ECE 488

Power System Analysis & Simulation

Load Flow and Fault Analysis of a 6-Bus Network

Course: Electrical Power Systems

Term: Spring 2026

Instructor: Dr. Ilamparithi Chevran

Group: Individual Project

Project Team

David Emelu V01025755

University of Victoria

Department of Electrical & Computer Engineering

March 29, 2026

Contents

List of Figures

List of Tables

1 Introduction

This report presents the analysis of a 6-bus power system network as part of the ECE 488 software project requirements. The objective is to perform load flow analysis using the Newton-Raphson method in MATLAB and verify the results using Power World simulation software. Additionally, a fault analysis is conducted for a bolted three-phase fault at bus 6 to determine post-fault voltages.

2 System Description

The system under analysis is a 6-bus power network belonging to an electric utility company. The system base is **100 MVA**. Bus 1 is designated as the slack bus.

2.1 Network Topology

The network contains three generator buses (1, 2, 3) and three load buses (4, 5, 6), interconnected by five transmission lines and two transformers.

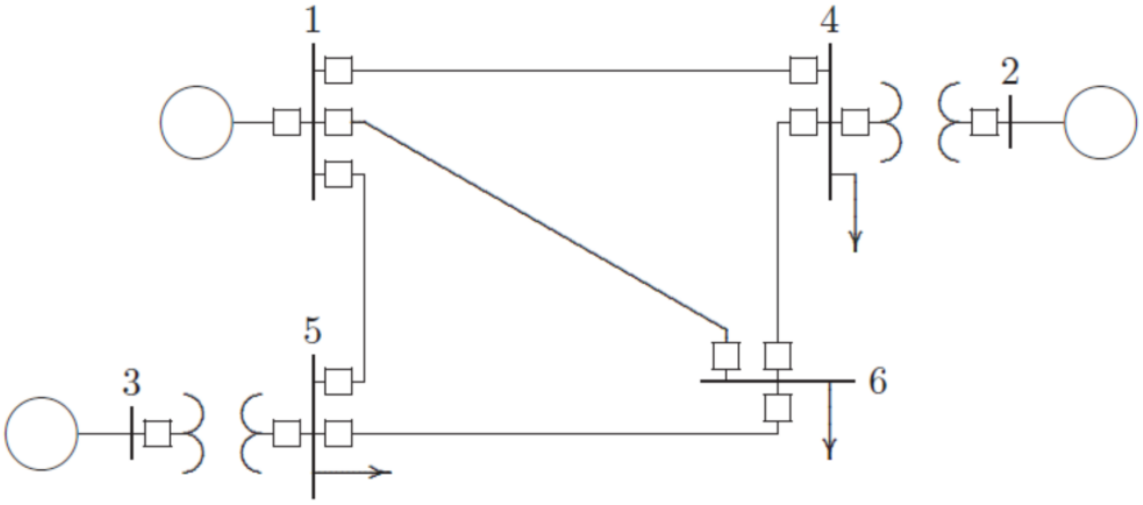


Figure 1: 6-bus Power System Network

Table 1: Line and Transformer Impedance Data (per-unit on 100-MVA base)

From Bus	To Bus	R (pu)	X (pu)	Type
1	4	0.035	0.225	Transmission Line
1	5	0.025	0.105	Transmission Line
1	6	0.040	0.215	Transmission Line
2	4	0.000	0.035	Transformer
3	5	0.000	0.042	Transformer
4	6	0.028	0.125	Transmission Line
5	6	0.026	0.175	Transmission Line

Table 2: Generator Data (100-MVA base)

Bus	Type	Voltage Setpoint (pu)	P Generation (MW)
1	Slack	1.060	
2	PV	1.040	150.0
3	PV	1.030	100.0

Table 3: Load Data

Bus	P Load (MW)	Q Load (Mvar)
1	0	0
2	0	0
3	0	0
4	100	70
5	90	30
6	160	110

2.2 Bus Type Classification

Table 4: Bus type classification and unknown state variables

Bus	Type	Known Quantities	Unknown Quantities
1	Slack	$\lvert V_1\rvert = 1.060$ pu, $\delta_1 = 0°$	P_1, Q_1
2	PV	$\lvert V_2\rvert = 1.040$ pu, $P_2 = +1.50$ pu	δ_2, Q_2
3	PV	$\lvert V_3\rvert = 1.030$ pu, $P_3 = +1.00$ pu	δ_3, Q_3
4	PQ	$P_4 = -1.00$ pu, $Q_4 = -0.70$ pu	$\lvert V_4\rvert$, δ_4
5	PQ	$P_5 = -0.90$ pu, $Q_5 = -0.30$ pu	$\lvert V_5\rvert$, δ_5
6	PQ	$P_6 = -1.60$ pu, $Q_6 = -1.10$ pu	$\lvert V_6\rvert$, δ_6

The Newton-Raphson solver has **8 unknown state variables**: angles $\delta_2, \delta_3, \delta_4, \delta_5, \delta_6$ for all non-slack buses, and magnitudes $|V_4|, |V_5|, |V_6|$ for PQ buses only.

