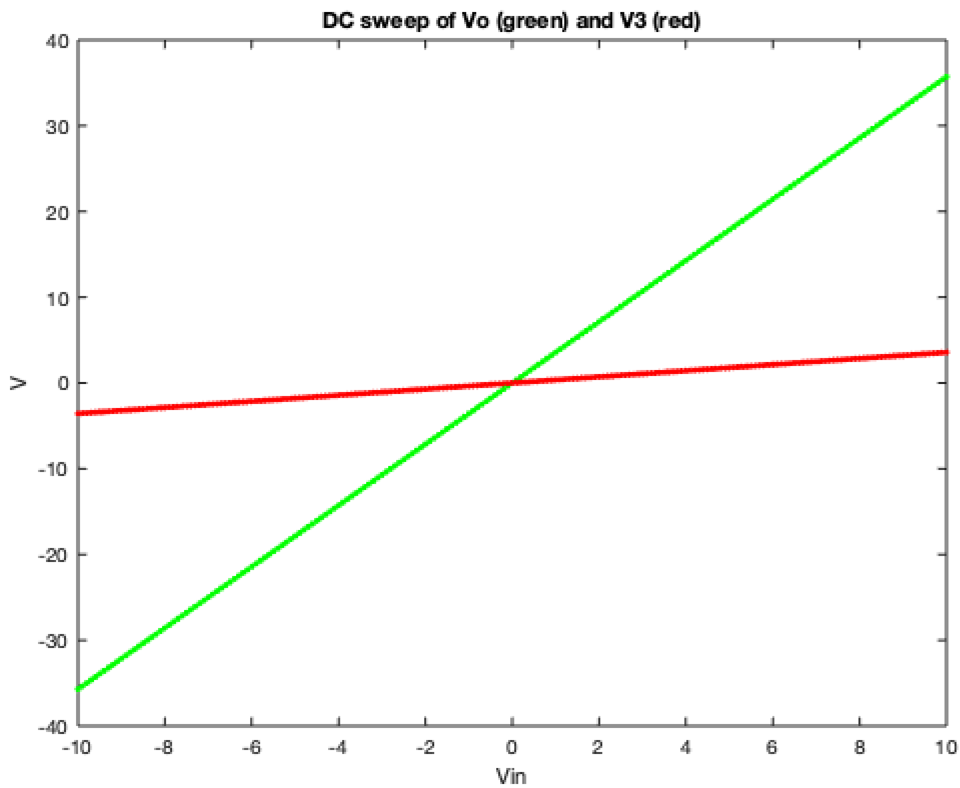
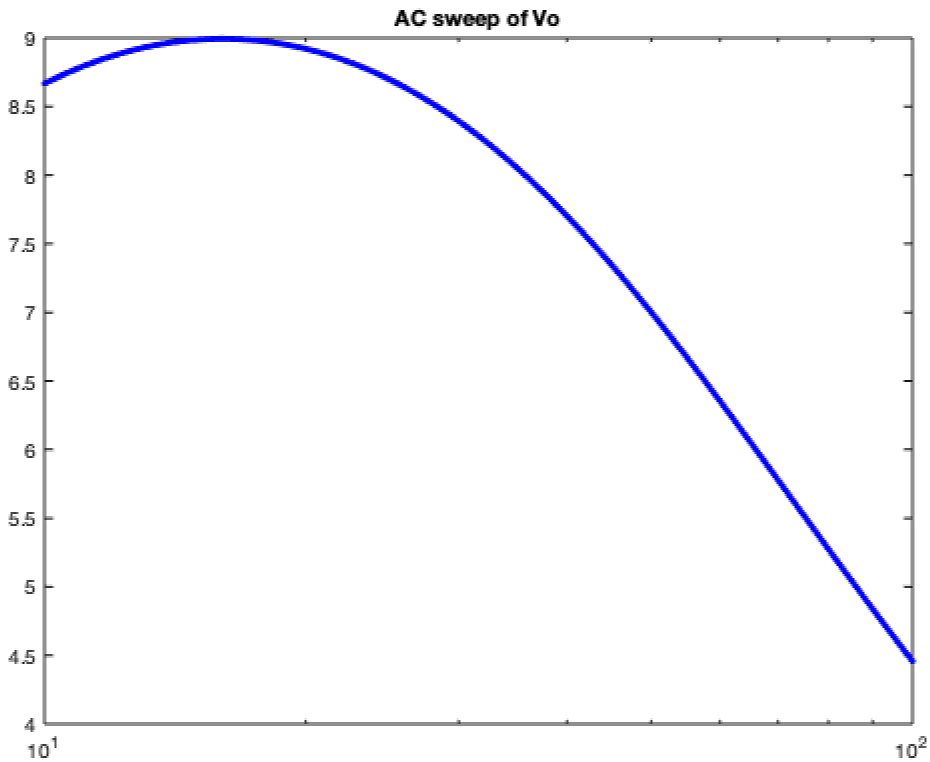
Assignment 4 Part 1

