Numerical Analysis
Homework 4. Linear Iterative Methods

102061125 陳冠鈞

1. Objective

In this assignment, I will use different iterative methods to compute the node voltages, and compare the outputs with the LU decomposition method.

2. Approach

First, I form the linear system for a resistor network.

Algorithm. System Equation for a Resistor Network

Let the unknown vector be all node voltages, x_i, $i = 1,\ldots,n$.

Create an $n \times n$ matrix A and an n-vector b and initialize both to 0.

for each node i not connecting to voltage sources,

for each resistor, with conductance $g_k = 1/r_k$, connecting node i and j,

$$A_{ii} = A_{ii} + g_k,$$

$$A_{ij} = A_{ij} - g_k.$$

for each node i connecting to a fixed voltage V_i,

$$A_{ii} = 1,$$

$$A_{ij} = 0, \quad j \neq i,$$

$$b_i = V_i.$$

Second, formulate each iterative methods.

Algorithm. Jacobi Method

Input: initial guess $\mathbf{x}^{(0)}$ to the diagonal dominant matrix **A**, right-hand side vector **b**, converge criterion.

Output: solution when maxIteration or convergence is reached

$k = 0$
while (k < maxIteration) **and** (convergence not reached) **do**
  for i := 1 **to** n **do**
    $\sigma = 0$
    for j := 1 **to** n **do**
      if i ≠ j **then**
        $\sigma = \sigma + a_{ij}x_j^{(k)}$
      end
    end
    $x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sigma)$
  end
  $k = k + 1$
end

Algorithm. Gauss-Seidel Method

Input: initial guess $\mathbf{x}^{(0)}$ to the diagonal dominant matrix **A**, right-hand side vector **b**, converge criterion.

Output: solution when maxIteration or convergence is reached

$x^{(1)} = x^{(0)}$

$k = 0$

while (k < maxIteration) **and** (convergence not reached) **do**

  for i := 1 **to** n **do**

    $\sigma = 0$

    for j := 1 **to** n **do**

      if i ≠ j **then**

        $\sigma = \sigma + a_{ij} x_j^{(k+1)}$

      end

    end

    $x_i^{(k+1)} = \frac{1}{a_{ii}} (b_i - \sigma)$

  end

  $k = k + 1$

end

Algorithm. Symmetric Gauss-Seidel Method

Input: initial guess $\mathbf{x}^{(0)}$ to the diagonal dominant matrix **A**, right-hand side vector **b**, converge criterion.

Output: solution when maxIteration or convergence is reached

$x^{(1)} = x^{(0)}$

$k = 0$

while (k < maxIteration) **and** (convergence not reached) **do**

  for i := 1 **to** n **do**

    $\sigma = 0$

    for j := 1 **to** n **do**

      if i ≠ j **then**

        $\sigma = \sigma + a_{ij} x_j^{(k+1)}$

      end

    end

    $x_i^{(k+1)} = \frac{1}{a_{ii}} (b_i - \sigma)$

  end

  for i := n **to** 1 **do**

    $\sigma = 0$

    for j := 1 **to** n **do**

      if i ≠ j **then**

        $\sigma = \sigma + a_{ij} x_j^{(k+1)}$

      end

    end

    $x_i^{(k+1)} = \frac{1}{a_{ii}} (b_i - \sigma)$

  end

  $k = k + 1$

end

Third, I want to find out the tolerance that enables solution accuracy of 10^{-7} volts, so I write another function finding the tolerance.
Take Jacobi method for example.

Algorithm. Jacobi Method using 1-norm for tolerance checking

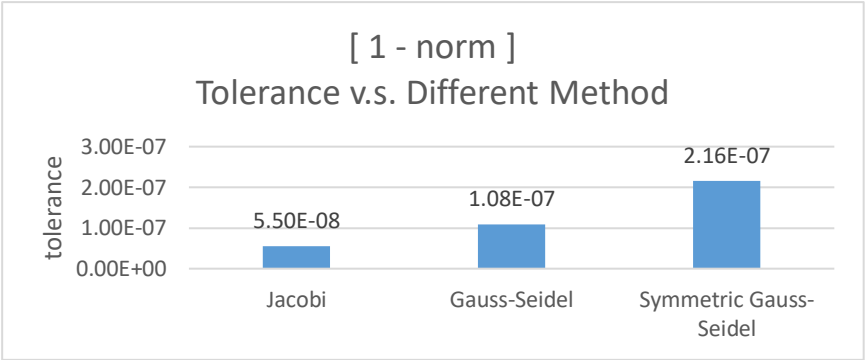
Input: system matrix **A**, right-hand side vector **b**, initial guess $\mathbf{x^{(0)}}$, and $\mathbf{x_{LU}}$ solved using LU decomposition.