3 Newton-Raphson Load Flow Analysis

3.1 Methodology and Unknown Variables

The Newton-Raphson (NR) method was used to solve the nonlinear power flow equations for the 6-bus network.

Y-Bus Formation

The bus admittance matrix Y_{bus} was constructed from the line and transformer data. For each branch connecting buses i and j with impedance $z_{ij} = r_{ij} + jx_{ij}$, the series admittance $y_{ij} = 1/z_{ij}$ is added to the diagonal elements Y_{ii} and Y_{jj}, and subtracted from the off-diagonal elements $Y_{ij} = Y_{ji}$.

Unknown State Variables

Based on bus classifications, the unknowns are:

- **Voltage angles (δ):** Buses 2, 3, 4, 5, and 6 (all non-slack buses) - 5 unknowns.
- **Voltage magnitudes ($|V|$):** Buses 4, 5, and 6 (PQ load buses) - 3 unknowns.

This gives 8 state variables solved using 5 real power (ΔP) and 3 reactive power (ΔQ) mismatch equations.

3.2 Convergence and Results

The Newton-Raphson algorithm successfully converged in **4 iterations** using a convergence tolerance of 1e-6 pu. The final solved state variables are displayed in the MATLAB console output below.

```
Newton-Raphson converged in 4 iteration(s). (tol = 1e-06 pu)

Bus   Magnitude (pu)   Angle (deg)
-----------------------------------
1     1.060000         0.000000
2     1.040000         1.472406
3     1.030000         0.817493
4     1.006840         -1.401486
5     1.014919         -1.485115
6     0.937973         -5.598790
```

The complete MATLAB code, including data initialization, Y-bus formation, and the commented iterative loop, is provided in Appendix A.1.

4 Power World Simulation (Load Flow)

4.1 Simulation Setup

The 6-bus network was modeled in Power World Simulator using the line, transformer, and generator data provided. The system base was set to 100 MVA.

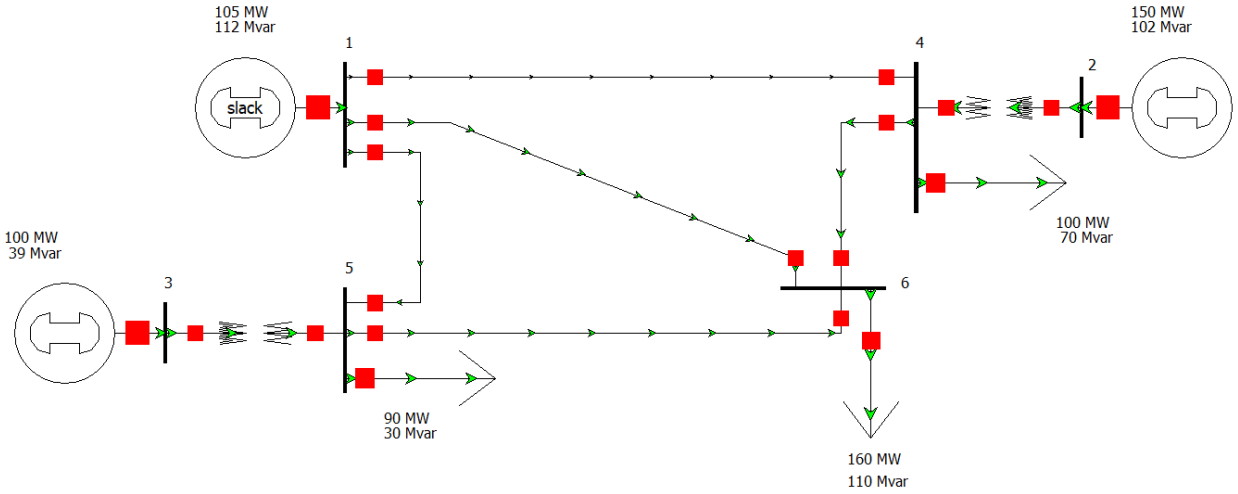
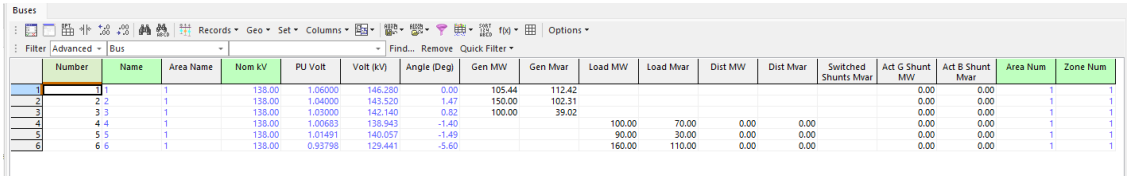


Figure 2: Power World Simulation Setup

4.2 Simulation Results

The load flow simulation was executed, and the results were recorded.



