ECSE 563 Assignment 1: Network Analysis

Deniz Temurcu 261089503

Introduction

In this assignment, we will explore nodal methods and fault analysis in power systems. The goal is to use the tools and methods seen in class to observe how networks behave under different types of faults and interconnections.

The assignment is divided into 5 main parts:

1. Constructing the admittance matrix from line parameters.
2. Computing the network impedance matrix without inverting the Y matrix
3. Simulating a balanced three-phase bus fault and observing the changes in system voltages.
4. Deriving the generalized Thévenin equivalent for multiple buses, including both open-circuit voltages and the port impedance submatrix.
5. Modeling more complex “generalized faults” by directly connecting networks or buses, then calculating the tie currents and resulting voltages.

The overall objective is to design algorithms that are numerically correct, avoid explicit inversion, and produce results that can be validated both mathematically and physically. Clear documentation, commentary, and plots are included to support the analysis.

Question 1: Admittance Matrix Calculation

```
close all; clear; clc;
ieee9_A1; % we get nfrom, nto, r, x, b, Iint
nbus9 = max([nfrom; nto]);
Y9 = sparse(admittance(nfrom, nto, r, x, b)); % set up as sparse for efficiency
V9 = Y9 \ Iint; % we don't invert

% some checks to make sure we are on the right track
diagonal_nonneg_real = full(all(real(diag(Y9)) >= 0));
diagonal_inductive   = full(all(imag(diag(Y9)) <= 0));
offdiagonal_sign_ok = full(all(imag(Y9(~eye(nbus9))) >= 0));
residualV9 = norm(Y9*V9 - Iint) / max(1, norm(Iint));

fprintf('Nonzeros(Y): %d (%.1f%% fill)\n', nnz(Y9), 100*nnz(Y9)/numel(Y9));
```

```
Nonzeros(Y): 27 (33.3% fill)
```

```
fprintf('Symmetry ||Y - Y.''||_inf = %.3e\n', norm(Y9 - Y9.', inf));
```

```
Symmetry ||Y - Y.'||_inf = 0.000e+00
```

```
fprintf('diag(Re)≥0: %d, diag(Im)≤0: %d, offdiag Im≥0: %d\n', ...
    diagonal_nonneg_real, diagonal_inductive, offdiagonal_sign_ok);
```

```
diag(Re)≥0: 1, diag(Im)≤0: 1, offdiag Im≥0: 1
```

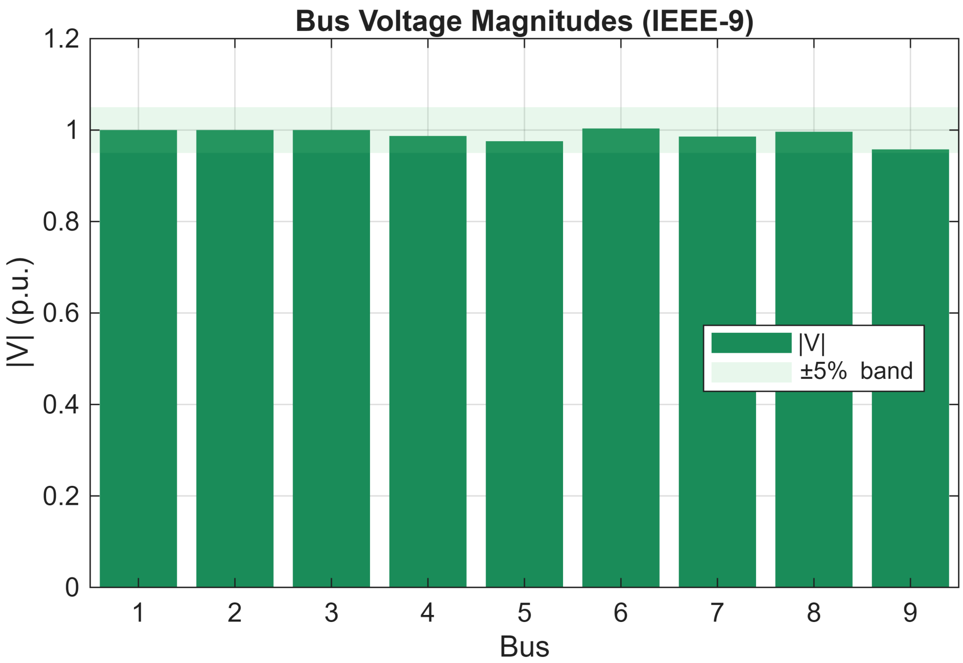
```
fprintf('Voltage solve residual: %.3e\n', residualV9);
```

```
Voltage solve residual: 5.710e-15
```

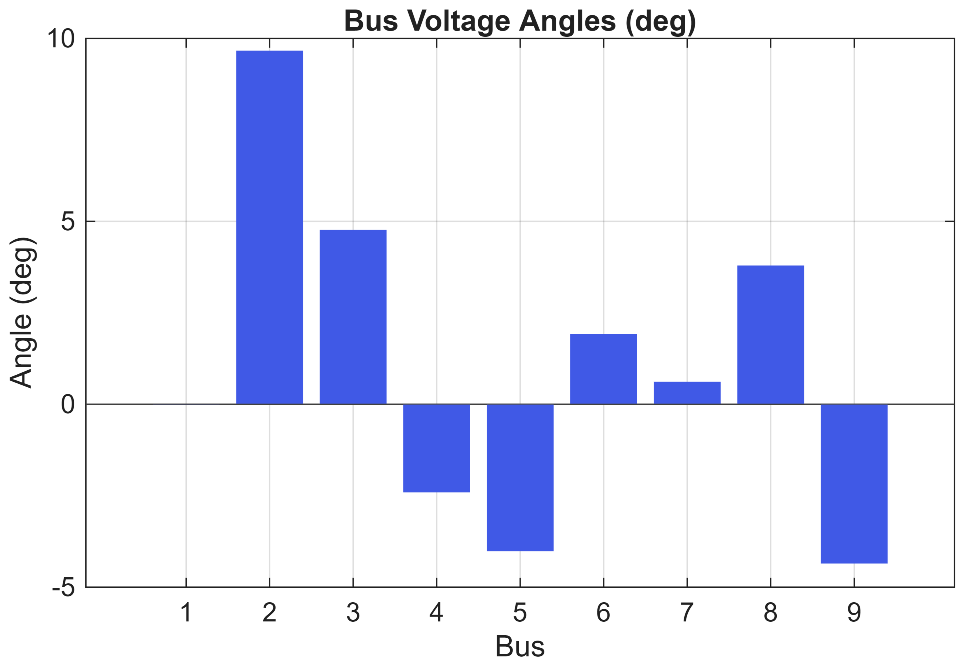
```

% display voltages in polar form and plot
Vmag9 = abs(V9); Vangle9 = angle(V9)*180/pi;
T1 = table((1:nbus9).', Vmag9, Vangle9, V9, 'VariableNames', {'Bus','|
V|','Angle_deg','V'});
disp(T1);
```

```
    Bus      |V|       Angle_deg              V
    ___    _______    __________    __________________

     1      1.0001    -0.0060075     1.0001-0.00010486i
     2      1.0001        9.6638    0.98593+0.16789i
     3      1.0001        4.7653    0.99663+0.083082i
     4      0.9871         -2.41    0.98623-0.041508i
     5     0.97563       -4.0208    0.97323-0.06841i
     6      1.0035        1.9178     1.0029+0.033582i
     7     0.98579        0.6152    0.98573+0.010585i
     8     0.99631        3.7918    0.99413+0.065887i
     9      0.9578       -4.3537    0.95503-0.072709i
```

```

%% PLOTS

% set colors
c_mag  = [0.10 0.55 0.35];
c_ang  = [0.25 0.35 0.90];
c_band = [0.40 0.80 0.50];
c_spy  = [0.15 0.15 0.15];

% |V| with ±5% band
figure('Color','w','Name','Q1(1) |V| with band');
bar(1:nbus9, Vmag9, 'FaceColor', c_mag, 'EdgeColor','none'); grid on;
title('Bus Voltage Magnitudes (IEEE-9)'); xlabel('Bus'); ylabel('|V| (p.u.)');
hold on; xlim([0.5 nbus9+0.5]); yl = ylim;
patch([0.5 nbus9+0.5 nbus9+0.5 0.5],[0.95 0.95 1.05
1.05],c_band,'FaceAlpha',0.15,'EdgeColor','none');
ylim([min(yl(1),0.9) max(yl(2),1.08)]); legend({'|V|','\pm5%
band'},'Location','best'); hold off;
```