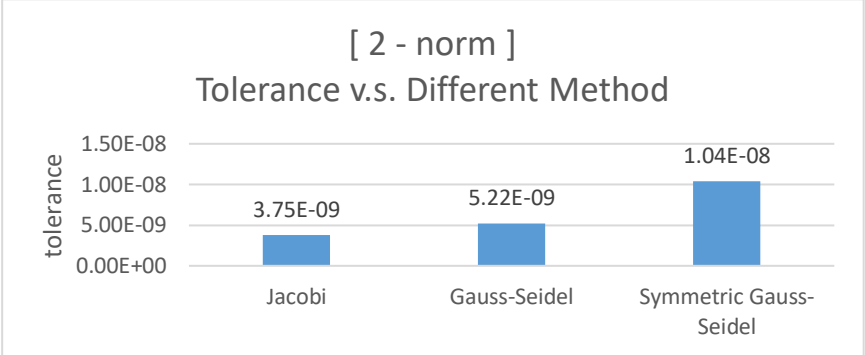
Output: the tolerance

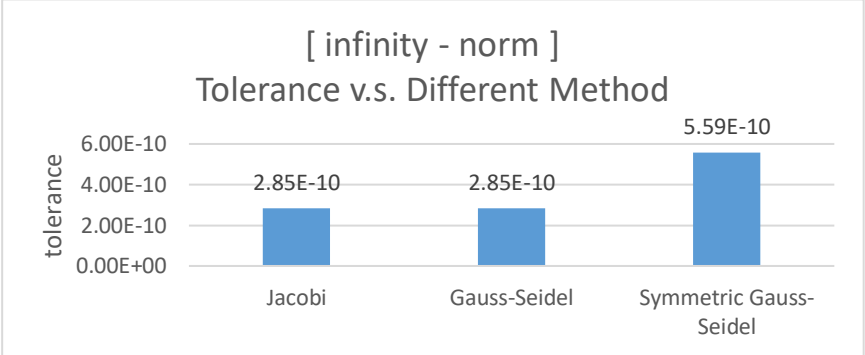
while $(||\mathrm{x_{LU}} - \mathrm{x^{(k)}}||_{infinity} > 10^{-7})$ **do**
  for i := 1 **to** n **do**
    $\sigma = 0$
    for j := 1 **to** n **do**
      if i ≠ j **then**
        $\sigma = \sigma + \mathrm{a}_{ij}\mathrm{x}_j{}^{(k)}$
      end
    end
    $x_i{}^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sigma)$
  end
  $k = k + 1$
end
$tolerance = ||x_{LU} - x^{(k)}||_1$
return tolerance