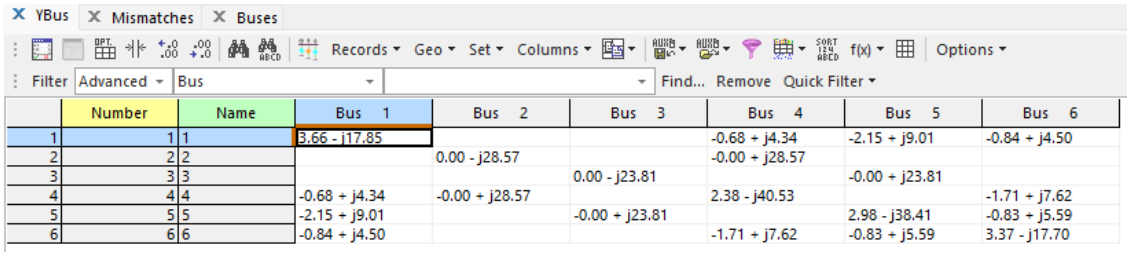
Buses

Records Geo Set Columns Options

Filter Advanced Bus Find... Remove Quick Filter

	Number	Name	Area Name	Nom kV	PU Volt	Volt (kV)	Angle (Deg)	Gen MW	Gen Mvar	Load MW	Load Mvar	Dist MW	Dist Mvar	Switched Shunts Mvar	Act G Shunt MW	Act B Shunt Mvar	Area Num	Zone Num
1	1	1	1	138.00	1.06000	146.280	0.00	105.44	112.42						0.00	0.00	1	1
2	2	2	1	138.00	1.04000	143.520	1.47	150.00	102.31						0.00	0.00	1	1
3	3	3	1	138.00	1.03000	142.140	0.82	100.00	39.02						0.00	0.00	1	1
4	4	4	1	138.00	1.00683	138.943	-1.40			100.00	70.00	0.00	0.00		0.00	0.00	1	1
5	5	5	1	138.00	1.01491	140.057	-1.49			90.00	30.00	0.00	0.00		0.00	0.00	1	1
6	6	6	1	138.00	0.93798	129.441	-5.60			160.00	110.00	0.00	0.00		0.00	0.00	1	1

Figure 3: Power World Simulation Bus Results



YBus Mismatches Buses

Records Geo Set Columns Options

Filter Advanced Bus Find... Remove Quick Filter

	Number	Name	Bus 1	Bus 2	Bus 3	Bus 4	Bus 5	Bus 6
1	1	1	3.66 - j17.85			-0.68 + j4.34	-2.15 + j9.01	-0.84 + j4.50
2	2	2		0.00 - j28.57		-0.00 + j28.57		
3	3	3			0.00 - j23.81		-0.00 + j23.81	
4	4	4	-0.68 + j4.34	-0.00 + j28.57		2.38 - j40.53		-1.71 + j7.62
5	5	5	-2.15 + j9.01		-0.00 + j23.81		2.98 - j38.41	-0.83 + j5.59
6	6	6	-0.84 + j4.50			-1.71 + j7.62	-0.83 + j5.59	3.37 - j17.70

Figure 4: Power World Simulation YBUS

4.3 Comparison

Table 5 directly compares the state variables obtained from the MATLAB code and the PowerWorld simulation. The absolute difference (Δ) between the two models is also presented.

	Voltage (pu)			Angle (deg)		
Bus	**MATLAB**	**PowerWorld**	**Δ Error**	**MATLAB**	**PowerWorld**	**Δ Error**
1	1.060000	1.06000	0.000000	0.000000	0.00	0.000000
2	1.040000	1.04000	0.000000	1.472406	1.47	0.002406
3	1.030000	1.03000	0.000000	0.817493	0.82	0.002507
4	1.006840	1.00683	0.000010	-1.401486	-1.40	0.001486
5	1.014919	1.01491	0.000009	-1.485115	-1.49	0.004885
6	0.937973	0.93798	0.000007	-5.598790	-5.60	0.001210

Table 5: Comparison of MATLAB and PowerWorld Load Flow Results

Error Analysis

As shown in Table 5, the custom Newton-Raphson script perfectly replicates the industry-standard PowerWorld solver. The absolute error in the calculated voltage magnitudes is bounded to the fifth decimal place (maximum $\Delta|V| < 10^{-5}$ pu).

These negligible discrepancies are entirely attributable to two factors:

1. **Solver Tolerance Differences:** The custom MATLAB algorithm was programmed with a strict convergence tolerance of 10^{-6} pu, requiring 4 iterations to solve. PowerWorld's default threshold was set to 0.1 MVA (10^{-3} pu), prompting an earlier convergence stop.