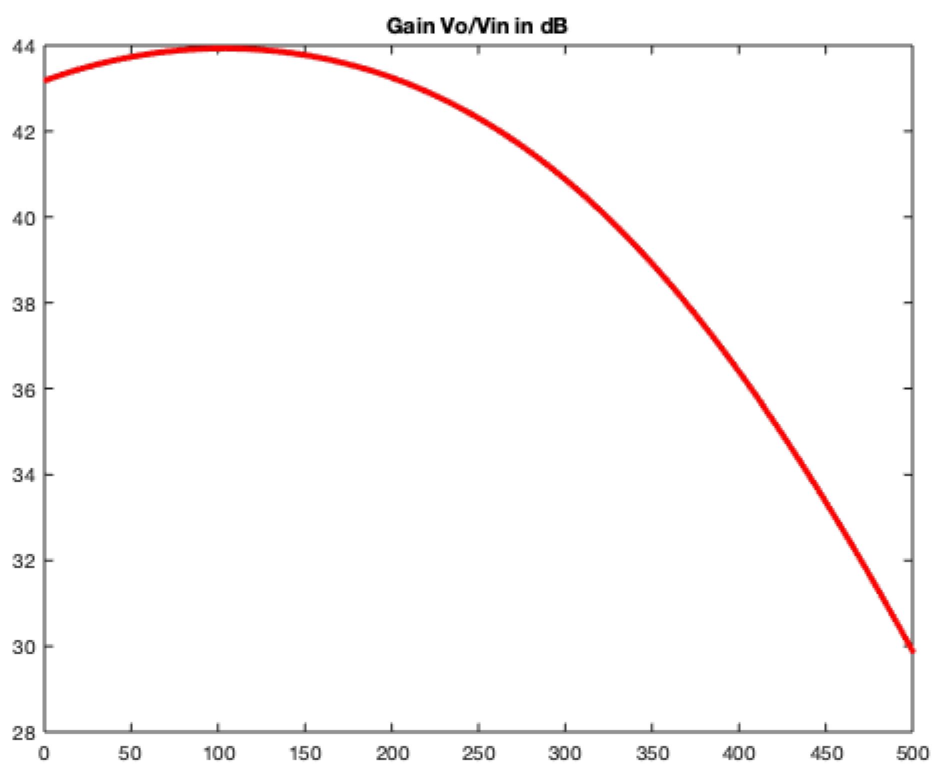
```
% Umeh Ifeanyi 101045786

%In this part of the assignment, we are repeating the work completed
 in
%pa9 and reporting on it.

%In part a)the C, G matrices and the F vector was created to describe
 the
%circuit network

%In pat b), the input voltage was DC swept from -10V to 10V and Vo and
 V3 was
%plotted. Then for the AC case, Vo was plotted as a function of w, and
 the
%gain, Vo/V1 was plotted in dB. Then,for the AC case, the gain was
 plotted
%as a function of random peturbations on C using a normal
 distribution
%with stf = 0.5 at w = pi. The gain was then plotted using histograms

% Definition of variables based on the components present in the
 circuit
R1 = 1;
G1 = 1/R1;
C = 0.25;
R2 = 2;
G2 = 1/R2;
L = 0.2;
R3 = 10;
G3 = 1/R3;
alpha = 100;
R4 = 0.1;
G4 = 1/R4;
RO = 1000;
GO = 1/RO;

% Definition of Matrices
C_matrix = [0 0 0 0 0 0 0;
           -C C 0 0 0 0 0;
            0 0 -L 0 0 0 0;
            0 0 0 0 0 0 0;
            0 0 0 0 0 0 0;
            0 0 0 0 0 0 0;
            0 0 0 0 0 0 0;];

G_Matrix = [1 0 0 0 0 0 0;
           -G2 G1+G2 -1 0 0 0 0;
            0 1 0 -1 0 0 0;
            0 0 -1 G3 0 0 0;
            0 0 0 0 -alpha 1 0;
```

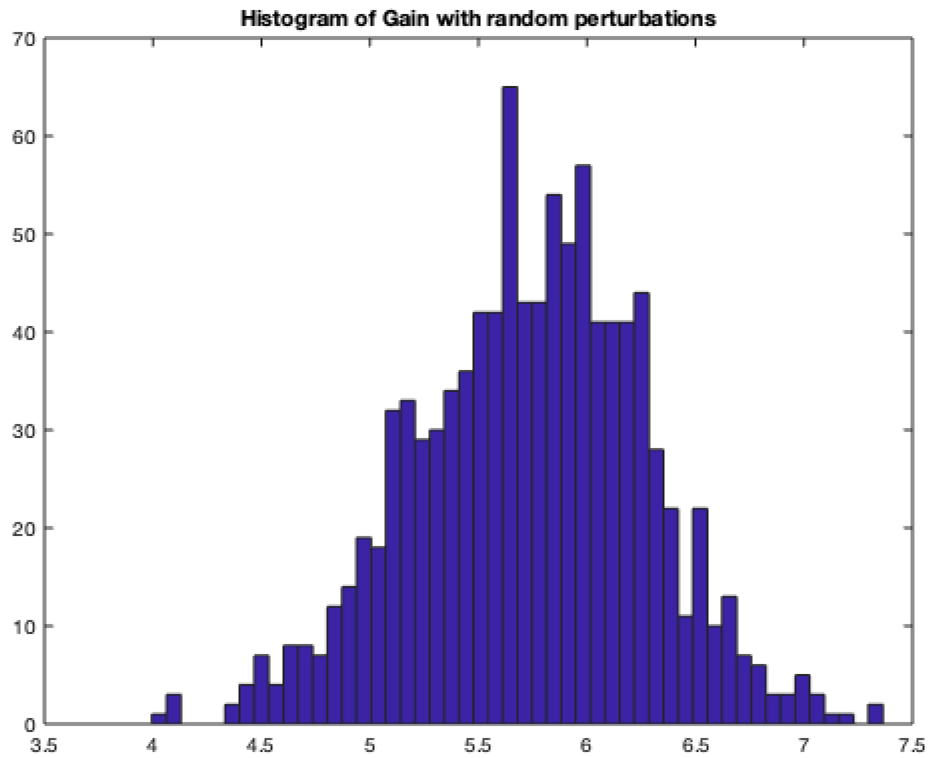
```
            0 0 0 G3 -1 0 0;
            0 0 0 0 0 -G4 G4+GO];

% Defining DC and AC voltage matrices, as well as the F matrix

V_DC = zeros(7,1);
V_AC = zeros(7,1);
F_Matrix = zeros(7,1);

% DC Sweep Plot
for vol = -10:0.1: 10

    F_Matrix(1,1) = vol;
    V_DC = G_Matrix\F_Matrix;                          % DC sweep
 calculation

    figure(1)
    plot(vol, V_DC(7,1), 'g.')
    hold on

    plot(vol, V_DC(4,1), 'r.')
    hold on
    title('DC sweep of Vo (green) and V3 (red)')
    xlabel('Vin')
    ylabel('V')

end

% AC Sweep and Gain Plot
w = logspace(1,2,500);
F_Matrix(1) = 1;

for i = 1:length(w)

    V_AC = (G_Matrix+C_matrix*1j*w(i))\F_Matrix;  % calculating the
 voltage matrix using AC sweep
    figure(2)
    semilogx(w(i), abs(V_AC(7,1)), 'b.')
    hold on
    title('AC sweep of Vo')

    dB = 20*log(abs(V_AC(7,1))/F_Matrix(1));    % Calculating the gain
    figure(3)
    plot(i, dB, 'r.')
    hold on
    title('Gain Vo/Vin in dB')
end

% AC case: voltage gain calculation as a function of random
 perturbations
% on C using a normal distribution with std = .05 and w = pi
pert =  0.25 + 0.05.*randn(1,1000);
w = pi;
Gain = zeros(1000,1);
```

```
for n = 1:length(Gain)

    C = pert(n);
    C_matrix(2,1) = -C;
    C_matrix(2,2) = C;
    V_AC = (G_Matrix+C_matrix*1j*w)\F_Matrix;   % Voltage calculation
 using AC sweep
    Gain(n,1) = abs(V_AC(7,1))/F_Matrix(1);    % Gain calculation
 using AC voltage
end

% Gain histogram
figure(4)
hist(Gain,50);
title('Histogram of Gain with random perturbations')
```