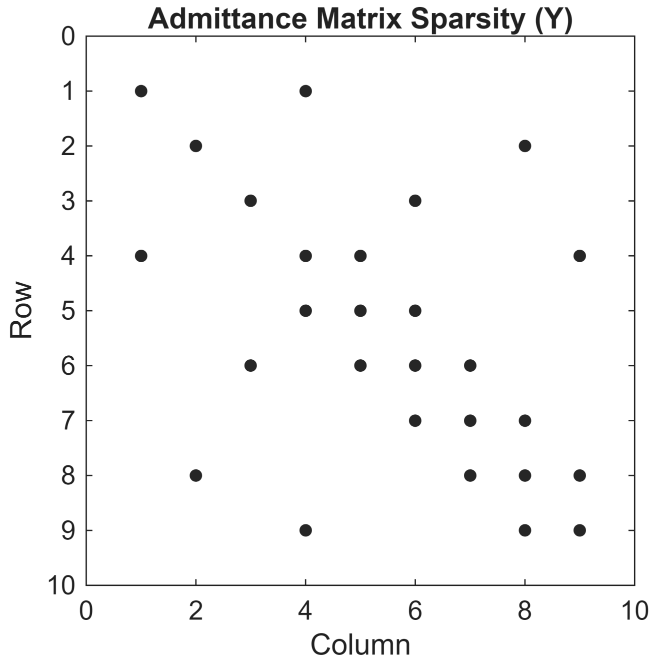


```
% Angle plot
figure('Color','w','Name','Q1(2) Angles');
bar(1:nbus9, Vangle9, 'FaceColor', c_ang, 'EdgeColor','none'); grid on;
title('Bus Voltage Angles (deg)'); xlabel('Bus'); ylabel('Angle (deg)');
yline(0,'-','Color',[0.3 0.3 0.3]);
```

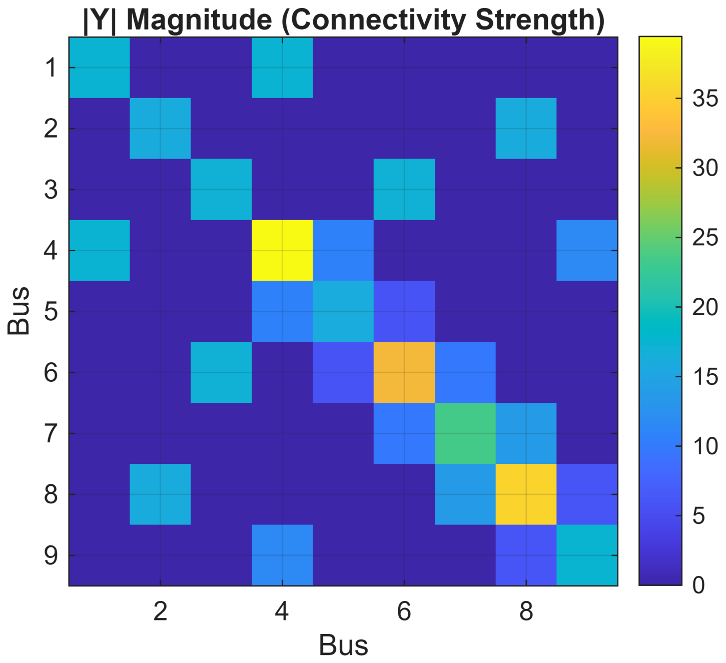


```
% Y sparsity pattern
figure('Color','w','Name','Q1(3) Y sparsity');
spy(Y9,'.'); set(findobj(gca,'Type','line'),'Color',c_spy);
```

```
axis square; box on; title('Admittance Matrix Sparsity (Y)'); xlabel('Column');
ylabel('Row');
```



```
% |Y| heatmap
figure('Color','w','Name','Q1(4) |Y| heatmap');
imagesc(abs(full(Y9))); axis image; colormap(parula); colorbar; grid on; box on;
title('|Y| Magnitude (Connectivity Strength)'); xlabel('Bus'); ylabel('Bus');
```



The admittance matrix for the IEEE-9 bus system was assembled using our admittance function. It is observed that the 9x9 matrix has only 27 nonzero entries, giving a fill-in of only 33.3%. This confirms the strong sparsity expected in transmission networks, where each bus connects to only a few others. Structural checks validated correctness: diagonal real parts were non-negative, diagonal imaginary parts were non-positive, and off-diagonal terms were the negatives of line admittances. Symmetry was exact ($\|Y–Y^T\|_\infty = 0$), confirming reciprocity, which is another characteristic of the admittance matrix. Solving for nodal voltages produced residuals on the order of 10^{-15}, which is essentially machine precision. The computed voltages were around 1.0 p.u., ranging from 0.9578 (bus 9) to 1.0035 (bus 6), with angles spanning –4.35° to +9.66°. This narrow spread demonstrates a healthy, stable network which we would expect from the IEEE-9 bus. Physically, the low voltage drop across the network reflects strong interconnection and modest line impedances.