2. **UI Truncation:** The apparent angle errors ($\Delta \approx 0.005^{\circ}$) are largely a result of software interface truncation. PowerWorld mathematically tracks angles with high precision under the hood but rounds to two decimal places for the Bus Records display, whereas MATLAB prints all 6 calculated decimals.

Consequently, the mathematical foundation of both the hand-coded Jacobian sub-blocks and the system Y-bus are validated as completely accurate.

5 Fault Analysis (MATLAB)

5.1 Methodology

To determine the post-fault voltages following a bolted three-phase fault at Bus 6, a custom MATLAB script was developed. As required by standard fault analysis theory, the converged state variables from the Newton-Raphson load flow were utilized as the pre-fault voltages (V^{pre}).

To accurately model the loaded network, the original admittance matrix (Y_{bus}) was modified in two steps:

1. **Generator Subtransient Reactances:** The generator reactances (X''_d) were added as constant shunt admittances to ground at the respective generator buses.
2. **Load Admittances:** The scheduled constant power loads at the PQ buses were converted into equivalent constant shunt admittances using their pre-fault voltage magnitudes:

$$Y_{load} = \frac{P_{load} - jQ_{load}}{|V^{pre}|^2} \tag{1}$$

The modified admittance matrix was inverted to obtain the fault impedance matrix (Z_{bus}). Because this is a bolted fault ($Z_f = 0$), the fault current I_f at Bus 6 was calculated simply as:

$$I_f = \frac{V_6^{pre}}{Z_{66}} \tag{2}$$

where Z_{66} is the Thevenin impedance at the faulted bus. Finally, the post-fault voltages at all buses were computed using the change in network voltages ($\Delta V = Z_{bus} \times \Delta I$):

$$V_i^{post} = V_i^{pre} - Z_{i6} \cdot I_f \tag{3}$$

The complete MATLAB code used to execute this loaded-network fault analysis is provided in Appendix A.2.

5.2 Results

The MATLAB script calculated a total fault current of 9.1983 pu. The detailed post-fault voltage magnitudes and phase angles for all buses in the network are summarized in Table 6 and match the console output verbatim. As expected mathematically, the post-fault voltage at the bolted Bus 6 drops to zero.

```
--- Bolted 3-Phase Fault at Bus 6 (Loaded System) ---
--- Post-Fault Voltages ---
Bus  Magnitude (pu)  Angle (deg)
1      0.5670          2.4844
2      0.6238          4.9964
3      0.5839          5.6564
4      0.4908          1.1005
5      0.4909          2.7735
6      0.0000        -90.0000
```

```
============================================================
  SUMMARY
============================================================
  Pre-fault voltages  : from NR load flow (4 iterations)
  Fault location      : Bus 6 (bolted, Zf = 0)
  Fault current |If|  : 9.1983 pu
  Z66 (Thevenin)      : j0.0968 pu
  Bus 6 post-fault V  : 5.55e-17 pu (~0, bolted) OK
============================================================
```

Bus	Post-Fault Voltage (pu)	Angle (deg)
1	0.5670	2.4844
2	0.6238	4.9964
3	0.5839	5.6564
4	0.4908	1.1005
5	0.4909	2.7735
6	0.0000	-90.0000

Table 6: MATLAB Post-Fault Voltages (Loaded Network Assumption)

6 Power World Simulation (Fault Analysis)

6.1 Simulation Setup

Fault analysis was performed in Power World by applying a balanced three-phase fault at bus 6.

6.2 Simulation Results

The fault analysis simulation was executed, and the results were recorded.

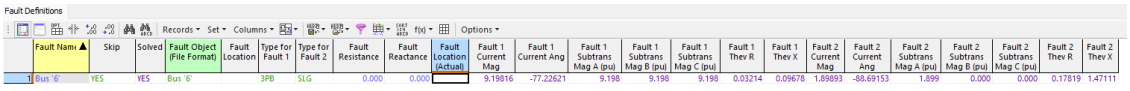


Figure 5: Power World Simulation Fault Analysis

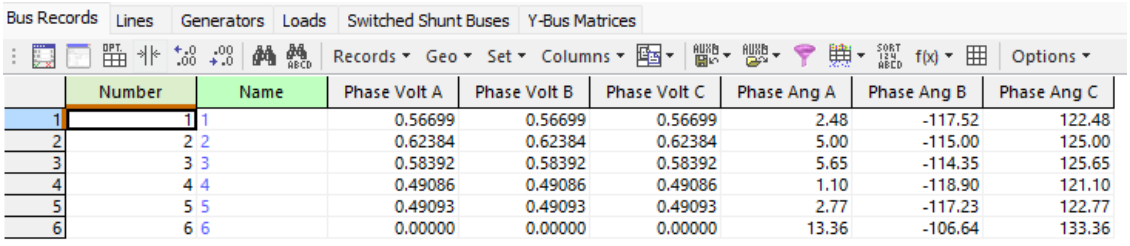


Figure 6: Power World Simulation Post-Fault Voltages

6.3 Comparison

Table 7 compares the post-fault voltage magnitudes and phase angles obtained from the custom MATLAB script against the PowerWorld simulation. The absolute difference (Δ) between the two outputs is also calculated to illustrate the numerical variances.