AC sweep of Vo
9
8.5
8
7.5
7
6.5
6
5.5
5
4.5
4
10^1
10^2


Gain Vo/Vin in dB
44
42
40
38
36
34
32
30
28
0
50
100
150
200
250
300
350
400
450
500



Published with MATLAB® R2019b

Assignment 4 Part 2

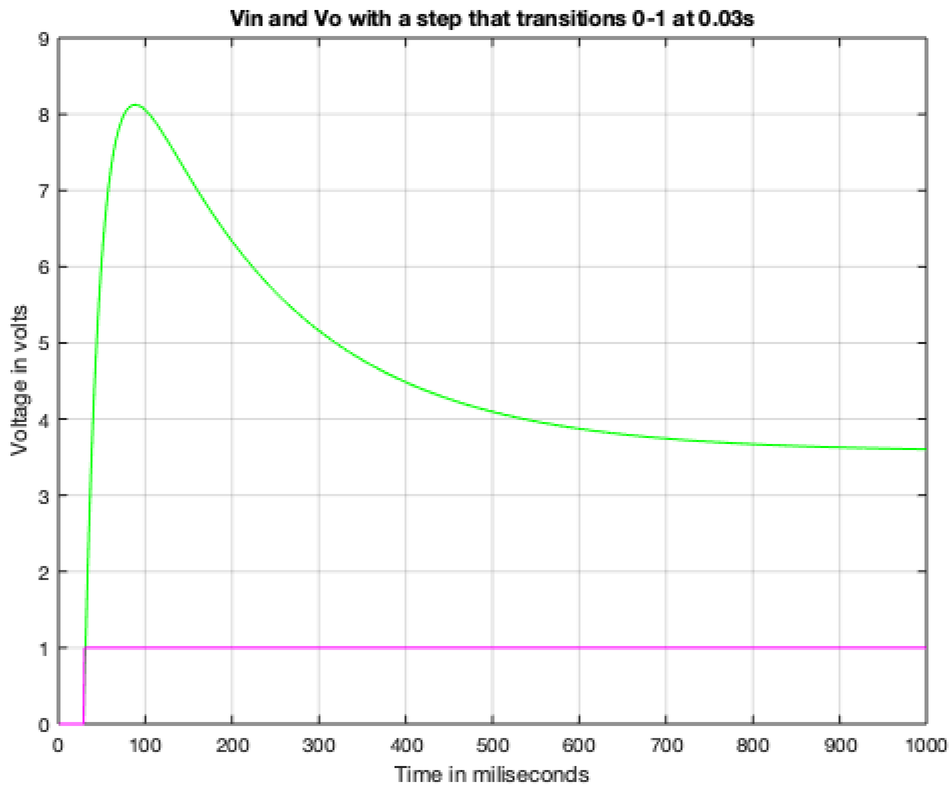
```
% In this part of the assignment, the transient response of the
 circuit was
% simulated

% Part a) Upon inspection of the circuit, this is a RLC circuit, due
 to the
% presence of capacitors, resistors and and inductors. DC transient
 response
% in RLC circuits can be observed when a sudden voltage or current is
% applied to it, which is what we will be doing in part 2

%Part b) In simple terms, RLC circuits experience "resonance" at a
 certain
%frequency. This frequency point is where the reactive inductance of
 the
%inductor equals the value of the capacitance reactance of the
 capacitor
%(xL = xC). At this frequency the gain of the circuit sharply rises,
 and is
%low at other frequencies. For this reason, this circuit has a very
 sharp
%bandpass response, and can be used as a filter or an amplifier
 functioning
%at a certain frequency.

% Definition of variables based on the components present in the
 circuit
R1 = 1;
G1 = 1/R1;
C = 0.25;
R2 = 2;
G2 = 1/R2;
L = 0.2;
R3 = 10;
G3 = 1/R3;
alpha = 100;
R4 = 0.1;
G4 = 1/R4;
RO = 1000;
GO = 1/RO;
Vin = 1;

% Define Matrices
C_Matrix = [0 0 0 0 0 0 0;
           -C C 0 0 0 0 0;
            0 0 -L 0 0 0 0;
            0 0 0 0 0 0 0;
            0 0 0 0 0 0 0;
            0 0 0 0 0 0 0;
            0 0 0 0 0 0 0;];
```