3. Results

- **Tolerance**



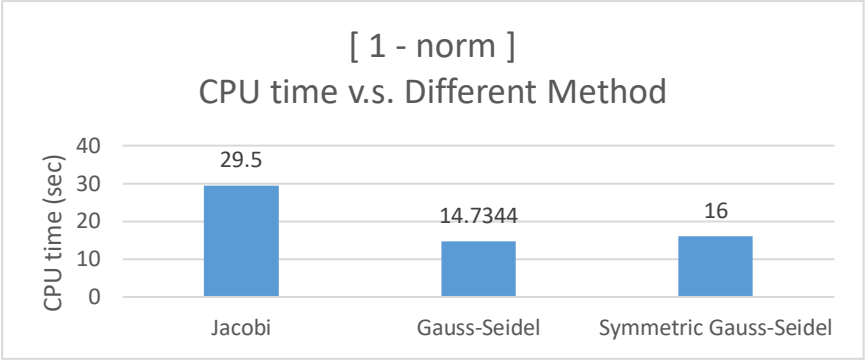


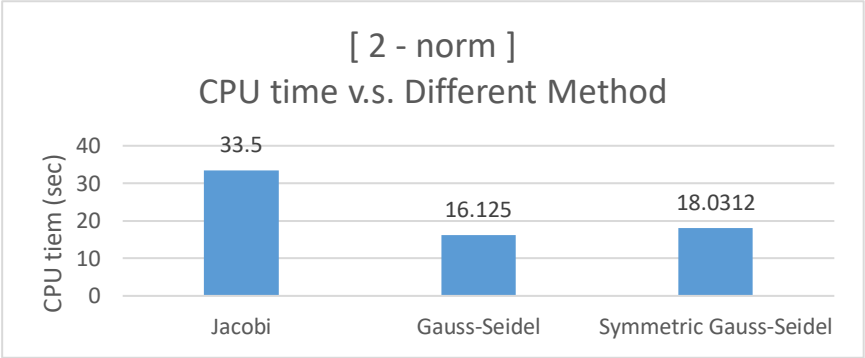


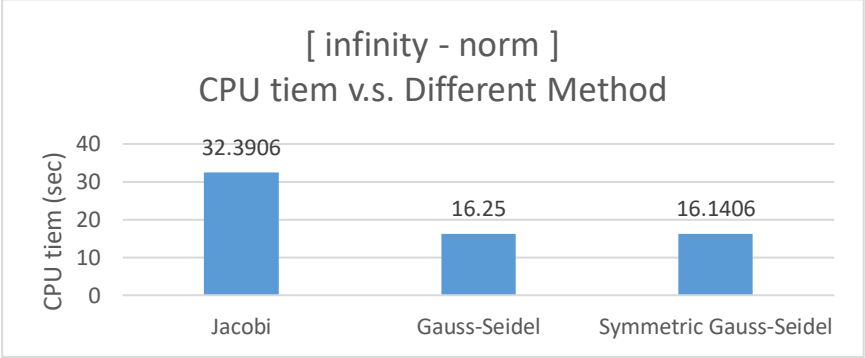
- **Different Method (same norm)**

  If we want to reach the same accuracy of 10^{-7} volts, tolerance: **Jacobi** < **Gauss-Seidel** < **Symmetric Gauss-Seidel**. It depends on the convergence rate, i.e. **Gauss-Seidel** has better convergence rate than **Jacobi**, and **Symmetric Gauss-Seidel** has better convergence rate than **Gauss-Seidel**.

- **CPU time**



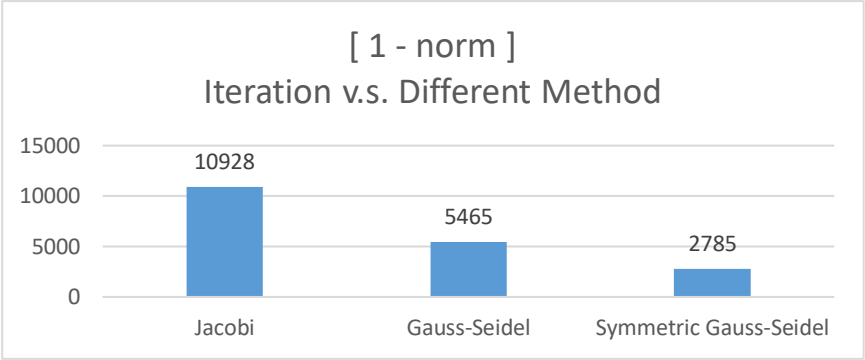


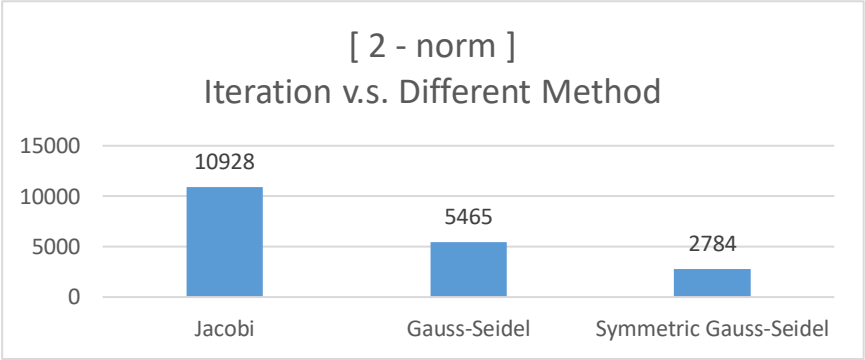


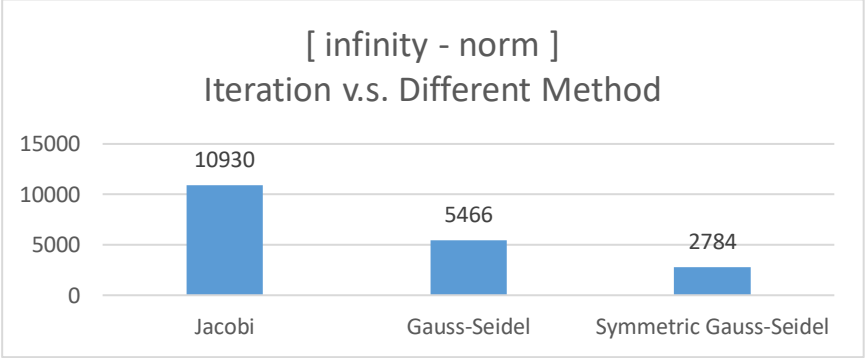
  - **Different Method (same norm)**

    Gauss-Seidel and **Symmetric Gauss-Seidel** method obviously spend less CPU time to compute the output than **Jacobi** method. But the difference of CPU time between **Gauss-Seidel** and **Symmetric Gauss-Seidel** method are not so much.

