1 Experiment No. 6

2 Experiment Title

Solving system of linear equations by Jacobi and Gauss-Seidel method using MATLAB

3 Objective

The objectives of this lab are:

- To gather knowledge about solving a system of linear equations using the Jacobi and Gauss-Seidel Method.
- To implement and show output using the Jacobi and Gauss-Seidel Method for solving linear equations.

4 Theory

4.1 Jacobi Method

The Jacobi method is an iterative technique for solving systems of linear equations. Named after Carl Gustav Jacob Jacobi (1804–1851), this method is based on two key assumptions:

- The system has a unique solution.
- The coefficient matrix A has no zero entries on its main diagonal. If any diagonal element is zero, rows or columns must be interchanged to ensure all diagonal entries are nonzero.

Consider a general system of n linear equations:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \quad (6.1.1)$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \quad (6.1.2)$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \quad (6.1.n)$$

To apply the Jacobi method, solve each equation for its respective variable:

$$x_1 = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3 - \cdots - \frac{a_{1n}}{a_{11}}x_n \quad (6.2.1)$$

$$x_2 = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}}x_1 - \frac{a_{23}}{a_{22}}x_3 - \cdots - \frac{a_{2n}}{a_{22}}x_n \quad (6.2.2)$$

$$\vdots$$

$$x_n = \frac{b_n}{a_{nn}} - \frac{a_{n1}}{a_{nn}}x_1 - \frac{a_{n2}}{a_{nn}}x_2 - \cdots - \frac{a_{n(n-1)}}{a_{nn}}x_{n-1} \quad \text{(6.2.n)}$$

Start with an initial guess:

$$\mathbf{x}^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, \ldots, x_n^{(0)}\right)$$

Use these values in the reformulated equations to obtain the first approximation:

$$\mathbf{x}^{(1)} = \left(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \ldots, x_n^{(1)}\right)$$

Repeat this process to obtain the second approximation:

$$\mathbf{x}^{(2)} = \left(x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, \ldots, x_n^{(2)}\right)$$

By continuing this process, a sequence of approximations is formed:

$$\mathbf{x}^{(m)} = \left(x_1^{(m)}, x_2^{(m)}, x_3^{(m)}, \ldots, x_n^{(m)}\right)$$

which eventually converges to the true solution of the system.

4.2 Gauss-Seidel Method

The Gauss-Seidel method, developed by Carl Friedrich Gauss (1777–1855) and Philipp L. Seidel (1821–1896), is an improvement over the Jacobi method. It uses the most recently calculated values during the iteration process.
Unlike the Jacobi method, which waits to update all values simultaneously, the Gauss-Seidel method updates each variable as soon as it is computed. For instance, once x_1 is computed from the first equation, it is immediately used to compute x_2 from the second equation. Similarly, updated values of x_1 and x_2 are used to compute x_3, and so on.
This strategy of immediate substitution often leads to faster convergence, particularly in systems with diagonally dominant coefficient matrices.

5 Algorithm

5.1 Jacobi Method

Algorithm: Jacobi Method for Solving System of Linear Equations
Input: Augmented matrix a of size $n \times (n+1)$, where A is the coefficient matrix and b is the constant vector. **Output:** Solution vector x such that $Ax = b$.

1. **Initialize and Define Inputs:**
   (a) Define matrix a to represent the augmented matrix of the system of linear equations.
   (b) Extract coefficient matrix A (all columns of a except the last).
   (c) Extract constant vector b (last column of a).
2. **Verification Using Matrix Inversion:**

   (a) Compute

$$X = A^{-1}b$$

   for future verification.

3. **Initialize Guess Vectors:**
   (a) Set the initial guess for the solution vector x as a zero vector of size n.
   (b) Set the initial guess for the intermediate solution vector x_0 as a zero vector of size n.
4. **Print Header for Iteration Output:**
   (a) Display the column headers for the iteration number and solution variables.
5. **Jacobi Iteration Loop:**
   (a) For each iteration k from 1 to I_{max}:
      i. *Row-wise Solution Update:*
         A. For each row i from 1 to n:
            - Initialize sum $= 0$.
            - For each column j from 1 to n:
               – If $i \neq j$ (exclude diagonal elements):
               – Add the term $\left(\frac{a[i,j]}{a[i,i]}\right) x[j]$ to sum.
            - Compute the i^{th} element of the intermediate solution:

$$x_0(i) = \frac{a[i, n+1]}{a[i,i]} - \text{sum}$$

      ii. *Update Solution Vector:*
         A. Set $x = x_0$
      iii. *Print Current Iteration Results:*
         A. Display the current iteration number k and the values of the solution vector x.
6. **End Iteration Loop.**

5.2 Gauss-Seidel Method

Algorithm: Gauss-Seidel Method for Solving System of Linear Equations
Input: Augmented matrix a of size $n \times (n+1)$, where A is the coefficient matrix and b is the constant vector. **Output:** Solution vector x such that $Ax = b$.

