> test_3
Jacobi method converged after 32 iterations.
Gauss-Seidel method converged after 14 iterations.
Starting vector: [0  0  0]
Jacobi solution: [-0.14332     -1.3746     0.71987]
Gauss-Seidel solution: [-0.14332     -1.3746     0.71987]
```

6. (25%) This 2×2 matrix is obviously singular and is almost diagonally dominant. If the right-hand-side vector is [0, 0], the equations are satisfied by any pair where x = y.

$$\begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$$

   (a) What happens if you use the Jacobi method with these starting vectors: [1, 1], [1, -1], [-1, 1], [2, 5], [5, 2]?

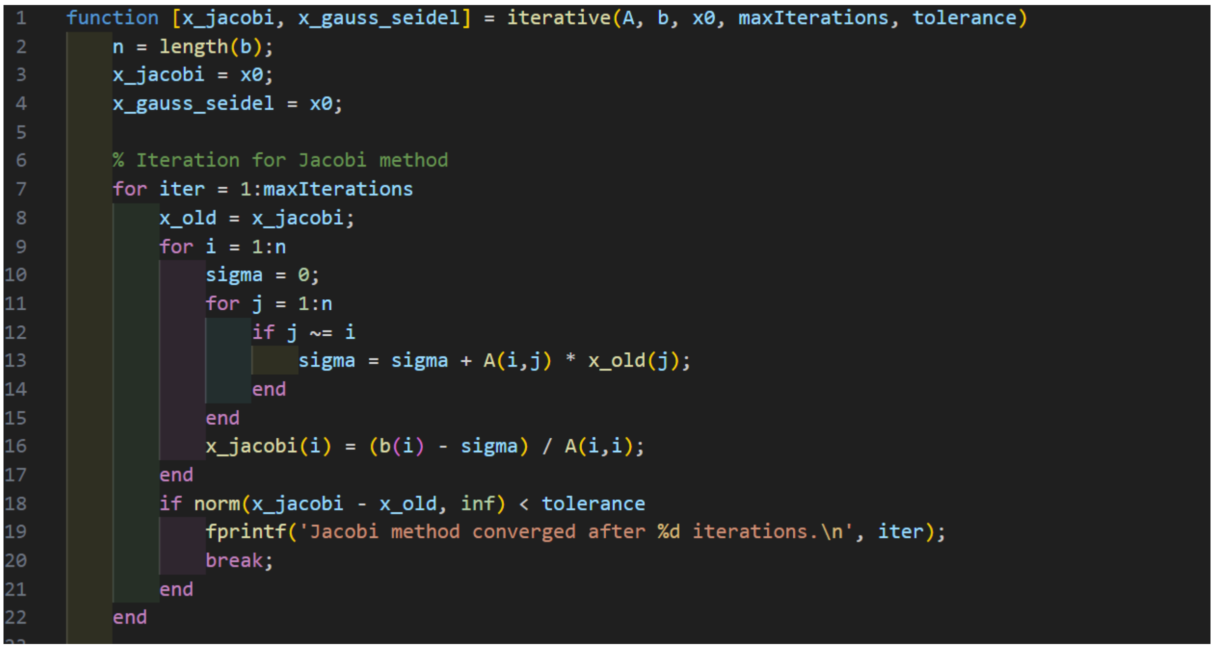
