A logo of a university

Description automatically generated

**Stephen J. R. Smith Faculty of Engineering & Applied Science**

**MREN-372: Numerical Methods and Optimization**

LAB-02: MATLAB Based Circuit Simulator v2.0

Aria Maz | 20278460

Ahmed Iqbal | 20272366

Thurvan Pakerathan | 20175944

February 22, 2024

Please find the Lab 2 Work Checklist in Figure 1.

A white paper with writing on it

Description automatically generatedFigure 1: Checked off and signed lab 2 Work Checklist.

# Prelab)

## Question 1)

A diagram of a circuit

Description automatically generated

Figure 2: Provided circuit, netlist, and component values for Fig. 1 circuit of lab manual.

### a)

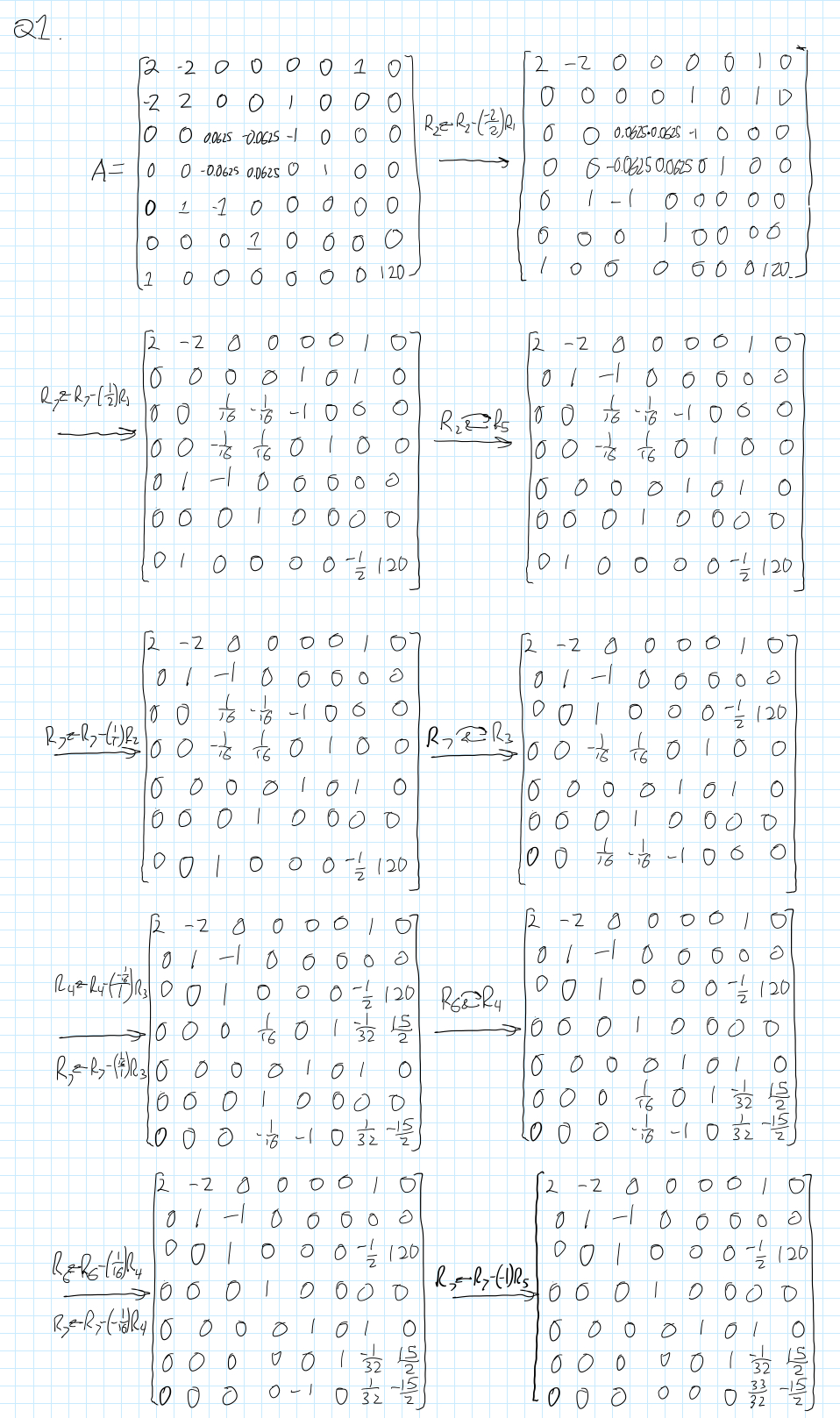


Figure 3: Manual Computation of node voltages and current of Figure 2, using Gaussian Elimination with partial pivoting.