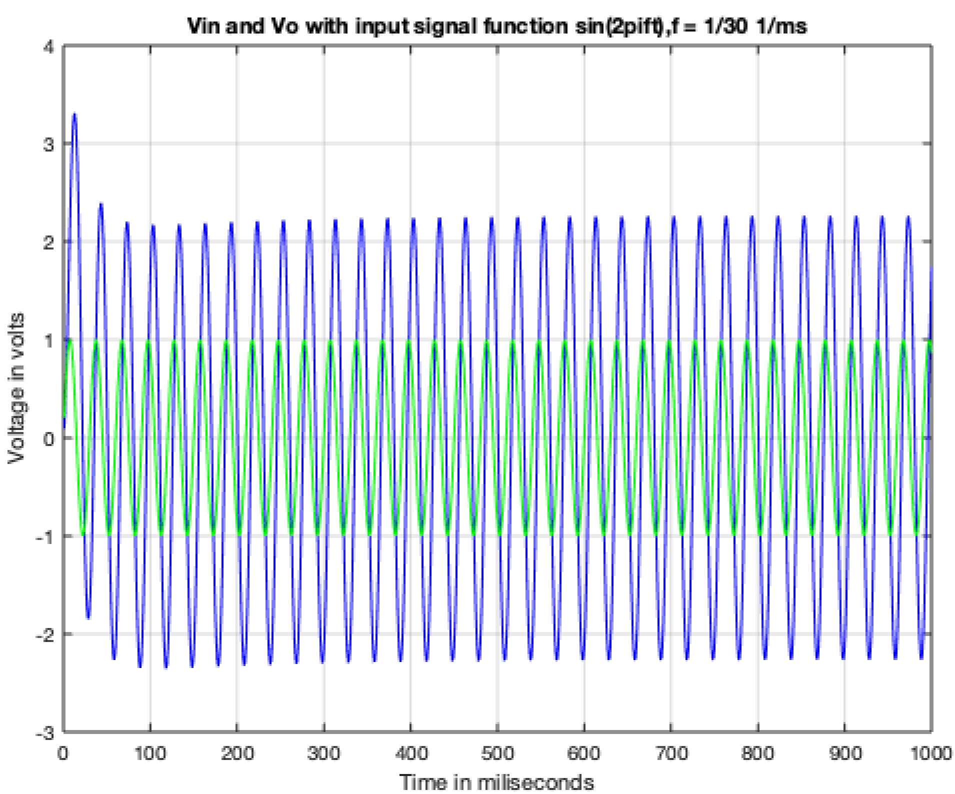
```
G_Matrix = [1 0 0 0 0 0 0;
           -G2 G1+G2 -1 0 0 0 0;
            0 1 0 -1 0 0 0;
            0 0 -1 G3 0 0 0;
            0 0 0 0 -alpha 1 0;
            0 0 0 G3 -1 0 0;
            0 0 0 0 0 -G4 G4+GO];

F_Matrix = [Vin;
             0;
             0;
             0;
             0;
             0;
             0;];

F0_Matrix = [Vin-Vin;
                0;
                0;
                0;
                0;
                0;
                0;];

%d) we will be simulating the circuit for 1 second using 1000 steps
step = 1000;


vol_1 = zeros(7, step);
vol_start = zeros(7, 1);
dt = 10^-3;
 %setting up the plot for the first input signal, a step that
 transitions
 %from 0 to 1 at 30 miliseconds
for i = 1:step

    if i < 30
        vol_1(:,i) = (C_Matrix./dt+G_Matrix)\(F0_Matrix
+C_Matrix*vol_start/dt);

    elseif i == 30
        vol_1(:,i) = (C_Matrix./dt+G_Matrix)\(F_Matrix
+C_Matrix*vol_start/dt);

    else
        vol_1(:,i) = (C_Matrix./dt+G_Matrix)\(F_Matrix
+C_Matrix*vol_old/dt);

    end

    vol_old = vol_1(:, i);

end
```

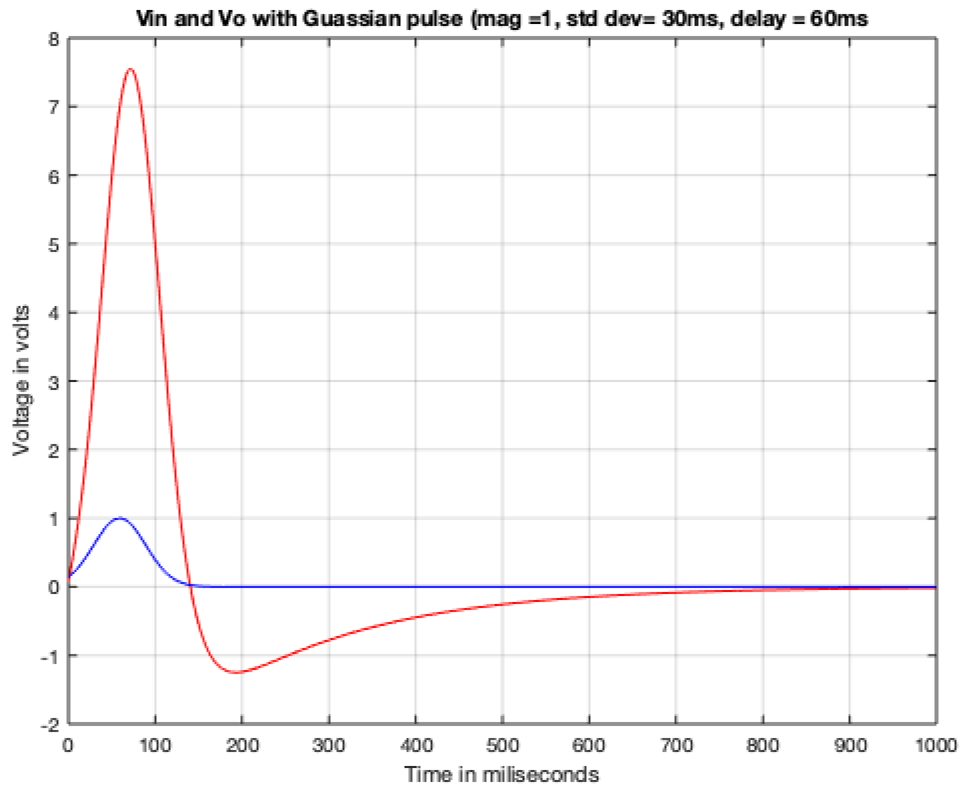
```
figure(1)
plot(1:step, vol_1(7,:), 'g')
hold on
plot(1:step, vol_1(1,:), 'm')
title('Vin and Vo with a step that transitions 0-1 at 0.03s')
xlabel('Time in miliseconds')
ylabel('Voltage in volts')
grid on


vol_2 = zeros(7, step);
function_F = zeros(7,1);

%setting up the plot for the second input signal, a sin(2*pi*f*t)
 signal,
%at a frequency of 1/(30) 1/ms.

for i_2 = 1:step

    function_vol = sin(2*pi*(1/0.03)*i_2/step);
    function_F(1,1) = function_vol;
    if i_2 == 1
        vol_2(:,i_2) = (C_Matrix./dt+G_Matrix)\(function_F
+C_Matrix*vol_start/dt);
    else
        vol_2(:,i_2) = (C_Matrix./dt+G_Matrix)\(function_F
+C_Matrix*vol_old/dt);
    end
    vol_old = vol_2(:, i_2);

end

figure(2)
plot(1:step, vol_2(7,:), 'b')
hold on
plot(1:step, vol_2(1,:), 'g')
title('Vin and Vo with input signal function sin(2pift),f = 1/30 1/
ms')
xlabel('Time in miliseconds')
ylabel('Voltage in volts')
grid on


%setting up the plot using a guassian pulse with mag=1, std dev = 30ms
 and
%delay of 60 ms

vol_3 = zeros(7, step);
Gaussian_F = zeros(7,1);

for i_3 = 1:step

    Guassian_vol = exp(-1/2*((i_3/step-0.06)/(0.03))^2);
```