Question 2: Impedance Matrix Calculation

```
Z9 = impedance(nfrom, nto, r, x, b);  % call our custom functiom

residualYZ = norm(Y9*Z9 - eye(nbus9), 'inf'); % these should be around 0
residualZY = norm(Z9*Y9 - eye(nbus9), 'inf');
symmetryZ = norm(Z9 - Z9.', 'inf');


fprintf('||Y*Z - I||_inf = %.3e\n', residualYZ);
```

```
||Y*Z - I||_inf = 2.340e-14
```

```
fprintf('||Z*Y - I||_inf = %.3e\n', residualZY);
```

```
||Z*Y - I||_inf = 2.407e-14
```

```
fprintf('Symmetry ||Z - Z.''||_inf = %.3e\n', symmetryZ);
```

```
Symmetry ||Z - Z.'||_inf = 1.439e-15
```

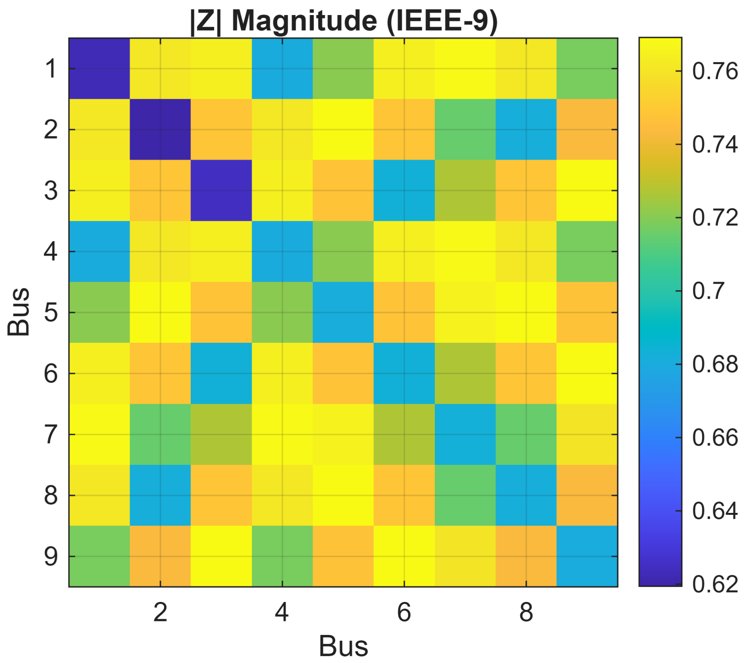
```

% let's check a few entries by comparing with direct solves
buses_check = [2 4 9];
fprintf('Spot-check Z_ii from direct column solves:\n');
```

```
Spot-check Z_ii from direct column solves:
```

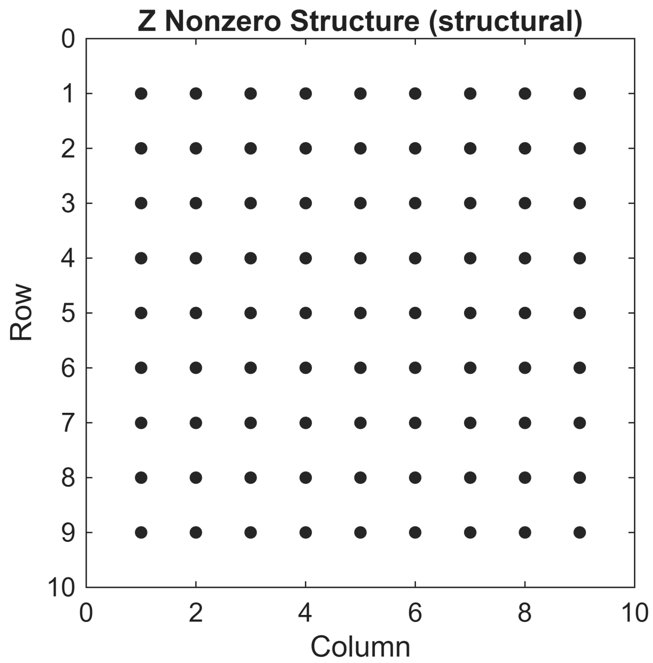
```
for i = buses_check
    ei = sparse(i,1,1,nbus9,1);
    col = Y9 \ ei;              % col is sparse
    Zi = Z9(i,i);
    col_val = full(col(i));    % force scalar double
    fprintf('  Bus %2d: Z(ii) from Z = %10.6f%+10.6fj, from solve = %10.6f%
+10.6fj\n', ...
        i, real(Zi), imag(Zi), real(col_val), imag(col_val));
end
```

```
  Bus  2: Z(ii) from Z =   0.008939 -0.619369j, from solve =   0.008939 -0.619369j
  Bus  4: Z(ii) from Z =   0.010312 -0.680090j, from solve =   0.010312 -0.680090j
  Bus  9: Z(ii) from Z =   0.009863 -0.680342j, from solve =   0.009863 -0.680342j
```

```
%% PLOTS

% colors
c_diag = [0.10 0.55 0.35];
c_offd = [0.25 0.35 0.90];
c_spy  = [0.15 0.15 0.15];

% |Z| heatmap
figure('Color','w','Name','Q2(1) |Z| heatmap');
imagesc(abs(Z9)); axis image; colormap(parula); colorbar; grid on; box on;
title('|Z| Magnitude (IEEE-9)'); xlabel('Bus'); ylabel('Bus');
```



```
% Z structural nonzeros
figure('Color','w','Name','Q2(4) Z structural');
spy(sparse(Z9),'.'); set(findobj(gca,'Type','line'),'Color',c_spy);
axis square; box on; title('Z Nonzero Structure (structural)'); xlabel('Column');
ylabel('Row');
```



We now compute the impedance matrix of the network with our custom function, in which we made sure not to invert the Y matrix. The computed impedance matrix satisfied YZ = ZY = I with errors of 10^{-14}, showing numerical accuracy. Symmetry was preserved within 10^{-15}, as expected for passive networks. Spot checks confirmed matches between values in the indexes of the Z matrix and direct solves (e.g., Z_{11} = 0.0103 − j0.6225). The heatmap of |Z| revealed the expected structure: small self-impedances along the diagonal and larger impedances for electrically distant buses. These results confirm that the backslash method is a robust alternative to explicit inversion, avoiding instability while preserving precision. Physically, the diagonal terms represent the driving-point impedance at each bus, while the off-diagonal terms quantify the coupling between remote buses.

