UNITED INTERNATIONAL UNIVERSITY

Department of Electrical and Electronics Engineering

EEE 3200: Numerical Techniques Lab

FALL 2023

Group- 08

[Project Report]

Submitted by

	ID	Name
1	021191058	Ayman Zafar
2	021211018	Noushadul Alam
3	021211020	Sayem Hossain Rafi

Submitted to

RaBr Mr. Raiyan Basher
Lecture
Dep. Of EEE

Title: Load flow study using Gauss-Seidel method and Newton Raphson methods

Introduction:

The Gauss-Seidel method is a numerical technique used to solve load flow analysis in power systems. It is an iterative method that starts with initial assumptions for the voltage magnitudes and phase angles at each bus, and then refines these values in each iteration until the solution converges. The method is advantageous because it requires simple calculations and less memory, making it useful for small to medium-sized systems. However, it may converge slowly or fail to converge for certain types of systems. It is commonly used in power system analysis software and is the focus of various educational materials and research papers.

Resources/Components:

- MATLAB
- Course Resources
- Internet

Methods:

Two known methods were followed in this project. They are-

- Gauss-Seidel method.
- Newton Raphson method.

Gauss-Seidel method

The Gauss-Seidel method is an iterative technique used for solving a system of linear equations. It's particularly useful in numerical analysis and computational mathematics. When applied to electrical power systems, specifically for calculating the voltage at a power bus (a node where multiple components are connected), it helps determine the steady-state voltages in a power network.

In power systems, the voltage at a bus is influenced by the power injections, line impedances, and other factors within the network. The basic idea behind the Gauss-Seidel method is to iteratively solve a set of equations until reaching an acceptable solution.

Newton Raphson method

The Newton-Raphson method is another iterative technique used to find successively better approximations to the roots of a real-valued function. It's widely applied in various fields of science and engineering for solving equations and finding the roots of nonlinear systems.

In the context of calculating the voltage of a power bus in an electrical network, the Newton-Raphson method can be used to determine the steady-state voltages in a power system.The Newton-Raphson method typically converges faster than the Gauss-Seidel method for many types of systems. However, it requires the calculation of the Jacobian matrix, which can be computationally intensive, especially for large power systems.

This method is powerful for solving nonlinear equations and is widely used in power system analysis to determine voltage profiles, power flows, and stability analysis, allowing engineers to understand and optimize the performance of electrical networks.

Code:

```
% The Gauss-Seidel iteration

if z == 1
iteration = 0;
max_iterations = 1000;Z
del = 1; % Initialize a value larger than the tolerance
while del > 1e-4 && iteration < max_iterations
    iteration = iteration + 1;
    %P-Q buses
    for i = 2:4
        tmp1 = (p(i) - 1i * q(i)) / conj(v(i));
        tmp2 = 0;
        for k = 1:5
            if i == k
                tmp2 = tmp2 + 0;
            else
                tmp2 = tmp2 + ybus(i, k) *v(k);
            end
        end
        v(i) = (tmp1 - tmp2) / ybus(i, i);
    end
    %P-V bus (Bus 5)
    q5 = 0;
    for i = 1:5
        q5 = q5 + ybus(5, i) *v(i);
    end
    q5 = -imag(conj(v(5)) * q5);
    tmp1 = (p(5) - 1i * q5) / conj(v(5));
    tmp2 = 0;
    for k = 1:4
        tmp2 = tmp2 + ybus(5, k) * v(k);
    end
```

```
    vt  = (tmp1  - tmp2) / ybus(5, 5);
    v (5) =  abs(vt) * exp(1i * angle(v(5)));
    % Calculate S for all buses
    S = zeros (5, 1);
    for i = 1:5
        sm = 0;
        for k = 1:5
            sm = sm + ybus(i, k) * v(k);
        end
        S(i) = conj(v(i)) * sm;
    end
    % Calculate the mismatch
    delp = p - real(S);
    delq = q + imag(S);
    delpq = [delp(2:5); delq(2:5)];
    del = max(abs(delpq));
    fprintf('Iteration: %d, Max Mismatch: %.6f\n', iteration, del);
end

% Display results
disp('Bus Voltages and Angles:')
for i = 1:5
    fprintf('Bus %d Voltage Magnitude: %.4f, Voltage Angle: %.4f degrees\n', ...
    i, abs(v(i)), rad2deg(angle(v(i))));
end
end
                        % The Newton-Raphson iteration

  if z == 2
  iteration = 0;
  max_iterations = 1000;
  tolerance = 1e-4;
  delta = inf;

    while delta > tolerance && iteration < max_iterations
        iteration = iteration + 1;

        % Construct Jacobian matrix
        J = zeros(10, 10); % Jacobian matrix size (5 buses, 2 unknowns per bus)
        F = zeros(10, 1); % Function evaluations

        for i = 1:5
            for j = 1:5
                if i ~= j
                    % Off-diagonal elements
                    J(i, j) = -imag(conj(v(i)) * ybus(i, j));
                    J(i+5, j+5) = -real(conj(v(i)) * ybus(i, j));
                else
                    % Diagonal elements
                    sm = 0;
                    for k = 1:5
                        if k ~= i
                            sm = sm + ybus(i, k) * v(k);
                        end
                    end
                    J(i, j) = real(conj(v(i)) * sm) + q(i);
                    J(i+5, j+5) = -imag(conj(v(i)) * sm) - p(i);
                end
            end
            % Function evaluations
            sm = 0;
            for k = 1:5
                if k ~= i
                    sm = sm + ybus(i, k) * v(k);
                end
```

```
            end
            F(i) = v(i) * conj(sm) - p(i) - 1i * q(i);
            F(i+5) = imag(v(i) * conj(sm)) - q(i) + 1i * p(i);
        end

        % Newton-Raphson step
        increment = J \ (-F);
        v = v + increment(1:5);
        th = th + increment(6:10);

        % Recalculate maximum mismatch
        S = zeros(5, 1);
        for i = 1:5
            sm = 0;
            for k = 1:5
                sm = sm + ybus(i, k) * v(k);
            end
            S(i) = conj(v(i)) * sm;
        end
        delp = p - real(S);
        delq = q + imag(S);
        delpq = [delp(2:5); delq(2:5)];
        delta = max(abs(delpq));

        fprintf('Iteration: %d, Max Mismatch: %.6f\n', iteration, delta);
    end

    % Display results
    disp('Bus Voltages and Angles:')
    for i = 1:5
        fprintf('Bus %d Voltage Magnitude: %.4f, Voltage Angle: %.4f degrees\n', ...
            i, abs(v(i)), rad2deg(angle(v(i))));
    end
    end
```