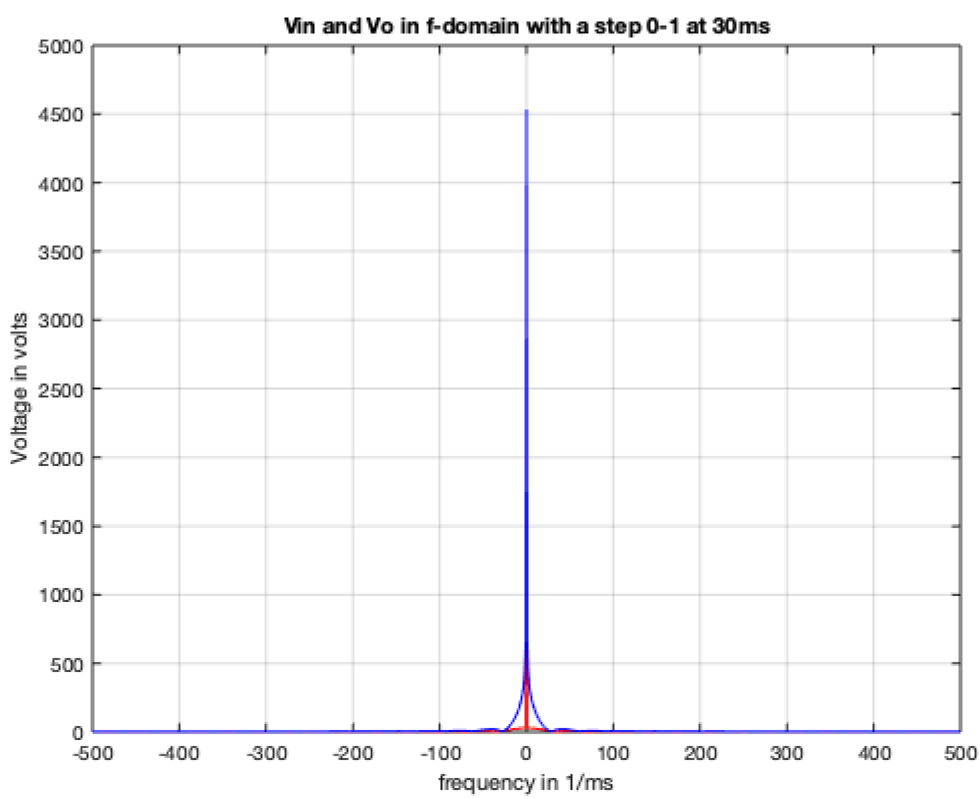
```
    Gaussian_F(1,1) = Guassian_vol;

    if i_3 == 1
        vol_3(:,i_3) = (C_Matrix./dt+G_Matrix)\(Gaussian_F
+C_Matrix*vol_start/dt);

    else
        vol_3(:,i_3) = (C_Matrix./dt+G_Matrix)\(Gaussian_F
+C_Matrix*vol_old/dt);

    end

    vol_old = vol_3(:, i_3);

end

figure(3)
plot(0:step-1, vol_3(7,:), 'r')
hold on
plot(0:step-1, vol_3(1,:), 'b')
title('Vin and Vo with Guassian pulse (mag =1, std dev= 30ms, delay =
 60ms')
xlabel('Time in miliseconds')
ylabel('Voltage in volts')
grid on

% Part d) iv. Now that the simulation is complete, the frequency
 content of
% the input and output signals will be plotted using the built-in
 matlab
% functions fft() and fftshift().

freq = (-step/2:step/2-1);

%Plot of Vin, Vo with first input signal in f-domain

fft_vol1_in = fft(vol_1(1, :));
fft_vol1_out = fft(vol_1(7, :));
ffts_vol1_in = fftshift(fft_vol1_in);
ffts_vol1_out = fftshift(fft_vol1_out);

figure(4)
plot(freq, abs(ffts_vol1_in), 'r')
hold on
plot(freq, abs(ffts_vol1_out), 'b')
title('Vin and Vo in f-domain with a step 0-1 at 30ms')
xlabel('frequency in 1/ms')
ylabel('Voltage in volts')
grid on

%Plot of Vin, Vo with second input signal in f-domain

fft_vol2 = fft(vol_2.');
ffts_vol2 = fftshift(fft_vol2);
```