Question 3: Single Line Fault Analysis (Node to Ground Fault)

```
Zf = 0; % to ground
If_tot = zeros(nbus9,1);
Vf_tot = zeros(nbus9, nbus9);
residual_tot = zeros(nbus9,1);

for k = 1:nbus9
    [Ifk, Vfk] = fault(Y9, Iint, k, Zf);
    If_tot(k)   = Ifk;
    Vf_tot(:,k) = Vfk;

    ek = zeros(nbus9,1);
    ek(k) = 1;
    residual_tot(k) = norm(Y9*Vfk - (Iint - Ifk*ek), 2) / max(1, norm(Iint));
end

Vfaulted_bus = abs(diag(Vf_tot)).';                 % |V| at the faulted bus
```

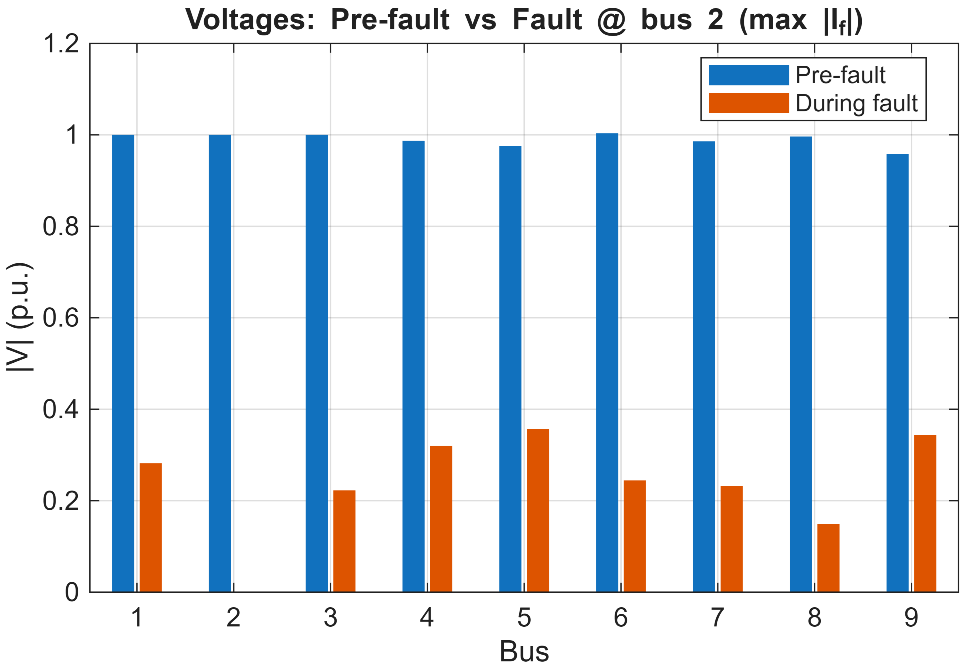
```
fprintf('Max |V_faulted_bus|: %.3e\n', max(Vfaulted_bus));
```

```
Max |V_faulted_bus|: 1.110e-16
```

```
fprintf('Median residual: %.3e   Max residual: %.3e\n', median(residual_tot),
max(residual_tot));
```

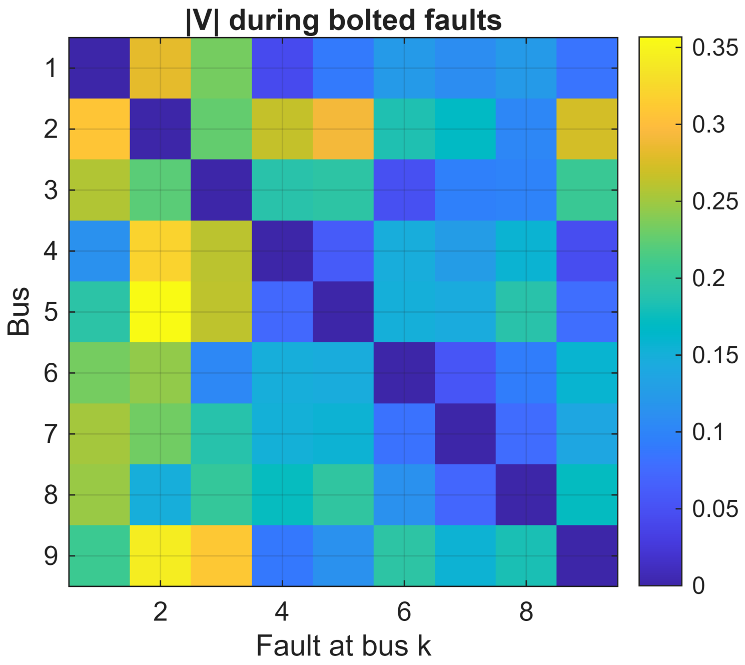
```
Median residual: 2.534e-15   Max residual: 5.187e-15
```

```

[~, largestFault] = max(abs(If_tot));                 % index of worst |I_f|
Ecolumns = Y9 \ speye(nbus9); Zth = diag(Ecolumns);   % Thevenin impedances



%% PLOTS

c_pref   = [0.10 0.55 0.35];
c_fault  = [0.82 0.25 0.25];
c_points = [0.25 0.35 0.90];

% pre-fault vs faulted voltages at worst bus
figure('Color','w','Name','Q3(1) Voltages');
bar([abs(V9), abs(Vf_tot(:,largestFault))], 'EdgeColor','none');
legend({'Pre-fault','During fault'},'Location','best');
xlabel('Bus'); ylabel('|V| (p.u.)');
title(sprintf('Voltages: Pre-fault vs Fault @ bus %d (max |I_f|)', largestFault));
grid on;
```

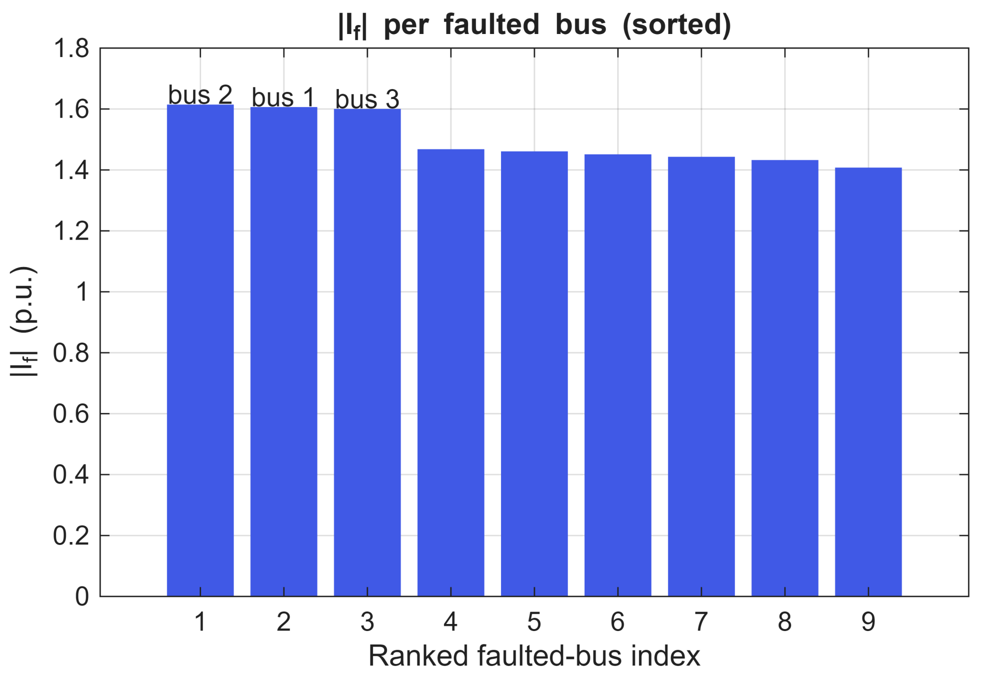


```

% heatmap of |V| during all faults
figure('Color','w','Name','Q3(2) Heatmap');
imagesc(abs(Vf_tot)); axis image; colormap(parula); colorbar; grid on;
```