A math equations on a graph paper

Description automatically generated

Figure 4: Node voltage and current Results of Figure 3.

### b)

A graph paper with numbers and equations

Description automatically generated

Figure 5: Manual computation of node voltages and currents of Figure 2, using PLU Decomposition.

## Question 2)

A math equations on a graph paper

Description automatically generated

Figure 6: Repeat of question 1 using vs = 60V.

When using gaussian elimination with partial decomposition, the source voltage in the rightmost column is switched out with 60V. Compared to PLU decomposition, Gaussian Elimination would be better for sweeping voltages.

# MATLAB Output)

## Question 2)

A number of numbers and symbols

Description automatically generated with medium confidence

Figure 7: MAPNETLIST output Matrix A and vector B of Figure 2. The results are the same as the provided calculations.

## Question 4)

A white screen with black text

Description automatically generated

Figure 8: Gaussian with partial pivoting and PLU decomposition solution of Figure 2, at 0Hz. Since the circuit is DC only, there is no phasor angle for any of the nodes. Note that the results are the same.

A screenshot of a computer code

Description automatically generated

Figure 9: Gaussian with partial pivoting and PLU decomposition solution of Figure 2, at 60Hz. Note that the results are the same.

Comparing Gaussian elimination with partial pivoting and PLU decomposition with forward/backward substitution result in the same solution. Hence either method should provide the same level of accuracy.

## A graph of a voltage Description automatically generatedQuestion 5)

Figure 10: Magnitude of node three voltage as the source voltage is swept from -60V to 60V with 0.01V steps, at 60Hz. The plot takes the form of an absolute function.

A diagram of a voltage

Description automatically generated

Figure 11: Phase of node three as the source voltage is swept.

## Question 6)

A black text with black text

Description automatically generated with medium confidence

Figure 12: Estimated execution time of PLU and Gaussian calculations.

As seen by Figure 12, Gaussian elimination with partial pivoting is faster by approximately 8 percent.

## Question 7)

### a)

The netlists for Fig. 2 and Fig. 3 from the lab two manual can be found in the MATLAB code appendix.

### A graph of a frequency response Description automatically generatedb)

A graph of a frequency response

Description automatically generatedFigure 13: Frequency response plot of Fig. 2 from the lab manual. Frequency range is from 0 Hz to 1 MHz, with 100 Hz step sizes.

Figure 14: Frequency response plot of Fig. 3 from the lab manual. Frequency range is from 0 Hz to 1 MHz, with 100 Hz step sizes.

## Question 8)

The frequency response curves of Fig. 2 and Fig. 3 are that of a low-pass and high-pass filter, respectively. The Fig. 2 circuit would filter out high frequency noise, while Fig. 3 would filter out low frequency noise. Note that on the frequency response curve in Figure 14, there is a “bump” at around 10 kHz, which is the resonant frequency of the circuit.

## Question 9)



Figure 15: -3dB corner frequency of Figure 13 is 998.6 Hz.

# Conclusion

In conclusion, this lab expanded our knowledge on circuit analysis and simulation, specifically with analyzing DC and AC sources. We learned how