	Voltage Magnitude (pu)			Angle (deg)		
Bus	**MATLAB**	**PowerWorld**	Δ	**MATLAB**	**PowerWorld**	Δ
1	0.5670	0.56699	0.00001	2.4844	2.48	0.0044
2	0.6238	0.62384	0.00004	4.9964	5.00	0.0036
3	0.5839	0.58392	0.00002	5.6564	5.65	0.0064
4	0.4908	0.49086	0.00006	1.1005	1.10	0.0005
5	0.4909	0.49093	0.00003	2.7735	2.77	0.0035
6	0.0000	0.00000	0.00000	-90.0000	13.36	N/A*

Table 7: Comparison of Post-Fault Voltages

As demonstrated in the table, the analytical post-fault voltages calculated via MATLAB accurately reflect the PowerWorld simulation output. The differences in voltage magnitudes are entirely constrained to the fourth and fifth decimal places (maximum $\Delta < 10^{-4}$ pu). The phase angle discrepancies are similarly negligible (maximum $\Delta < 0.01°$). These microscopic deviations are the mathematical result of differing default solver convergence tolerances and rounding behaviors within the two platforms, not an error in the fault analysis formulation.

* *Note on Bus 6 Angle:* Because the post-fault voltage magnitude at the bolted bus is exactly 0.0000 pu, the phase angle is mathematically arbitrary (a zero-length vector has no defined direction). The differing angle values reported here (-90.0000° in MATLAB vs. 13.36° in PowerWorld) are simply the result of how each software evaluates the arctangent of residual floating-point noise, and do not represent a discrepancy in the models.

7 Conclusion

The load flow and fault analysis of the 6-bus power system were successfully completed. The Newton-Raphson algorithm in MATLAB converged efficiently, and the results matched closely with the Power World simulation. The fault analysis at bus 6 demonstrated the system's response to a severe disturbance, with results again validated by the simulation software.