   (b) What happens if the Gauss-Seidel method is used with the same starting vectors as in part (a)?

   (c) If the elements whose values are -2 in the matrix are changed slightly, to -1.99, the matrix is no longer singular but is almost singular. Repeat parts (a) and (b) with these new matrix.

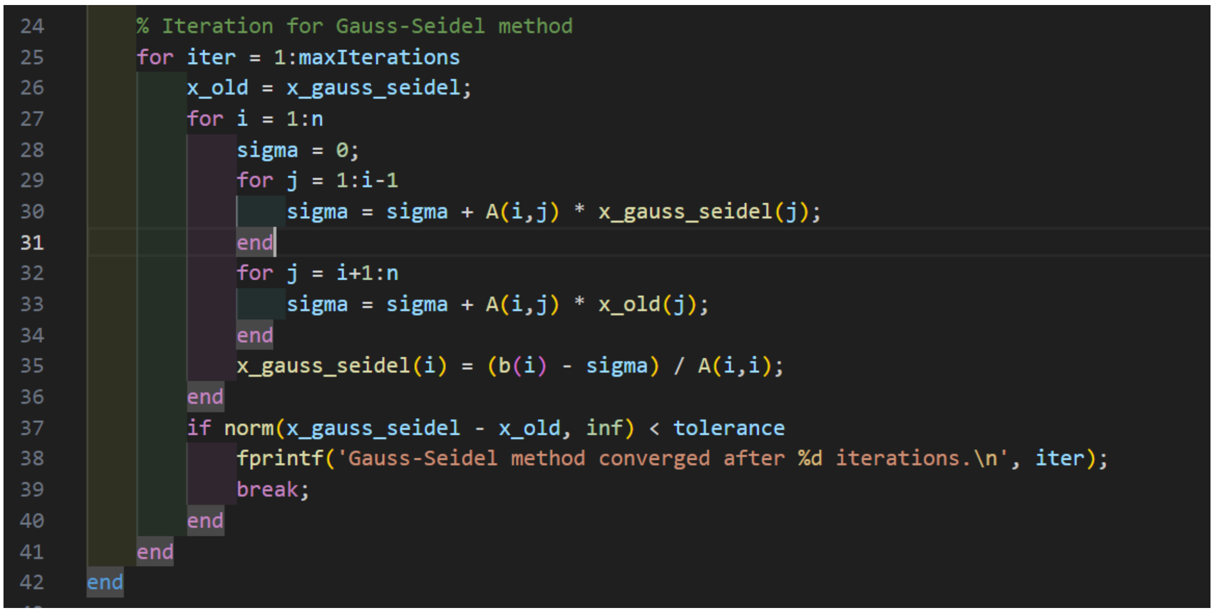
In part (a) and part (b), through the Jacobi and Gauss-Seidel methodthe, iterations will not converge to a unique solution since the matrix A is singular. The solutions may oscillate or diverge, depending on the starting vector. There is no unique solution to the system.

In part (c), we write a matlab code to do the iterations.

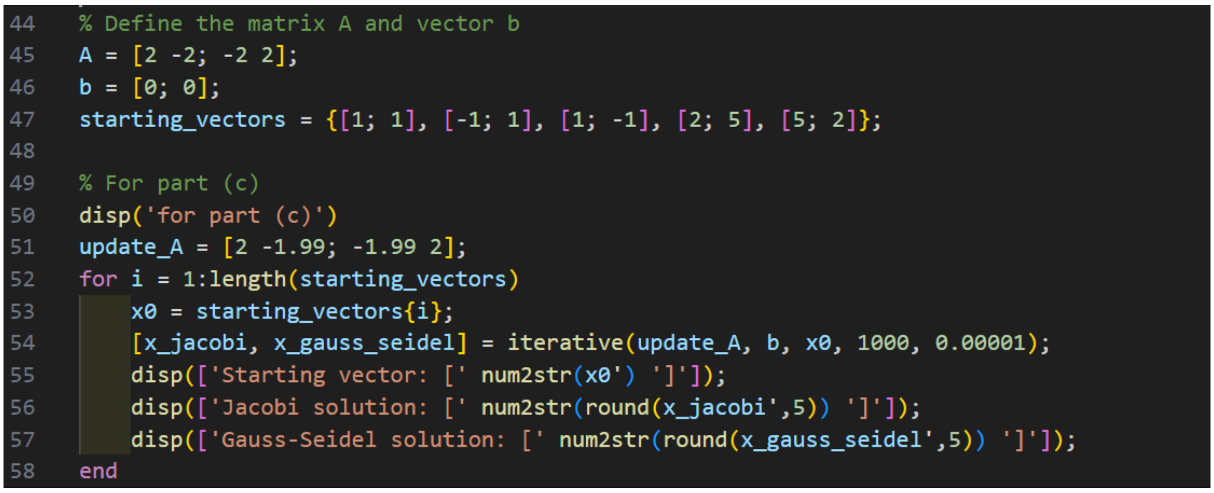
- Here is the code.