# MATLAB Code)

## Question 1)

|  |
| --- |
| function [A, b] = mappNETLIST(filename, f)  % Open the file  fid = fopen(filename, 'r');    % Initialize a counter for the maximum node number  maxNode = 0;  extraRow = 0;  counter = 0;  %initalize s for inductor and capacitor  s = (2\*pi\*f)\*1i;  % First pass to determine the size of matrices A and b  while ~feof(fid)  line = fgetl(fid);  tokens = strsplit(line);  type = tokens{1}(1);  % Check node numbers and update maxNode if necessary  n1 = str2double(tokens{2});  n2 = str2double(tokens{3});  if type == 'V' || type == 'L'  extraRow = extraRow + 1;  end  if n1 > maxNode  maxNode = max(maxNode, n1);  end  if n2 > maxNode  maxNode = max(maxNode, n2);  end  end    % Close and reopen the file to reset the read position to the beginning  fclose(fid);  fid = fopen(filename, 'r');    % Initialize matrix A and vector b with the correct size  A = zeros(maxNode + extraRow);  b = zeros(maxNode + extraRow, 1);    % Read the file line by line and update A and b  while ~feof(fid)  line = fgetl(fid);  tokens = strsplit(line);    % Determine the type of component (Resistor or Current Source)  type = tokens{1}(1);  n1 = str2double(tokens{2});  n2 = str2double(tokens{3});  value = str2double(tokens{4});  % Update the matrices A and b based on the component type  if type == 'R'  % For resistors, update A matrix  G = 1 / value; % Conductance is the inverse of resistance  if n1 ~= 0  A(n1, n1) = A(n1, n1) + G;  end  if n2 ~= 0  A(n2, n2) = A(n2, n2) + G;  end  if n1 ~= 0 && n2 ~= 0  A(n1, n2) = A(n1, n2) - G;  A(n2, n1) = A(n2, n1) - G;  end  elseif type == 'I'  % For current sources, update b vector  if n1 ~= 0  b(n1) = b(n1) - value; % Current entering node n1, assumed negative  end  if n2 ~= 0  b(n2) = b(n2) + value; % Current leaving node n2, assumed positive  end  elseif type == 'V'  counter = counter + 1;  if n1 ~= 0 && n2 ~= 0  A(n1, maxNode + counter) = A(n1, maxNode + counter) - 1;  A(maxNode + counter, n1) = A(maxNode + counter, n1) - 1;  A(n2, maxNode + counter) = A(n2, maxNode + counter) + 1;  A(maxNode + counter, n2) = A(maxNode + counter, n2) + 1;    elseif n1 ~= 0 && n2 == 0  A(n1, maxNode + counter) = A(n1, maxNode + counter) - 1;  A(maxNode + counter, n1) = A(maxNode + counter, n1) - 1;  elseif n1 == 0 && n2 ~= 0  A(n2, maxNode + counter) = A(n2, maxNode + counter) + 1;  A(maxNode + counter, n2) = A(maxNode + counter, n2) + 1;  end  b(maxNode + counter) = value;  elseif type == 'L'  counter = counter + 1;  if n1 ~= 0 && n2 ~= 0  A(n1, maxNode + counter) = A(n1, maxNode + counter) - 1;  A(maxNode + counter, n1) = A(maxNode + counter, n1) - 1;  A(n2, maxNode + counter) = A(n2, maxNode + counter) + 1;  A(maxNode + counter, n2) = A(maxNode + counter, n2) + 1;  A(maxNode + counter, maxNode + counter) = A(maxNode + counter, maxNode + counter) - s\*value;    elseif n1 ~= 0 && n2 == 0  A(n1, maxNode + counter) = A(n1, maxNode + counter) - 1;  A(maxNode + counter, n1) = A(maxNode + counter, n1) - 1;  A(maxNode + counter, maxNode + counter) = A(maxNode + counter, maxNode + counter) - s\*value;  elseif n1 == 0 && n2 ~= 0  A(n2, maxNode + counter) = A(n2, maxNode + counter) + 1;  A(maxNode + counter, n2) = A(maxNode + counter, n2) + 1;  A(maxNode + counter, maxNode + counter) = A(maxNode + counter, maxNode + counter) - s\*value;  end  elseif type == 'C'  if n1 ~= 0 && n2 ~= 0  A(n1, n1) = A(n1, n1) + s\*value;  A(n1, n2) = A(n1, n2) - s\*value;  A(n2, n1) = A(n2, n1) - s\*value;  A(n2, n2) = A(n2, n2) + s\*value;  elseif n1 ~= 0 && n2 == 0  A(n1, n1) = A(n1, n1) + s\*value;  elseif n1 == 0 && n2 ~= 0  A(n2, n2) = A(n2, n2) + s\*value;  end  end  end    % Close the file  fclose(fid);  end |