1. **Initialize and Define Inputs:**
   (a) Define matrix a to represent the augmented matrix of the system of linear equations.
   (b) Extract coefficient matrix A (all columns of a except the last).
   (c) Extract constant vector b (last column of a).
2. **Verification Using Matrix Inversion:**

   (a) Compute

$$X = A^{-1}b$$

   for future verification.

3. **Initialize Guess Vectors:**
   (a) Set the initial guess for the solution vector x as a zero vector of size n.
   (b) Set the initial guess for the intermediate solution vector x_0 as a zero vector of size n.

4. **Print Header for Iteration Output:**
   (a) Display the column headers for the iteration number and solution variables.

5. **Gauss-Seidel Iteration Loop:**
   (a) For each iteration k from 1 to I_{max}:
      i. *Row-wise Solution Update:*
         A. For each row i from 1 to n:
            - Initialize sum $= 0$.
            - For each column j from 1 to n:
               – If $i \neq j$ (exclude diagonal elements):
               – Add the term $\left(\frac{a[i,j]}{a[i,i]}\right) x[j]$ to sum.
            - Compute the i^{th} element of the solution:

$$x(i) = \frac{a[i, n+1]}{a[i, i]} - \text{sum}$$

      ii. *Print Current Iteration Results:*
         A. Display the current iteration number k and the values of the solution vector x.

6. **End Iteration Loop.**

5.3 Solving Non-linear Equation Using Jacobi Method

5.3.1 MATLAB Code:

```matlab
a = [3 -0.1 -0.2 7.85;
0.1 7 -0.3 -19.3;
0.3 -0.2 10 71.4];
tol = 0.000001;
n = size(a, 1);
A = a(:, 1:n);
b = a(:, end);
X_actual = inv(A) * b;

x = zeros(n, 1);
x0 = zeros(n, 1);
Imax = 100;

fprintf('\n%-10s', 'Iter');
for i = 1:n
fprintf('x(%d)        ', i);
end
fprintf('\n');

% Step 5: Jacobi Iteration Loop
for k = 1:Imax
for i = 1:n
s = 0;
for j = 1:n
if i ~= j
s = s + (A(i, j) / A(i, i)) * x(j);
end
end
x0(i) = (b(i) / A(i, i)) - s ;
end
x = x0;

fprintf('%-10d', k);
fprintf('%-12.6f', x);
fprintf('\n');

if abs(x - X_actual) < tol
fprintf('Converged to actual solution at iteration %d\n', k);
break;
end
end
disp(X_actual);
```

Listing 1: Solving Non-linear Equation Using Jacobi Method in MATLAB.

5.4 Result Shown in Command Window

```
>> jacobi
Iter      x(1)        x(2)        x(3)
1         2.616667    -2.757143   7.140000
2         3.000762    -2.488524   7.006357
3         3.000806    -2.499738   7.000207
4         3.000022    -2.500003   6.999981
5         2.999999    -2.500001   6.999999
6         3.000000    -2.500000   7.000000
Converged to actual solution at iteration 6
3.0000
-2.5000
```

Listing 2: Command Window for Jacobi Method

5.5 Solving Non-linear Equation Using Gauss-Seidel Method

5.5.1 MATLAB Code:

```

a = [3 -0.1 -0.2 7.85;
0.1 7 -0.3 -19.3;
0.3 -0.2 10 71.4];

tol = 0.000001;
n = size(a, 1);
A = a(:, 1:n);
b = a(:, end);
X_actual = inv(A) * b;
x = zeros(n, 1);

for k = 1:100
for i = 1:n
s = 0;
for j = 1:n
if i ~= j
s = s + (A(i,j) / A(i,i)) * x(j);
end
end
x(i) = (b(i) / A(i,i)) - s;
end

fprintf('%-10d', k);
fprintf('%-12.6f', x);
fprintf('\n');

if abs(x - X_actual) < tol
fprintf('Converged to actual solution at iteration %d\n', k);
break;
end
end
disp(X_actual);
```

Listing 3: Solving Non-linear Equation Using Gauss-Seidel Method in MATLAB.

5.6 Result Shown in Command Window

```
>> gaussseidel
1         2.616667    -2.794524   7.005610
2         2.990557    -2.499625   7.000291
3         3.000032    -2.499988   6.999999
4         3.000000    -2.500000   7.000000
Converged to actual solution at iteration 4
3.0000
-2.5000
7.0000
```

Listing 4: Command Window for Gauss-Seidel Method

6 Discussion

In this experiment, the Jacobi and Gauss-Seidel methods were used to solve systems of linear equations. Both methods were implemented in MATLAB and tested on standard systems. Initial guesses were assumed, and iterative updates were performed.
The Jacobi method was observed to converge more slowly which was iterated about 6 times, while the Gauss-Seidel method showed faster convergence due to immediate value updates which was about 4 iterations. The results from both methods were verified with MATLAB's built-in solver and used tolerance level was 0.000001 that found to be accurate.