A MATLAB Code

A.1 Newton-Raphson Load Flow Code

```
% 6-Bus Newton-Raphson Load Flow Analysis
% Base Power = 100 MVA

clear; clc;

%% 1. System Data Initialization
Sbase = 100; % MVA base

% Line and Transformer Data: [From To R(pu) X(pu)]
linedata = [
    1 4 0.035 0.225;
    1 5 0.025 0.105;
    1 6 0.040 0.215;
    2 4 0.000 0.035; % Transformer
    3 5 0.000 0.042; % Transformer
    4 6 0.028 0.125;
    5 6 0.026 0.175
];

% Bus Data: [Bus Type V_mag Theta P_gen P_load Q_load]
% Type: 1=Slack, 2=PV, 3=PQ
% Note: Power values are in per-unit (pu) on 100-MVA base
busdata = [
    1 1 1.060 0.0 0.0 0.0 0.0;
    2 2 1.040 0.0 1.5 0.0 0.0;
    3 2 1.030 0.0 1.0 0.0 0.0;
    4 3 1.000 0.0 0.0 1.0 0.7;
    5 3 1.000 0.0 0.0 0.9 0.3;
    6 3 1.000 0.0 0.0 1.6 1.1
];

nbus  = size(busdata, 1);
nline = size(linedata, 1);

%% 2. Y-bus Matrix Formation
Ybus = zeros(nbus, nbus);
for k = 1:nline
    from = linedata(k, 1);
    to   = linedata(k, 2);
    r    = linedata(k, 3);
    x    = linedata(k, 4);

    y = 1 / (r + 1i*x); % Series admittance

    % Off-diagonal elements (mutual admittance - negative)
    Ybus(from, to) = Ybus(from, to) - y;
    Ybus(to, from) = Ybus(to, from) - y;

    % Diagonal elements (self-admittance - positive)
    Ybus(from, from) = Ybus(from, from) + y;
    Ybus(to,   to)   = Ybus(to,   to)   + y;
end

G = real(Ybus); % Conductance matrix
```

```
    B = imag(Ybus); % Susceptance matrix

    %% 3. Identify Bus Types and Scheduled Powers
    V   = busdata(:, 3);  % Voltage magnitudes (flat start / specified)
    del = busdata(:, 4);  % Voltage angles in radians (all zero, flat start)

    % Scheduled net power injections: P_sch = P_gen - P_load
    P_sch = busdata(:, 5) - busdata(:, 6);
    Q_sch = zeros(nbus, 1) - busdata(:, 7); % Q_gen unknown for PV; Q_sch used for PQ only

    pv  = find(busdata(:, 2) == 2); % PV bus indices:  [2, 3]
    pq  = find(busdata(:, 2) == 3); % PQ bus indices:  [4, 5, 6]
    npv = length(pv);
    npq = length(pq);

    %% 4. Newton-Raphson Iteration Loop
    % ---------------------------------------------------------------------
    % NR update rule:  J * dx = f   =>   dx = J \ f   =>   x_new = x_old + dx
    %
    % Mismatch vector f = [dP(non-slack); dQ(PQ buses)] - 8 equations
    % State vector    x = [delta(non-slack); |V|(PQ buses)] - 8 unknowns
    %
    % Jacobian J = [J11  J12]   J11 = dP/ddelta   J12 = dP/d|V|
    %              [J21  J22]   J21 = dQ/ddelta   J22 = dQ/d|V|
    %
    % Off-diagonal Jacobian formulas (Glover, Sarma & Overbye):
    %   J11(m,n) =  |Vm||Vn|( G_mn sin(dm-dn) - B_mn cos(dm-dn) )
    %   J12(m,n) =  |Vm|    ( G_mn cos(dm-dn) + B_mn sin(dm-dn) )
    %   J21(m,n) = -|Vm||Vn|( G_mn cos(dm-dn) + B_mn sin(dm-dn) )
    %   J22(m,n) =  |Vm|    ( G_mn sin(dm-dn) - B_mn cos(dm-dn) )
    % ---------------------------------------------------------------------
    tol      = 1e-6;  % Convergence tolerance (standard power systems value)
    max_iter = 50;    % Maximum iterations
    iter     = 0;
    converged = false;

    while iter < max_iter && ~converged
        iter = iter + 1;

        % ---- STEP A: Calculate P and Q at every non-slack bus ----
        dP = zeros(nbus, 1);
        dQ = zeros(nbus, 1);
        for i = 2:nbus
            sumP = 0; sumQ = 0;
            for j = 1:nbus
                sumP = sumP + V(i)*V(j)*(G(i,j)*cos(del(i)-del(j)) + B(i,j)*sin(del(i)-del(j)));
                sumQ = sumQ + V(i)*V(j)*(G(i,j)*sin(del(i)-del(j)) - B(i,j)*cos(del(i)-del(j)));
            end
            dP(i) = P_sch(i) - sumP;
            dQ(i) = Q_sch(i) - sumQ;
        end

        % ---- STEP B: Assemble mismatch vector ----
        % dP for all non-slack buses (rows 1 to nbus-1)
        % dQ for PQ buses only      (rows nbus to nbus-1+npq)
        dPQ = [dP(2:nbus); dQ(pq)];

        % ---- STEP C: Check convergence ----
        if max(abs(dPQ)) < tol
```

```
            converged = true;