## Question 2 & 4)

|  |
| --- |
| %q2, find netlist from figure 1 with mappNETLIST  [A\_Fig1, b\_Fig1] = mappNETLIST('Figure\_1.txt', 0);  disp('Question 2:');  disp('Matrix A:');  disp(A\_Fig1);  disp('Vector b:');  disp(b\_Fig1);  %%Q4a - 0 Hz  x\_fig1\_gauss = GaussElimPivot(A\_Fig1, b\_Fig1);  x\_fig1\_PLU = PLUSolver(A\_Fig1, b\_Fig1);  disp('Q4a - 0 Hz Results:');  disp('Gaussian Partial Pivot Solution:');  fprintf('');  fprintf("%0.4fV\n", x\_fig1\_gauss);  disp('PLU Decomposition Solution:');  fprintf("%0.4fV\n", x\_fig1\_PLU);  disp("Note: the phasor angle is 0° as there is no imaginary component.")  %%Q4b - 60Hz  %Gaussian solution  disp('Q4b - 60 Hz Results:')  [A\_Fig1, b\_Fig1] = mappNETLIST('Figure\_1.txt', 60);  disp('Gaussian Partial Pivot Solution:');  x\_fig1\_gauss = GaussElimPivot(A\_Fig1, b\_Fig1);  for i = x\_fig1\_gauss  x\_fig1\_gauss\_mag = abs(i);  x\_fig1\_gauss\_phase = angle(i);  x\_fig1\_gauss\_phase\_deg = rad2deg(x\_fig1\_gauss\_phase);  fprintf("%0.4fV ∠%0.4f° \n", x\_fig1\_gauss\_mag, x\_fig1\_gauss\_phase\_deg);  end  fprintf("\n");  %%PLU solution  disp('PLU Decomposition Solution:');  x\_fig1\_PLU = PLUSolver(A\_Fig1, b\_Fig1);  for i = x\_fig1\_PLU  x\_fig1\_PLU\_mag = abs(i);  x\_fig1\_PLU\_phase = angle(i);  x\_fig1\_PLU\_phase\_deg = rad2deg(x\_fig1\_PLU\_phase);  fprintf("%0.4fV ∠%0.4f° \n", x\_fig1\_gauss\_mag, x\_fig1\_gauss\_phase\_deg);  end  fprintf("\n"); |

## Question 3a)

|  |
| --- |
| function [x] = GaussElimPivot(A,b)  % GaussPivot: Gauss elimination pivoting  % x = GaussPivot(A,b): Gauss elimination with pivoting.  % input:  % A = coefficient matrix  % b = right hand side vector  % output:  % x = solution vector  [m,n]=size(A);  if m~=n, error('Matrix A must be square'); end  nb=n+1;  Aug=[A b];  % forward elimination  for k = 1:n-1  % partial pivoting  [big,i]=max(abs(Aug(k:n,k)));  ipr=i+k-1;  if ipr~=k  Aug([k,ipr],:)=Aug([ipr,k],:);  end  for i = k+1:n  factor=Aug(i,k)/Aug(k,k);  Aug(i,k:nb)=Aug(i,k:nb)-factor\*Aug(k,k:nb);  end  end  % back substitution  x=zeros(n,1);  x(n)=Aug(n,nb)/Aug(n,n);  for i = n-1:-1:1  x(i)=(Aug(i,nb)-Aug(i,i+1:n)\*x(i+1:n))/Aug(i,i);  end |

## Question 3 b)

|  |
| --- |
| function [x] = PLUSolver(A, b)  [L, U, P] = PLU(A);  y = L\(P\*b);  x = U\y;  end |