```
function [x_jacobi, x_gauss_seidel] = iterative(A, b, x0, maxIterations, tolerance)
    n = length(b);
    x_jacobi = x0;
    x_gauss_seidel = x0;

    % Iteration for Jacobi method
    for iter = 1:maxIterations
        x_old = x_jacobi;
        for i = 1:n
            sigma = 0;
            for j = 1:n
                if j ~= i
                    sigma = sigma + A(i,j) * x_old(j);
                end
            end
            x_jacobi(i) = (b(i) - sigma) / A(i,i);
        end
        if norm(x_jacobi - x_old, inf) < tolerance
            fprintf('Jacobi method converged after %d iterations.\n', iter);
            break;
        end
    end
```

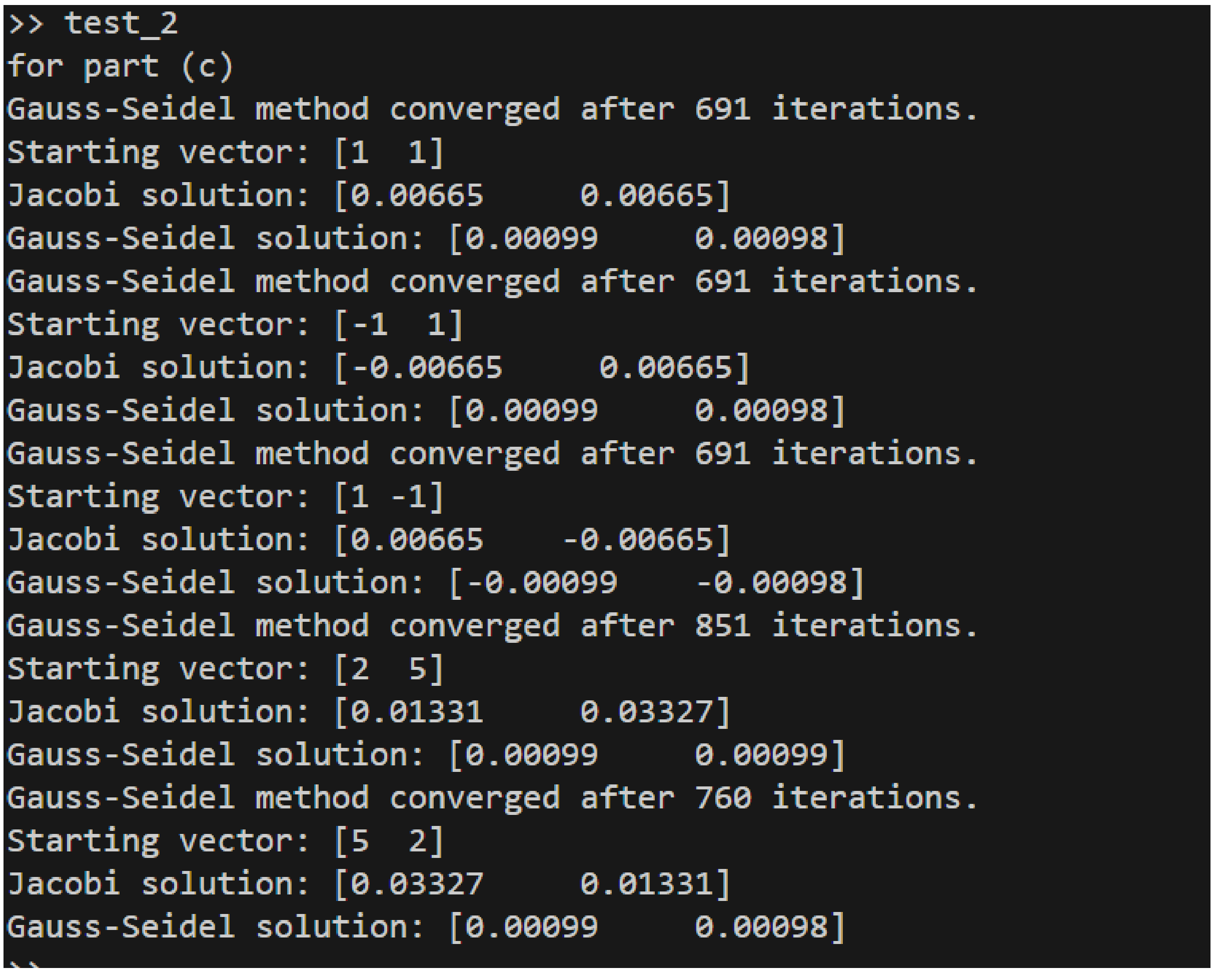


```
    % Iteration for Gauss-Seidel method
    for iter = 1:maxIterations
        x_old = x_gauss_seidel;
        for i = 1:n
            sigma = 0;
            for j = 1:i-1
                sigma = sigma + A(i,j) * x_gauss_seidel(j);
            end
            for j = i+1:n
                sigma = sigma + A(i,j) * x_old(j);
            end
            x_gauss_seidel(i) = (b(i) - sigma) / A(i,i);
        end
        if norm(x_gauss_seidel - x_old, inf) < tolerance
            fprintf('Gauss-Seidel method converged after %d iterations.\n', iter);
            break;
        end
    end
end
```



```matlab
% Define the matrix A and vector b
A = [2 -2; -2 2];
b = [0; 0];
starting_vectors = {[1; 1], [-1; 1], [1; -1], [2; 5], [5; 2]};

% For part (c)
disp('for part (c)')
update_A = [2 -1.99; -1.99 2];
for i = 1:length(starting_vectors)
    x0 = starting_vectors{i};
    [x_jacobi, x_gauss_seidel] = iterative(update_A, b, x0, 1000, 0.00001);
    disp(['Starting vector: [' num2str(x0') ']']);
    disp(['Jacobi solution: [' num2str(round(x_jacobi',5)) ']']);
    disp(['Gauss-Seidel solution: [' num2str(round(x_gauss_seidel',5)) ']']);
end
```

Here is the result of part (c).



```
>> test_2
for part (c)
Gauss-Seidel method converged after 691 iterations.
Starting vector: [1  1]
Jacobi solution: [0.00665     0.00665]
Gauss-Seidel solution: [0.00099     0.00098]
Gauss-Seidel method converged after 691 iterations.
Starting vector: [-1  1]
Jacobi solution: [-0.00665     0.00665]
Gauss-Seidel solution: [0.00099     0.00098]
Gauss-Seidel method converged after 691 iterations.
Starting vector: [1 -1]
Jacobi solution: [0.00665    -0.00665]
Gauss-Seidel solution: [-0.00099    -0.00098]
Gauss-Seidel method converged after 851 iterations.
Starting vector: [2  5]
Jacobi solution: [0.01331     0.03327]
Gauss-Seidel solution: [0.00099     0.00099]
Gauss-Seidel method converged after 760 iterations.
Starting vector: [5  2]
Jacobi solution: [0.03327     0.01331]
Gauss-Seidel solution: [0.00099     0.00098]
```