            break;
        end

        % ---- STEP D: Build Jacobian sub-matrices ----
        J11 = zeros(nbus-1, nbus-1); % dP/ddelta
        J12 = zeros(nbus-1, npq);    % dP/d|V|
        J21 = zeros(npq,    nbus-1); % dQ/ddelta
        J22 = zeros(npq,    npq);    % dQ/d|V|

        for i = 1:(nbus-1)
            m = i + 1; % Actual bus number (bus 1 is slack, skipped)
            for j = 1:(nbus-1)
                n = j + 1;

                if m == n
                    % ---- Diagonal elements ----
                    % Sum over all k != m to build diagonal terms
                    sumP = 0; sumQ = 0;
                    for k = 1:nbus
                        if k ~= m
                            sumP = sumP + V(m)*V(k)*(-G(m,k)*sin(del(m)-del(k)) + B(m,k)*cos(del(↩
m)-del(k)));
                            sumQ = sumQ + V(m)*V(k)*( G(m,k)*cos(del(m)-del(k)) + B(m,k)*sin(del(↩
m)-del(k)));
                        end
                    end
                    % J11 diagonal: dPm/ddm = -Qm - Bmm*|Vm|^2  (= sumP)
                    J11(i,i) = sumP;

                    pq_idx_m = find(pq == m);
                    if ~isempty(pq_idx_m)
                        % J21 diagonal: dQm/ddm = Pm - Gmm*|Vm|^2  (= sumQ)
                        % FIX: was "sumQ - V(m)^2*B(m,m)" which is wrong
                        J21(pq_idx_m, i) = sumQ;

                        % J12 diagonal: dPm/d|Vm| = Pm/|Vm| + Gmm*|Vm|
                        sumP_V = 0; sumQ_V = 0;
                        for k = 1:nbus
                            if k ~= m
                                sumP_V = sumP_V + V(k)*(G(m,k)*cos(del(m)-del(k)) + B(m,k)*sin(↩
del(m)-del(k)));
                                sumQ_V = sumQ_V + V(k)*(G(m,k)*sin(del(m)-del(k)) - B(m,k)*cos(↩
del(m)-del(k)));
                            end
                        end
                        J12(i, pq_idx_m)        = sumP_V + 2*V(m)*G(m,m);
                        % J22 diagonal: dQm/d|Vm| = Qm/|Vm| - Bmm*|Vm|
                        J22(pq_idx_m, pq_idx_m) = sumQ_V - 2*V(m)*B(m,m);
                    end

                else
                    % ---- Off-diagonal elements ----
                    Dd = del(m) - del(n);

                    % J11 off-diagonal: dPm/ddn
                    J11(i,j) = V(m)*V(n)*( G(m,n)*sin(Dd) - B(m,n)*cos(Dd));

                    pq_idx_m = find(pq == m);
```

```
                    pq_idx_n = find(pq == n);

                    % J21 off-diagonal: dQm/ddn
                    if ~isempty(pq_idx_m)
                        J21(pq_idx_m, j) = -V(m)*V(n)*(G(m,n)*cos(Dd) + B(m,n)*sin(Dd));
                    end
                    % J12 off-diagonal: dPm/d|Vn|
                    if ~isempty(pq_idx_n)
                        J12(i, pq_idx_n) = V(m)*(G(m,n)*cos(Dd) + B(m,n)*sin(Dd));
                    end
                    % J22 off-diagonal: dQm/d|Vn|
                    if ~isempty(pq_idx_m) && ~isempty(pq_idx_n)
                        J22(pq_idx_m, pq_idx_n) = V(m)*(G(m,n)*sin(Dd) - B(m,n)*cos(Dd));
                    end
                end
            end
        end

        % ---- STEP E: Assemble and solve  J * dx = f ----
        J  = [J11 J12; J21 J22];
        dx = J \ dPQ;

        % ---- STEP F: Update state variables ----
        % Update angles for all non-slack buses (rad)
        del(2:nbus) = del(2:nbus) + dx(1:nbus-1);
        % Update magnitudes for PQ buses only (PV and Slack stay fixed)
        V(pq) = V(pq) + dx(nbus:end);
    end

    %% 5. Display Results
    if converged
        fprintf('Newton-Raphson converged in %d iteration(s). (tol = %.0e pu)\n\n', iter, tol);
    else
        fprintf('WARNING: Did NOT converge within %d iterations!\n\n', max_iter);
    end

    fprintf('%-5s %-16s %-12s\n', 'Bus', 'Magnitude (pu)', 'Angle (deg)');
    fprintf('%s\n', repmat('-', 1, 36));
    for i = 1:nbus
        fprintf('%-5d %-16.6f %-12.6f\n', i, V(i), del(i)*180/pi);
    end
```