- **Iteration**





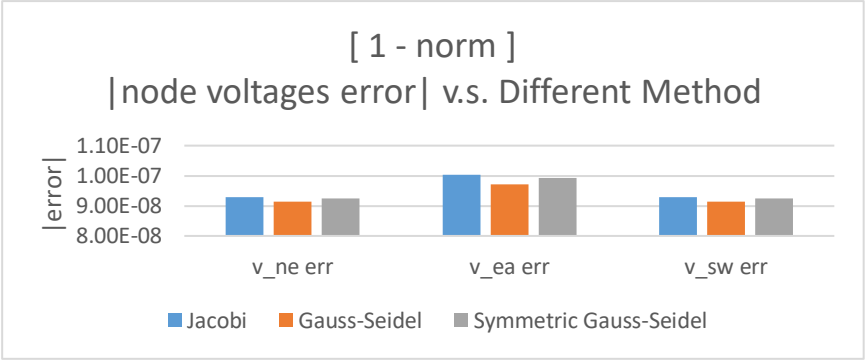


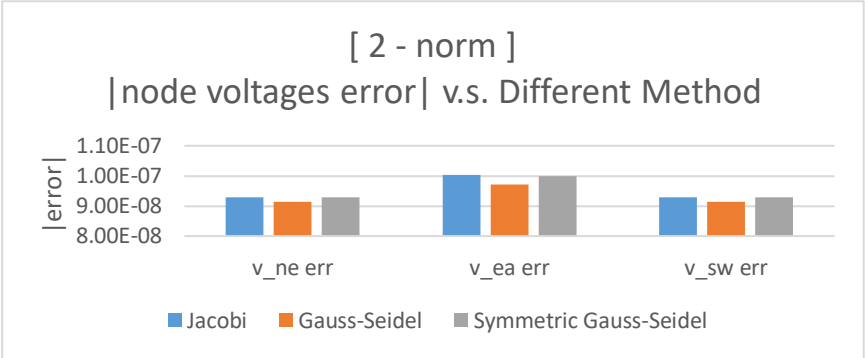
  - **Different Method (same norm)**

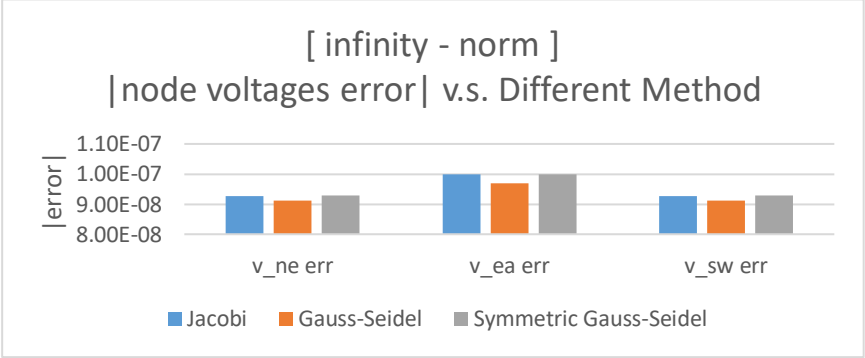
    Gauss-Seidel method has the same iteration process as **Jacobi** method, and it uses updated values whenever possible. **Symmetric Gauss-Seidel** method adds backward **Gauss-Seidel** method in same iteration, so it appears to converge faster (iteration) than the **Gauss-Seidel** method.

    So, I can roughly say that the iteration of **Gauss-Seidel** is half of **Jacobi**, and the iteration of **Symmetric Gauss-Seidel** is half of **Gauss-Seidel**.