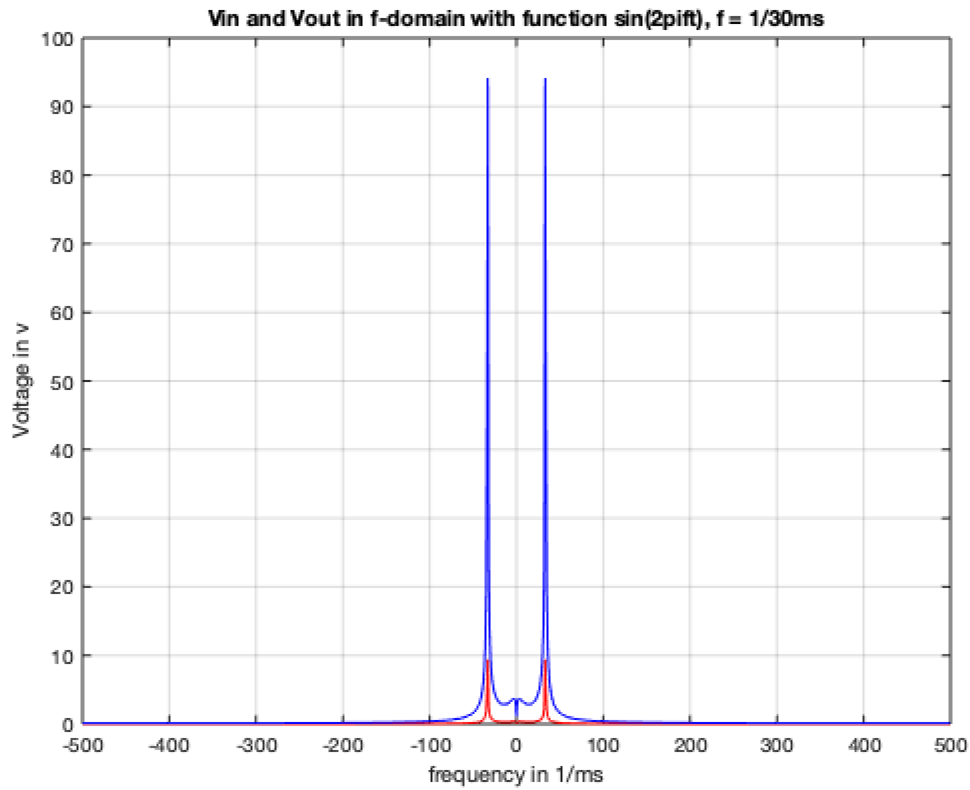
```
figure(5)
plot(freq, abs(ffts_vol2(:, 1)), 'r')
hold on
plot(freq, abs(ffts_vol2(:, 7)), 'b')
title('Vin and Vout in f-domain with function sin(2pift), f = 1/30ms')
xlabel('frequency in 1/ms')
ylabel('Voltage in v')
grid on

%Plot of Vin, Vo with third input signal in f-domain

fft_vol3 = fft(vol_3.');
ffts_vol3 = fftshift(fft_vol3);


figure(6)
plot(freq, abs(ffts_vol3(:, 1)), 'r')
hold on
plot(freq, abs(ffts_vol3(:, 7)), 'b')
title('Vin and Vout in f-domain with Guassian pulse (mag =1, std dev =
 30ms, delay = 60ms')
xlabel('frequency in 1/ms')
ylabel('Voltage in volts')
grid on

%%%%%%%%end of part 2%%%%%%%%%%%%%
```

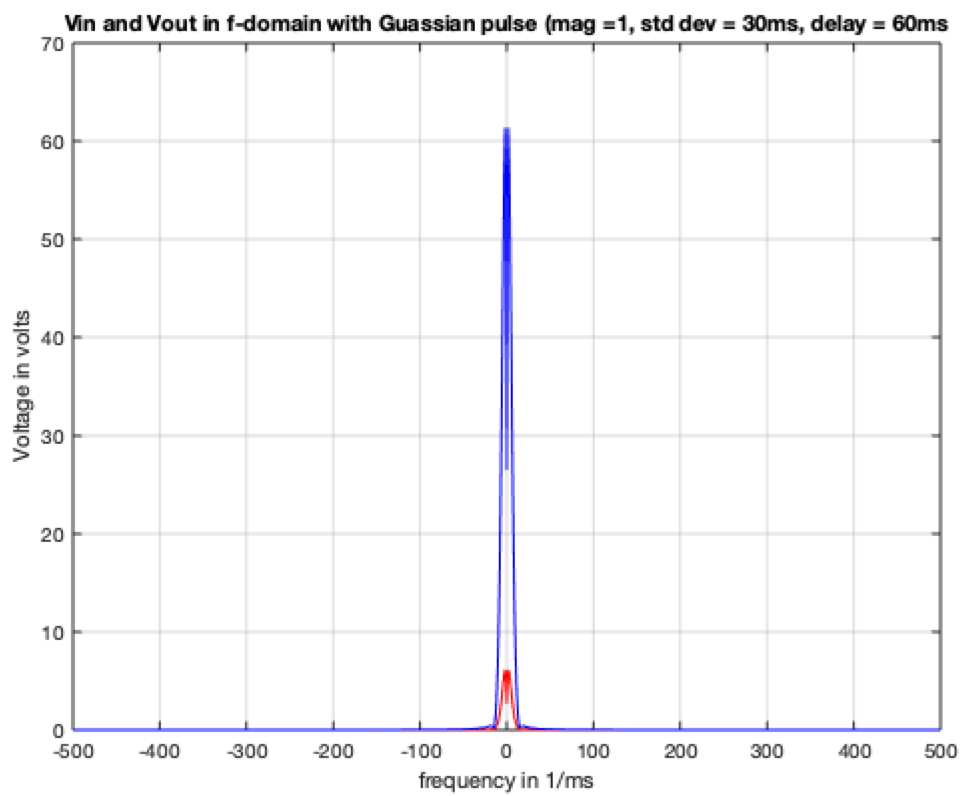

Vin and Vo with a step that transitions 0-1 at 0.03s
Voltage in volts
Time in miliseconds
0
1
2
3
4
5
6
7
8
9
0
100
200
300
400
500
600
700
800
900
1000


Vin and Vo with input signal function sin(2pift),f = 1/30 1/ms
Voltage in volts
Time in miliseconds
-3
-2
-1
0
1
2
3
4
0
100
200
300
400
500
600
700
800
900
1000