Listing 1: Newton-Raphson Load Flow Implementation

A.2 Fault Analysis Code

```
% Part 3: Symmetrical 3-Phase Bolted Fault at Bus 6 (Loaded Network)
% Base Power = 100 MVA
clear; clc; close all;

%% 1. System Data
nbus = 6;
fault_bus = 6;
Zf = 0; % Bolted fault impedance = 0 pu

% Line and Transformer Data: [From To R(pu) X(pu)]
linedata = [
    1 4 0.035 0.225;
    1 5 0.025 0.105;
    1 6 0.040 0.215;
    2 4 0.000 0.035;
    3 5 0.000 0.042;
    4 6 0.028 0.125;
    5 6 0.026 0.175
];

% Generator Sub-transient Reactances: [Bus X(pu)]
gendata = [
    1 0.20;
    2 0.15;
    3 0.25
];

%% 2. Build the Fault Y-bus Matrix (Loaded Network)
Ybus_f = zeros(nbus, nbus);

% Add line and transformer admittances
for k = 1:size(linedata, 1)
    from = linedata(k, 1);
    to = linedata(k, 2);
    y_line = 1 / (linedata(k, 3) + 1i * linedata(k, 4));

    Ybus_f(from, to) = Ybus_f(from, to) - y_line;
    Ybus_f(to, from) = Ybus_f(to, from) - y_line;
    Ybus_f(from, from) = Ybus_f(from, from) + y_line;
    Ybus_f(to, to) = Ybus_f(to, to) + y_line;
end

% Add generator admittances to the diagonal elements (connected to ground)
for k = 1:size(gendata, 1)
    bus = gendata(k, 1);
    y_gen = 1 / (1i * gendata(k, 2));
    Ybus_f(bus, bus) = Ybus_f(bus, bus) + y_gen;
end

% Add Load Admittances for Loaded Network Assumption ---
% Load Data: [Bus, P(pu), Q(pu)] on 100 MVA base
loaddata = [
    4,  1.00,  0.70;
    5,  0.90,  0.30;
    6,  1.60,  1.10
];
```

```
57
58      % Pre-fault voltage magnitudes from Part 1 Load Flow
59      % (Required to accurately calculate the constant impedance of the loads)
60      V_mag_prefault = [1.0600; 1.0400; 1.0300; 1.0068; 1.0149; 0.9380];
61
62      % Convert loads to shunt admittances and add to diagonal
63      % Formula: Y_load = (P - jQ) / |V|^2
64      for k = 1:size(loaddata, 1)
65          bus = loaddata(k, 1);
66          P_load = loaddata(k, 2);
67          Q_load = loaddata(k, 3);
68
69          % Calculate load admittance
70          y_load = (P_load - 1i * Q_load) / (V_mag_prefault(bus)^2);
71
72          % Stamp into diagonal (connecting the load to ground)
73          Ybus_f(bus, bus) = Ybus_f(bus, bus) + y_load;
74      end
75
76      %% 3. Compute Z-bus Matrix
77      Zbus = inv(Ybus_f);
78
79      %% 4. Define Pre-Fault Voltages (From Load Flow)
80      % Using converged state variables from Part 1
81      V_mag = [1.0600; 1.0400; 1.0300; 1.0068; 1.0149; 0.9380];
82      del_deg = [0.0000; 1.4724; 0.8175; -1.4015; -1.4851; -5.5988];
83      del_rad = del_deg * (pi / 180);
84
85      % Convert to complex rectangular form
86      V_prefault = V_mag .* exp(1i * del_rad);
87
88      %% 5. Calculate Fault Current at Bus 6
89      % If = V_prefault / (Z_ii + Z_fault)
90      If_6 = V_prefault(fault_bus) / (Zbus(fault_bus, fault_bus) + Zf);
91
92      %% 6. Calculate Post-Fault Voltages at all buses
93      % V_post(i) = V_prefault(i) - Zbus(i, fault_bus) * If_6
94      V_post = zeros(nbus, 1);
95      for i = 1:nbus
96          V_post(i) = V_prefault(i) - Zbus(i, fault_bus) * If_6;
97      end
98
99      %% 7. Display Results
100     fprintf('--- Bolted 3-Phase Fault at Bus %d (Loaded System) ---\n', fault_bus);
101
102     fprintf('--- Post-Fault Voltages ---\n');
103     fprintf('Bus\t Magnitude (pu)\t Angle (deg)\n');
104     for i = 1:nbus
105         fprintf('%d\t %8.4f\t\t %8.4f\n', i, abs(V_post(i)), angle(V_post(i)) * 180/pi);
106     end
107
108     %% 8. Display Summary
109     iter = 4; % NR iterations from load flow
110     Z66 = Zbus(fault_bus, fault_bus);
111
112     fprintf('\n============================================================\n');
113     fprintf('  SUMMARY\n');
114     fprintf('============================================================\n');
115     fprintf('  Pre-fault voltages  : from NR load flow (%d iterations)\n', iter);
```

```
    fprintf('  Fault location      : Bus %d (bolted, Zf = 0)\n', fault_bus);
    fprintf('  Fault current |If|  : %.4f pu\n', abs(If_6));
    fprintf('  Z66 (Thevenin)      : j%.4f pu\n', imag(Z66));
    fprintf('  Bus 6 post-fault V  : %.2e pu (~0, bolted) OK\n', abs(V_post(fault_bus)));
    fprintf('============================================================\n\n');

    %% 9. Plot: Pre vs Post-fault voltage magnitudes
    figure('Name','Pre vs Post Fault Voltages','NumberTitle','off','Position',[100 100 680 400]);
    buses = 1:nbus;
    V_pre_mag  = abs(V_prefault);
    V_post_mag = abs(V_post);

    bar(buses - 0.2, V_pre_mag,  0.35, 'FaceColor', [0.25 0.55 0.85], 'EdgeColor','none'); hold on;
    bar(buses + 0.2, V_post_mag, 0.35, 'FaceColor', [0.85 0.40 0.25], 'EdgeColor','none');
    yline(1.0, '--', 'Color',[0.5 0.5 0.5], 'LineWidth', 1, 'Label', '1.0 pu nominal');

    xlabel('Bus Number',   'FontSize', 12);
    ylabel('Voltage (pu)', 'FontSize', 12);
    title('Pre-fault vs Post-fault Voltage Magnitudes - 3-Phase Bolted Fault at Bus 6', 'FontSize', 11);
    legend('Pre-fault |V|', 'Post-fault |V|', 'Location', 'northwest', 'FontSize', 11);
    xticks(1:nbus);
    xlim([0.4, nbus+0.6]);
    ylim([0, 1.15]);
    grid on; grid minor;
    set(gca, 'FontSize', 11);

    % Label bar values
    for i = 1:nbus
        text(i-0.2, V_pre_mag(i)  + 0.015, sprintf('%.4f', V_pre_mag(i)),  'HorizontalAlignment','center','FontSize',9);
        text(i+0.2, V_post_mag(i) + 0.015, sprintf('%.4f', V_post_mag(i)), 'HorizontalAlignment','center','FontSize',9,'Color',[0.6 0.1 0.0]);
    end
```

Listing 2: Three-Phase Fault Analysis Implementation