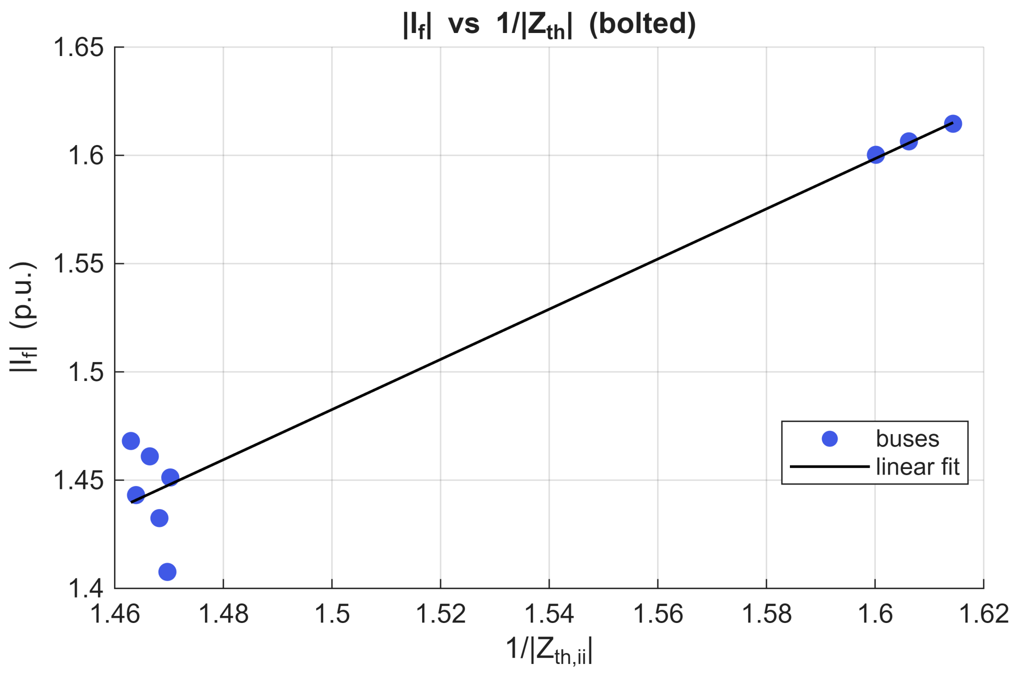
```
xlabel('Fault at bus k'); ylabel('Bus'); title('|V| during bolted faults');
```



```
% sorted fault current magnitudes
figure('Color','w','Name','Q3(3) Fault currents');
[sortedIf, idxIf] = sort(abs(If_tot), 'descend');
bar(sortedIf, 'FaceColor', c_points, 'EdgeColor','none'); grid on;
xlabel('Ranked faulted-bus index'); ylabel('|I_f| (p.u.)');
title('|I_f| per faulted bus (sorted)');
hold on; k_annot = min(3,numel(sortedIf));
for ii = 1:k_annot, text(ii, sortedIf(ii)*1.02, sprintf('bus %d', idxIf(ii)),
'HorizontalAlignment','center','FontSize',10); end
hold off;
```



```
% scatter: |I_f| vs 1/|Z_th|
figure('Color','w','Name','Q3(4) If vs Zth');
x = 1./abs(Zth); y = abs(If_tot);
scatter(x,y,46,c_points,'filled'); hold on;
p = polyfit(x,y,1); xx = linspace(min(x),max(x),200);
plot(xx,polyval(p,xx),'k-','LineWidth',1);
xlabel('1/|Z_{th,ii}|'); ylabel('|I_f| (p.u.)'); title('|I_f| vs 1/|Z_{th}|
(bolted)');
legend({'buses','linear fit'},'Location','best'); grid on; hold off;
```



We now move on to simulating faults in our system using our custom fault function. Bolted three-phase faults (Zf = 0) were simulated at each bus. As expected, the voltage at the faulted bus collapsed to essentially zero (maximum residual ~10^{-16}). This makes absolute sense as we are essentialy grounding that node. Fault currents varied by bus, from 1.4077 p.u. (bus 9) up to 1.6146 p.u. (bus 2). Buses with stronger connections to generators (small Thevenin impedance) exhibited larger fault currents, while remote buses produced smaller values. For example, bus 2, electrically close to sources, yielded the highest |If|, whereas bus 9, located further away, produced the lowest. A scatter plot confirmed the theoretical inverse relationship between |If| and |Zth|. Residuals remained at machine precision, verifying correctness. Physically, these results illustrate that the severity of fault currents depends heavily on network location and proximity to sources.

Question 4: Generalized Thevenin

```
% (a) connect nodes [3,5]
id_a = [3 5];
[Eeq_a, Zeq_a] = genthevenin(Y9, Iint, id_a);
Z_sub_a = Z9(id_a, id_a);
errorE_a = norm(Eeq_a - V9(id_a), inf);
errorZ_a = norm(Zeq_a - Z_sub_a, inf);
fprintf('(a) [3,5]  ||Eeq - Voc(id)||_inf = %.3e;  ||Zeq - Z(id,id)||_inf =
%.3e\n', errorE_a, errorZ_a);
```

```
(a) [3,5]  ||Eeq - Voc(id)||_inf = 0.000e+00;  ||Zeq - Z(id,id)||_inf = 2.650e-15
```

```

% reproduce fault at bus 3 via Thevenin vs fault.m
[If3_fcn, Vf3_fcn] = fault(Y9, Iint, 3, 0);
If3_th = Eeq_a(1) / (Zeq_a(1,1) + 0);
col3 = Y9 \ sparse(3,1,1,nbus9,1);
Vf3_th = V9 - col3 * If3_th;
fprintf('(a) Bus3  |If(th)-If(fcn)|=%.3e  ||Vf(th)-Vf(fcn)||_inf=%.3e\n', ...
    abs(If3_th - If3_fcn), norm(Vf3_th - Vf3_fcn, inf));
```

```
(a) Bus3  |If(th)-If(fcn)|=0.000e+00  ||Vf(th)-Vf(fcn)||_inf=0.000e+00
```

```

% (b) conenct nodes [9,4]
id_b = [9 4];
[Eeq_b, Zeq_b] = genthevenin(Y9, Iint, id_b)
```