Vin and Vo with Guassian pulse (mag =1, std dev= 30ms, delay = 60ms
Voltage in volts
Time in miliseconds
8
7
6
5
4
3
2
1
0
-1
-2
0
100
200
300
400
500
600
700
800
900
1000


Vin and Vo in f-domain with a step 0-1 at 30ms
Voltage in volts
frequency in 1/ms
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-500
-400
-300
-200
-100
0
100
200
300
400
500


Vin and Vout in f-domain with function sin(2pift), f = 1/30ms
Voltage in v
frequency in 1/ms
100
90
80
70
60
50
40
30
20
10
0
-500
-400
-300
-200
-100
0
100
200
300
400
500


Vin and Vout in f-domain with Guassian pulse (mag =1, std dev = 30ms, delay = 60ms
Voltage in volts
frequency in 1/ms
70
60
50
40
30
20
10
0
-500
-400
-300
-200
-100
0
100
200
300
400
500

Published with MATLAB® R2019b

Table of Contents

Assignment 4 Part 3

```
% In this part of the assignment, noise was added to the circuit, and
 the
% response to this noise was observed

% In part a), a current source In was added to the circuit, in
 parallel
% with R3. This causes thermal noise to be generated in resistor R3.

%In part b), a  capacitor Cn = 0.00001 added in parallel with resistor
 to
%BW limit the noise. This causes changes to C matrix

% Definition of variables based on the components present in the
 circuit
R1 = 1;
G1 = 1/R1;
c = 0.25;
R2 = 2;
G2 = 1/R2;
L = 0.2;
R3 = 10;
G3 = 1/R3;
alpha = 100;
R4 = 0.1;
G4 = 1/R4;
RO = 1000;
GO = 1/RO;
Vin = 1;
Cn_1 = 0.00001;                % Capacitance value given in part b)
Cn_2 = 10^-8;                  % Cn_2 and Cn_3 are for part d) vi.
 where
Cn_3 = 2.6*2e-5;                 % we change Cn to observe the change
 in BW

% part a) Updating the C matrices

C_Matrix1 = [0 0 0 0 0 0 0;
            -c c 0 0 0 0 0;
             0 0 -L 0 0 0 0;
             0 0 0 -Cn_1 0 0 0;
             0 0 0 0 0 0 0;
             0 0 0 -Cn_1 0 0 0;
```

```
             0 0 0 0 0 0 0;];

C_Matrix2 = [0 0 0 0 0 0 0;
            -c c 0 0 0 0 0;
             0 0 -L 0 0 0 0;
             0 0 0 -Cn_2 0 0 0;
             0 0 0 0 0 0 0;
             0 0 0 -Cn_2 0 0 0;
             0 0 0 0 0 0 0;];

C_Matrix3 = [0 0 0 0 0 0 0;
            -c c 0 0 0 0 0;
             0 0 -L 0 0 0 0;
             0 0 0 -Cn_3 0 0 0;
             0 0 0 0 0 0 0;
             0 0 0 -Cn_3 0 0 0;
             0 0 0 0 0 0 0;];

GO = [1 0 0 0 0 0 0;
    -G2 G1+G2 -1 0 0 0 0;
      0 1 0 -1 0 0 0;
      0 0 -1 G3 0 0 0;
      0 0 0 0 -alpha 1 0;
      0 0 0 G3 -1 0 0;
      0 0 0 0 0 -G4 G4+GO];

F_Matrix = [Vin;
             0;
             0;
             0;
             0;
             0;
             0;];

F0_Matrix = [Vin-Vin;
                0;
                0;
                0;
                0;
                0;
                0;];

step_1 = 1000;
step_2 = 1.9898e4;          % time step 1 was the numer of time steps
 given
                            %   in part 2, step 2 is used
                            % to observe the effects of varying this
 value
                            % on the simulation

vol_1 = zeros(7, step_1);
vol_start = zeros(7, 1);
```

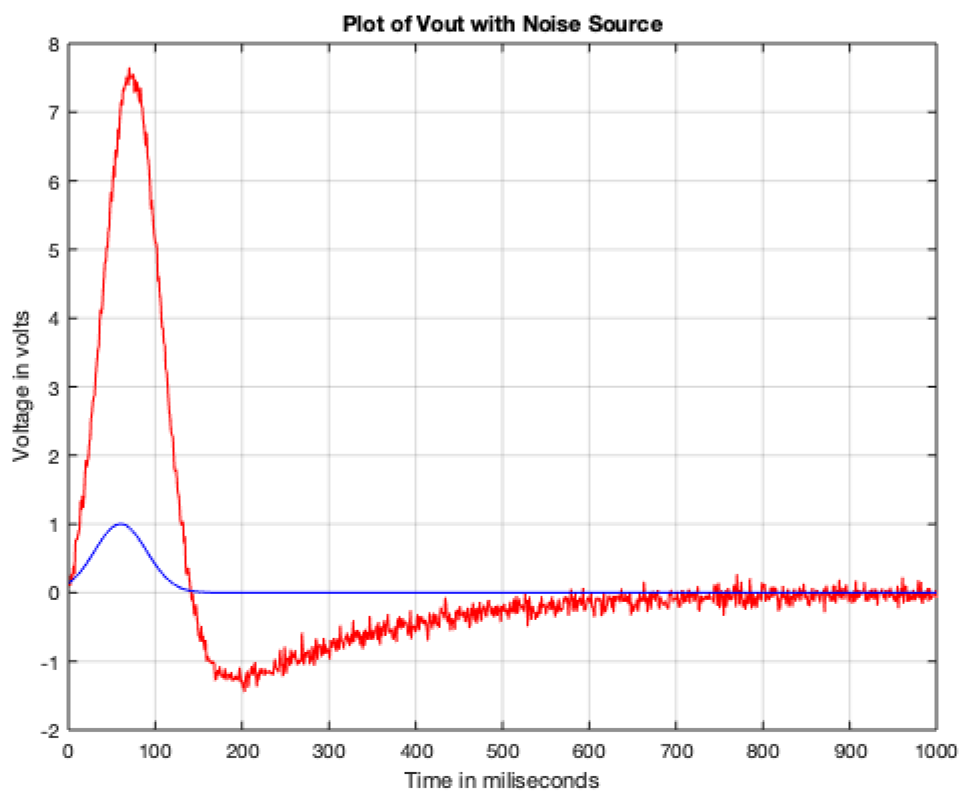
```
dt_1 = 10^-3;
dt_2 = 1.9898*10^-4;  %varying the timestep to observe the effects on
 the sim

% Circuit with Noise simulation with default time step
% Time domain simulation
%vol_1 = zeros(7, step_1);
Guassian_F = zeros(7,1);

for i = 1:step_1

    Guassian_F(1,1) = exp(-1/2*((i/step_1-0.06)/(0.03))^2);
    Guassian_F(4,1) = 0.001*randn();
    Guassian_F(7,1) = 0.001*randn();

    if i == 1
        vol_1(:,i) = (C_Matrix1./dt_1+GO)\(Guassian_F
+C_Matrix1*vol_start/dt_1);

    else
        vol_1(:,i) = (C_Matrix1./dt_1+GO)\(Guassian_F
+C_Matrix1*vol_old/dt_1);

    end

    vol_old = vol_1(:, i);

end

% Part b)i. modelling Vo signal with noise using the Guassian
 excitation

figure(1)
plot(1:step_1, vol_1(7,:), 'r')
hold on
plot(1:step_1, vol_1(1,:), 'b')
title('Plot of Vout with Noise Source')
xlabel('Time in miliseconds')
ylabel('Voltage in volts')
grid on


freq = (-step_1/2:step_1/2-1);

fft_vol1 = fft(vol_1.');
ffts_vol1 = fftshift(fft_vol1);

%Part c) Fourier Transform plot
figure(2)
plot(freq, abs(ffts_vol1(:, 1)), 'r')
hold on
plot(freq, abs(ffts_vol1(:, 7)), 'b')
title('Fourier-Transform Plot of Vout')
xlabel('frequency in 1/ms')
```