Input:

```
Command Window
  Which mathode you want
  For Gauss-Seidel Prees 1
  For raphson method solution Prees 2
  Input Only 1 or 2...1
  Input ybus in Matrix... [2.6923-13.4115i, -1.9231+9.61541i, 0, 0, -0.7692+3.8462i;
          -1.9231+9.6154i, 3.6538-18.1942i, -0.9615+4.8077i, 0, -0.7692+3.8462i;
          0, -0.9615+4.8077i, 2.2115-11.0027i, -0.7692+3.8462i, -0.4808+2.4030i;
          0, 0, -0.7692+3.8462i, 1.1538-5.6742i, -0.3846+1.9231i;
          -0.7692+3.8462i, -0.7692+3.8462i, -0.4808+2.4030i, -0.3846+1.9231i, 2.4030-23.6942i]
  Input p in Matrix... [0; -0.96; -0.35; -0.16; 0.24]
  Input q in Matrix... [0; 0.62; 0.14; 0.08; -0.35]
fx Input mv in Matrix... [1.05; 1; 1; 1; 1.02]
```

Result:

```
Iteration: 988, Max Mismatch: 4.087004
Iteration: 989, Max Mismatch: 4.086889
Iteration: 990, Max Mismatch: 4.086776
Iteration: 991, Max Mismatch: 4.086663
Iteration: 992, Max Mismatch: 4.086550
Iteration: 993, Max Mismatch: 4.086439
Iteration: 994, Max Mismatch: 4.086328
Iteration: 995, Max Mismatch: 4.086217
Iteration: 996, Max Mismatch: 4.086108
Iteration: 997, Max Mismatch: 4.085999
Iteration: 998, Max Mismatch: 4.085890
Iteration: 999, Max Mismatch: 4.085782
Iteration: 1000, Max Mismatch: 4.085675
Bus Voltages and Angles:
Bus 1 Voltage Magnitude: 1.0500, Voltage Angle: 0.0000 degrees
Bus 2 Voltage Magnitude: 0.9541, Voltage Angle: -5.7606 degrees
Bus 3 Voltage Magnitude: 0.8687, Voltage Angle: -8.2421 degrees
Bus 4 Voltage Magnitude: 0.8356, Voltage Angle: -8.5154 degrees
Bus 5 Voltage Magnitude: 0.7089, Voltage Angle: 0.0000 degrees
```

Comment:

The Gauss-Seidel method provide us very much accurate result where the magnitude is close to 1V. On the other hand, the output of Newton Raphson method is pretty much far from the required values.

Difference between Gauss Seidel and Newton Raphson:

	Gauss Seidel	Newton Raphson
Co-ordinates	Works well with rectangular coordinates.	Polar co-ordinates are preferred as rectangular coordinates occupy more memory.
Arithmetical operations	Least in number to complete one iteration.	Elements of jacobian to be calculated in each iteration

Time	Requires less time per iteration, but increases with an increase in the number of buses.	Time/iteration is 7 times of GS and increases with an increase in the number of buses.
Convergence	Linear convergence	Quadratic convergence
Slack bus selection	A large number, increases with an increase in buses.	Very less (3 to 5 only) for large system and is practically constant.
Accuracy	Less accurate	More accurate
Memory	Less memory because of the sparsity of the matrix.	Large memory even with compact storage scheme
Usage/application	Small size system	A large system, ill-conditioned problems, optimal load flow studies.
Programming Logic	Easy	Very difficult
Reliability	Reliable only for a small system.	Reliable only for a small system.

Conclusion:

In conclusion, the project demonstrated the importance of numerical methods in power system analysis and highlighted the advantages and limitations of the Gauss-Seidel and Newton-Raphson methods. The use of MATLAB allowed for efficient implementation and comparison of these methods, providing valuable insights into the performance of load flow studies.