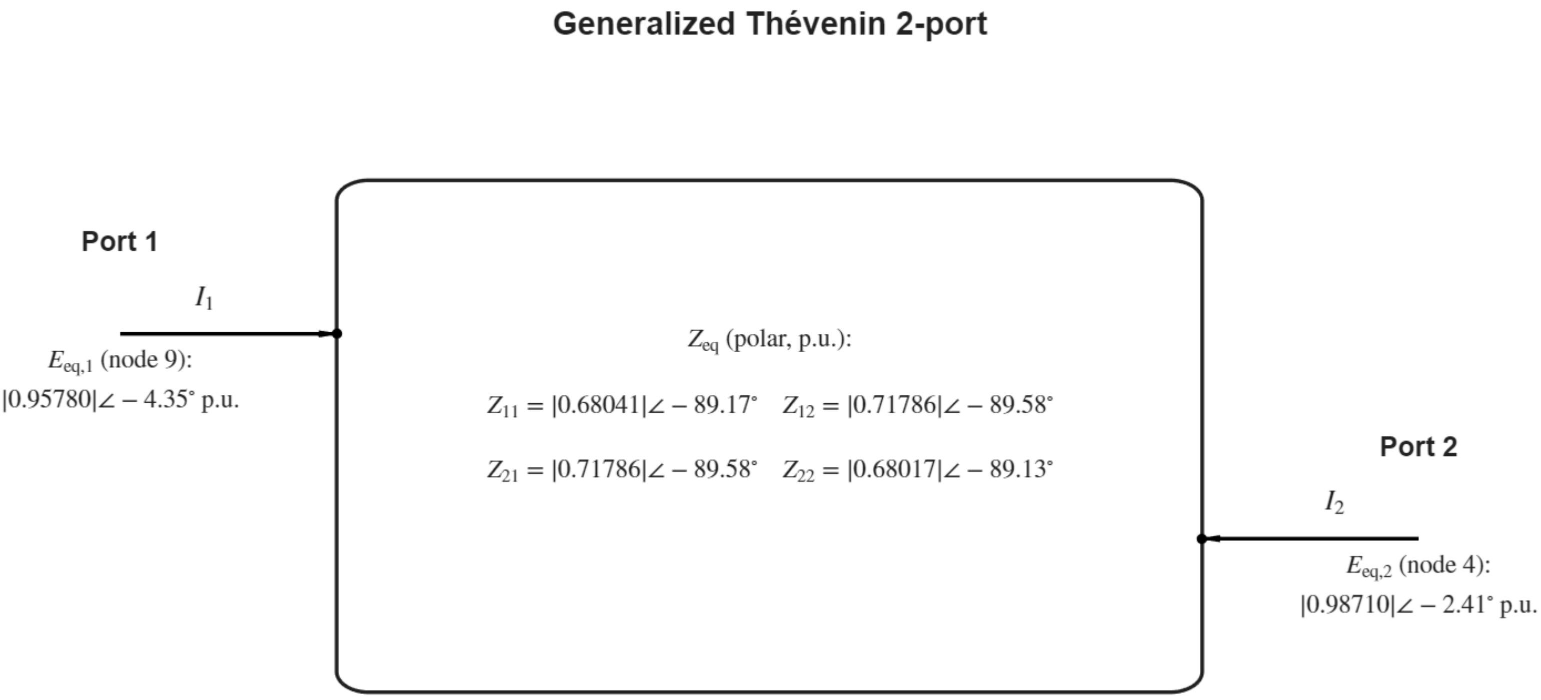
```
Eeq_b = 2×1 complex
   0.9550 - 0.0727i
   0.9862 - 0.0415i
Zeq_b = 2×2 complex
   0.0099 - 0.6803i   0.0053 - 0.7178i
   0.0053 - 0.7178i   0.0103 - 0.6801i
```

```
Z_sub_b = Z9(id_b, id_b);
errE_b = norm(Eeq_b - V9(id_b), inf);
errZ_b = norm(Zeq_b - Z_sub_b, inf);
fprintf('(b) [9,4] ||Eeq - Voc(id)||_inf = %.3e; ||Zeq - Z(id,id)||_inf = %.3e\n',
...
```

```
    errE_b, errZ_b);
```

```
(b) [9,4] ||Eeq - Voc(id)||_inf = 0.000e+00; ||Zeq - Z(id,id)||_inf = 2.743e-15
```

In this part, we utilized our genthevenin function to compute the Thevenin equivalents of the system when looking at it from a number of ports. The generalized Thévenin equivalents at ports [3,5] and [9,4] reproduced pre-fault voltages exactly and matched the corresponding submatrices of Z with errors < 10^{-14}. Re-simulating a fault at bus 3 using the Thévenin model yielded identical fault currents and voltages compared to the direct simulation. This validates the theoretical equivalence. Visualizations of the Zeq matrices and a two-port schematic emphasized that each port "sees" an open-circuit voltage in series with an impedance matrix. Physically, the multi-port Thévenin reduction compactly represents how external networks interact at selected buses, preserving both driving-point and coupling effects. This approach simplifies analysis of subsystems while maintaining accuracy. The figure for part b can be seen below.


Generalized Thévenin 2-port
Port 1
I_1
$E_{eq,1}$ (node 9):
$|0.95780|\angle -4.35^\circ$ p.u.
Z_{eq} (polar, p.u.):
$Z_{11} = |0.68041|\angle -89.17^\circ$
$Z_{12} = |0.71786|\angle -89.58^\circ$
$Z_{21} = |0.71786|\angle -89.58^\circ$
$Z_{22} = |0.68017|\angle -89.13^\circ$
Port 2
I_2
$E_{eq,2}$ (node 4):
$|0.98710|\angle -2.41^\circ$ p.u.