- Error



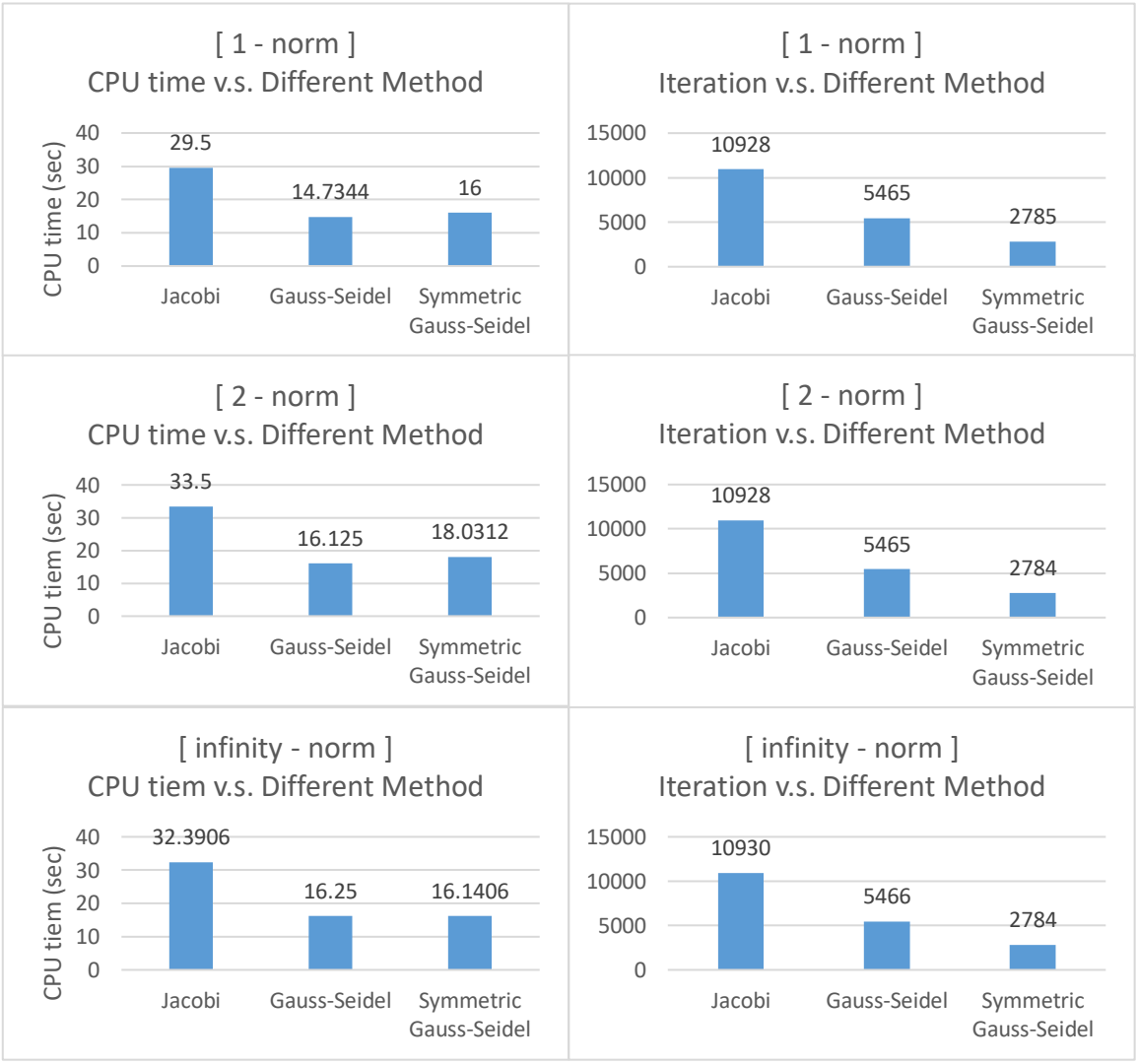




- **The Voltage Error**

  When we use the tolerance we find in different method for converge condition, all voltage error are smaller than 10^{-7}.

- **CPU time v.s. Iteration**



  - **Different Method (same norm)**
    Although CPU time is very close between **Gauss-Seidel** method and **Symmetric Gauss-Seidel** method, the iteration is very different between them.

4. Conclusion

1. If we compare the iteration of three different method, **Symmetric Gauss-Seidel** method has the best convergence rate, and **Jacobi** method has the worst convergence rate.
2. **1-norm** is the sum of the absolute values of the vector; **2-norm** is the square root of the inner product of the vector; **infinity-norm** is the maximum component after taking absolute. If we regarding this as an optimization problem, using **1-norm** or **2-norm**, the components will influence each other; however, using **infinity-norm**, it only considers the maximum component.
   Consequently, the tolerance using **infinity-norm** should be the smallest of three different norm for its strict constraint in this problem.

For my own applications, I prefer using **Symmetric Gauss-Seidel** method and using **2-norm**. **Symmetric Gauss-Seidel** has better convergence rate; **2-norm** is the most well-known application in the signal processing field, as known as Mean-Square Error (MSE) measurement, and it is used as a standard quantity for measuring a vector difference.