## Question 5 & 6)

|  |
| --- |
| %q5  close all;  vSourceMin=-60;  VSourceMax=60;  vSource\_Sampling = vSourceMin:0.01:VSourceMax;  v3\_Magnitude = [];  v3\_Phase = [];  v3\_Mag = [];  v3\_Ph = [];  v3\_Magnitude\_Gauss = [];  v3\_Phase\_Gauss = [];  v3\_Mag\_Gauss = [];  v3\_Ph\_Gauss = [];  t\_Gaussian = 0;  t\_PLU = 0;  %%PLU  for i = vSource\_Sampling  tic;  [A\_Fig1, b] = mappNETLIST('Figure\_1.txt', 60);  b(length(b)) = i;  [x\_fig1\_PLU] = PLUSolver(A\_Fig1, b);  v3\_Magnitude=abs(x\_fig1\_PLU(3));  v3\_Phase = angle(x\_fig1\_PLU(3))\*(180/pi);  v3\_Mag = [v3\_Mag, v3\_Magnitude];  v3\_Ph = [v3\_Ph, v3\_Phase];  t\_PLU = t\_PLU + toc;    end  fprintf('PLU CPU Time: %0.2f seconds\n', t\_PLU);  figure;  plot(vSource\_Sampling, v3\_Mag);  xlabel('Source Voltage (V)');  ylabel('Magnitude of Node 3 Voltage (V)');  title('Magnitude of Voltage at Node 3 vs Source Voltage (PLU)');  grid on;  figure;  plot(vSource\_Sampling, v3\_Ph);  xlabel('Source Voltage (V)');  ylabel('Phase of Node 3 Voltage (Deg)');  title('Phase of Voltage at Node 3 vs Source Voltage (PLU)');  grid on;  %%Gaussian calculation  for i = vSource\_Sampling  tic;  [A\_Fig1\_Gauss, b\_Gauss] = mappNETLIST('Figure\_1.txt', 60);  b\_Gauss(length(b\_Gauss)) = i;  [x\_fig1\_Gauss] = GaussElimPivot(A\_Fig1\_Gauss,b\_Gauss);  v3\_Magnitude\_Gauss=abs(x\_fig1\_Gauss(3));  v3\_Phase\_Gauss = angle(x\_fig1\_Gauss(3))\*(180/pi);  v3\_Mag\_Gauss = [v3\_Mag\_Gauss, v3\_Magnitude\_Gauss];  v3\_Ph\_Gauss = [v3\_Ph\_Gauss, v3\_Phase\_Gauss];  t\_Gaussian = t\_Gaussian + toc;  end  fprintf('Gaussian CPU Time:%0.2f seconds\n', t\_Gaussian);  figure;  plot(vSource\_Sampling, v3\_Mag\_Gauss);  xlabel('Source Voltage (V)');  ylabel('Magnitude of Node 3 Voltage (V)');  title('Magnitude of Voltage at Node 3 vs Source Voltage (Gaussian)');  grid on;  figure;  plot(vSource\_Sampling, v3\_Ph\_Gauss);  xlabel('Source Voltage (V)');  ylabel('Phase of Node 3 Voltage (Deg)');  title('Phase of Voltage at Node 3 vs Source Voltage (Gaussian)');  grid on; |

## Question 7 & 9)

|  |
| --- |
| %%Write a MATLAB script to call the mappNETLIST function and solve for the output voltage  % over a frequency range from 0 Hz to 1 MHz with a 100 Hz step size. The script should  % produce a plot of the magnitude voltage transfer function H (in dB) as a function of the frequency f.  %frequency range & sampling rate  f\_min = 0;  f\_max = 1e6;  f\_Sample = f\_min:100:f\_max;  %-3dB corner frequency  corner\_gain = -3;  %allocate space for voltage gains  voltage\_gain\_fig2 = zeros(1,10001);  voltage\_gain\_fig3 = zeros(1,10001);  counter = 0;  %Figure 2  for i = f\_Sample  counter = counter + 1 ;  [A, b] = mappNETLIST('figure\_2.txt', i);  x\_fig2 = PLUSolver(A,b);  voltage\_gain\_fig2(counter) = 20\*log10(abs(x\_fig2(2)/1));  end  figure;  semilogx(f\_Sample, voltage\_gain\_fig2);  xlabel('Frequency (Hz)');  ylabel('Magnitude Voltage Transfer Function H (dB)');  title('Frequency Response - Fig. 2');  grid on;  %q9, use interp1 function to find -3dB corner frequency  corner\_freq = interp1(voltage\_gain\_fig2, f\_Sample, corner\_gain);  fprintf('The -3dB corner frequency for the circuit in fig. 2 is: %0.2f Hz\n', corner\_freq);  %Figure 3  counter = 0;  for i = f\_Sample  counter = counter + 1;  [A, b] = mappNETLIST('figure\_3.txt', i);  x\_fig3 = PLUSolver(A,b);  voltage\_gain\_fig3(counter) = 20\*log(abs(x\_fig3(3)/10));  end  figure;  semilogx(f\_Sample, voltage\_gain\_fig3);  xlabel('Frequency (Hz)');  ylabel('Magnitude Voltage Transfer Function H (dB)');  title('Frequency Response - Fig. 3');  grid on; |

## Figure 1 Netlist)

|  |
| --- |
| Rg 1 2 500e-3  Lg 3 2 3.183e-3  Rm 3 4 16  Lm 0 4 31.83e-3  Cx 3 0 132.6e-6  Vs 0 1 120 |

## Figure 2 Netlist)

|  |
| --- |
| R1 1 2 1.59  C1 2 0 100e-6  Vin 0 1 1 |

## Figure 3 Netlist)

|  |
| --- |
| Rf 1 2 4  Cf 2 3 2e-6  Lf 3 0 127e-6  Vin 0 1 10 |