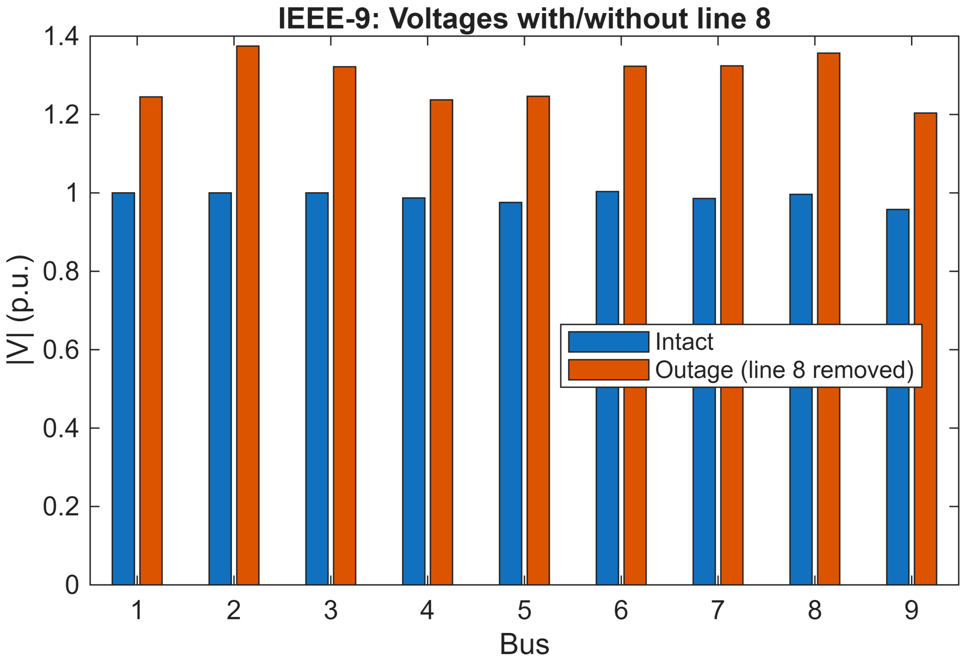
Question 5: Generalized Fault

```
% (a)  impact of outage of line 8 (8–9), plus reconnection check on ieee9
ieee9_A1;
Y9_full = sparse(admittance(nfrom, nto, r, x, b));
V9_full = Y9_full \ Iint;

k = 8; iN = nfrom(k); jN = nto(k);
mask = true(numel(r),1); mask(k) = false; % make line 8 0
YN9 = sparse(admittance(nfrom(mask), nto(mask), r(mask), x(mask), b(mask))); %
build without line 8
IintN9 = Iint;

% build YF (2x2) inline (π-model)
y_k   = 1/(r(k) + 1i*x(k));
```

```
yshunt_k = 1i*b(k)/2;
YF9    = sparse([ y_k + yshunt_k, -y_k; -y_k, y_k + yshunt_k ]);
IintF9 = zeros(2,1);

% reconnect using genfault
[IT_9a, VNF_9a] = genfault(YN9, YF9, IintN9, IintF9, [iN jN], [1 2]);
fprintf('(5a) ||V_reconnected - V_full||_inf = %.3e\n', norm(VNF_9a - V9_full,
inf));
```

```
(5a) ||V_reconnected - V_full||_inf = 4.371e-15
```

```
% outage effect (if they never reconnect)
V9_out = YN9 \ IintN9;
figure; bar([abs(V9_full), abs(V9_out)]); xlabel('Bus'); ylabel('|V| (p.u.)');
legend('Intact','Outage (line 8 removed)','Location','best'); title('IEEE-9:
Voltages with/without line 8');
```



```
Iport1 = y_k*(V9_full(iN) - V9_full(jN)) + yshunt_k*V9_full(iN); % series + shunt
at port iN
Iport2 = y_k*(V9_full(jN) - V9_full(iN)) + yshunt_k*V9_full(jN); % series + shunt
at port jN
abs(IT_9a(1) - Iport1)
```

```
ans =
1.8022e-15
```

```
abs(IT_9a(2) - Iport2)
```

```
ans =
2.1897e-15
```

```
% (b) two netwoeks interconnected at node 1 (N) ↔ node 5 (F)
ieee9_A1; Y9N = sparse(admittance(nfrom, nto, r, x, b)); I9N = Iint; nN =
size(Y9N,1);
ieee9_A1; Y9F = sparse(admittance(nfrom, nto, r, x, b)); I9F = Iint; nF =
size(Y9F,1);

[IT_9b, VNF_9b] = genfault(Y9N, Y9F, I9N, I9F, 1, 5);
VocF = Y9F \ I9F;
colF = Y9F \ sparse(5,1,1,nF,1);
VF_9b = VocF + colF * IT_9b;
fprintf('(5b) |VN(port1) - VF(port5)| = %.3e\n', abs(VNF_9b(1) - VF_9b(5)));
```

```
(5b) |VN(port1) - VF(port5)| = 1.388e-17
```

```
disp(table(abs(IT_9b), angle(IT_9b)*180/pi, 'VariableNames', {'|IT|','angle_deg'}));
```

```
      |IT|      angle_deg
    ________    _________

    0.056311     157.56
```

```
% (c) two networks interconnected at (3↔7) and (5↔4)
ieee9_A1; Y9Nc = sparse(admittance(nfrom, nto, r, x, b)); I9Nc = Iint; nNc =
size(Y9Nc,1);
ieee9_A1; Y9Fc = sparse(admittance(nfrom, nto, r, x, b)); I9Fc = Iint; nFc =
size(Y9Fc,1);

idN_c = [3 5]; idF_c = [7 4];
[IT_9c, VNF_9c] = genfault(Y9Nc, Y9Fc, I9Nc, I9Fc, idN_c, idF_c);
EportsF = sparse(idF_c, 1:numel(idF_c), 1, nFc, numel(idF_c));
colsF_c = Y9Fc \ EportsF;
VF_9c = (Y9Fc \ I9Fc) + colsF_c * IT_9c;
port_mismatch = norm(VNF_9c(idN_c) - VF_9c(idF_c), inf);
fprintf('(5c) ||VN(ports) - VF(ports)||_inf = %.3e\n', port_mismatch);
```

```
(5c) ||VN(ports) - VF(ports)||_inf = 9.021e-17
```

```
disp(table(idN_c(:), idF_c(:), abs(IT_9c), angle(IT_9c)*180/pi, ...
    'VariableNames', {'BusN','BusF','|IT|','angle_deg'}));
```

```
    BusN    BusF     |IT|      angle_deg
    ____    ____    _______    _________

     3       7      0.27351     -4.3887
     5       4      0.28421      175.93
```

```
% (d) IEEE-24 interconnected with itself at (7↔3), (13↔15), (23↔17)
ieee24_A1; Y24N = sparse(admittance(nfrom, nto, r, x, b)); I24N = Iint; n24N =
size(Y24N,1);
ieee24_A1; Y24F = sparse(admittance(nfrom, nto, r, x, b)); I24F = Iint; n24F =
size(Y24F,1);
```

```
idN_d = [7 13 23]; idF_d = [3 15 17];
[IT_24d, VNF_24d] = genfault(Y24N, Y24F, I24N, I24F, idN_d, idF_d);

EportsF24 = sparse(idF_d, 1:numel(idF_d), 1, n24F, numel(idF_d));
colsF24 = Y24F \ EportsF24;
VF_24d = (Y24F \ I24F) + colsF24 * IT_24d;
mismatch24 = norm(VNF_24d(idN_d) - VF_24d(idF_d), inf);
fprintf('(5d) ||VN(ports) - VF(ports)||_inf = %.3e\n', mismatch24);
```

```
(5d) ||VN(ports) - VF(ports)||_inf = 8.327e-17
```

```

% report voltages for first system and tie currents
Vm24 = abs(VNF_24d); Va24 = angle(VNF_24d)*180/pi;
T24V = table((1:n24N).', Vm24, Va24, 'VariableNames', {'Bus','|V|','Angle_deg'});
disp('IEEE-24 (first system) voltages after interconnection:');
```

```
IEEE-24 (first system) voltages after interconnection:
```

```
disp(T24V);
```

```
    Bus      |V|       Angle_deg
    ___    _______    _________

     1      1.0117     -4.5223
     2      1.0115     -4.8092
     3     0.95216     -1.5813
     4     0.96498     -6.4771
     5      1.0017     -6.5348
     6     0.97884     -13.408
     7      0.9957     -8.7532
     8     0.96511     -10.264
     9     0.96597     -3.6608
    10      1.0224     -5.4746
    11     0.98418      2.1057
    12      0.9999      2.9917
    13      1.0131       6.678
    14     0.96725      4.8457
    15      1.0138      15.799
    16      1.0152      14.706
    17      1.0417      19.147
    18      1.0552      20.413
    19      1.0163      12.152
    20      1.0297      11.861
    21      1.0562      21.232
    22      1.0629      26.868
    23      1.0409      12.425
    24     0.97357      9.4987
```

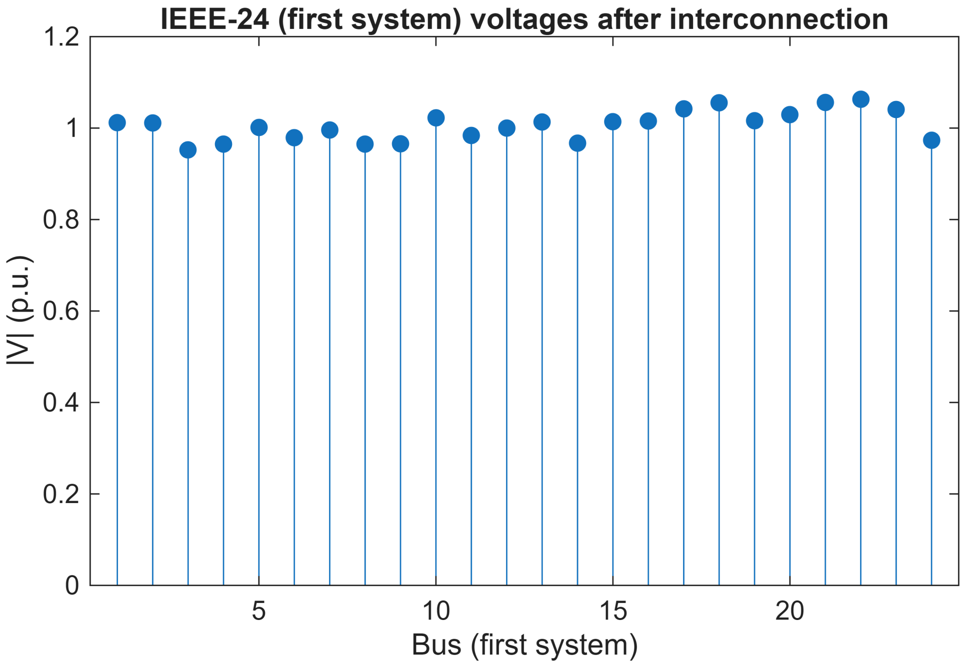
```

T24I = table(idN_d(:), idF_d(:), abs(IT_24d), angle(IT_24d)*180/pi, ...
    'VariableNames', {'BusN','BusF','|IT|','angle_deg'});
disp('IEEE-24 tie-line currents:');
```

```
IEEE-24 tie-line currents:
```

```
disp(T24I);
```

```
    BusN    BusF     |IT|     angle_deg
    ____    ____    ______    _________

     7       3      0.6285      5.0002 
    13      15      2.0075     -176.11 
    23      17      1.8032      4.8616 
```

```
figure; stem(1:n24N, Vm24, 'filled');
xlabel('Bus (first system)'); ylabel('|V| (p.u.)');
title('IEEE-24 (first system) voltages after interconnection');
```



In the final part of the assignment, we utilize our genfault function to simulate a variety of faults commonly seen in power systems, including line outages and interconnecting two systems.

a) Line outage (8–9): Removing line 8 caused modest voltage depressions near buses 8 and 9 compared to the intact case, as expected when connectivity weakens. Reconnecting the line via the genfault function reproduced the intact voltages with errors ~10^{-15}, confirming correctness.

b) Two IEEE-9 systems interconnected at nodes 1↔5: The tie current magnitude was 0.0563 p.u. at an angle of 157.6°, driven by voltage angle and magnitude differences. Port voltages matched across the boundary, consistent with continuity conditions.

c) Two IEEE-9 systems interconnected at (3↔7, 5↔4): Tie currents of 0.2735 p.u. and 0.2842 p.u. flowed, showing stronger exchanges due to multiple connections. Port voltage mismatches were negligible (~10^{-16}).

d) IEEE-24 self-interconnection at (7↔3, 13↔15, 23↔17): First-system voltages spanned 0.9522–1.0562 p.u. with angles between –13.4° and +26.9°. Tie currents were substantial, e.g., 2.0075 p.u. on the (13,15) link. These flows are consistent with angle differences driving power transfer.

Physically, these experiments confirm that genfault reproduces both outages and interconnections accurately. Voltage profiles change most at interconnection points, while remote buses remain largely unaffected. Tie currents align with power system theory: flows emerge to balance differences in voltage magnitude and phase across linked nodes.

Conclusion

Across all five problems, we validated both the mathematical correctness and physical realism of the results. Residuals, symmetry checks, and inverse relationships confirmed numerical accuracy, while voltage profiles and current behavior reflected real-world expectations.

The use of sparse formulations, avoidance of direct inversion, and consistent validation produced stable results throughout. Plots and tables reinforced the insights: voltages clustered near 1.0 p.u. under normal conditions, collapsed during faults, and shifted significantly under interconnections.

This assignment demonstrated how nodal methods, Thévenin equivalents, and generalized fault analysis together provide a robust and versatile toolkit for power system analysis.

Appendix

Link to repository containing all functions: https://github.com/dtemurcu/ECSE563_A1.git

Note: The repository is currently private, I will make it public after the submission deadline has passed.