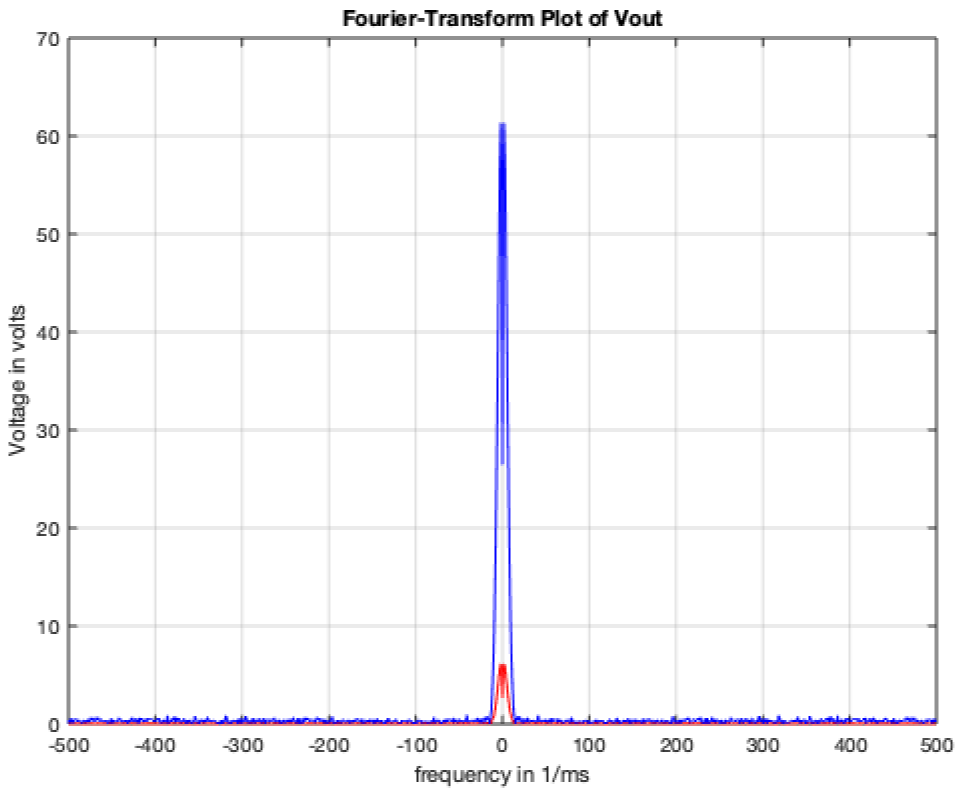
```
ylabel('Voltage in volts')
grid on


vol_2 = zeros(7, step_1);
Guassian_F = zeros(7,1);

for i_2 = 1:step_1

    Guassian_F(1,1) = exp(-1/2*((i_2/step_1-0.06)/(0.03))^2);
    Guassian_F(4,1) = 0.001*randn();
    Guassian_F(7,1) = 0.001*randn();

    if i_2 == 1

        vol_2(:,i_2) = (C_Matrix2./dt_1+GO)\(Guassian_F
+C_Matrix2*vol_start/dt_1);

    else

        vol_2(:,i_2) = (C_Matrix2./dt_1+GO)\(Guassian_F
+C_Matrix2*vol_old/dt_1);

    end

    vol_old = vol_2(:, i_2);

end

%   e) in part e), 3 plots of vout will be made, with each plot made
 using
%   a different value of Cout. A discussion on my findings is placed
 at the
%   end of this document.


% plotting Vout using smaller value of Cout
figure(3)
plot(1:step_1, vol_2(7,:), 'r')
hold on
plot(1:step_1, vol_2(1,:), 'b')
title('Vout plot using smaller value of Cout')
xlabel('Time im milliseconds)')
ylabel('Voltage in volts')
grid on

vol_3 = zeros(7, step_1);
Guassian_F = zeros(7,1);

for i_3 = 1:step_1

    Guassian_F(1,1) = exp(-1/2*((i_3/step_1-0.06)/(0.03))^2);
    Guassian_F(4,1) = 0.001*randn();
    Guassian_F(7,1) = 0.001*randn();
```

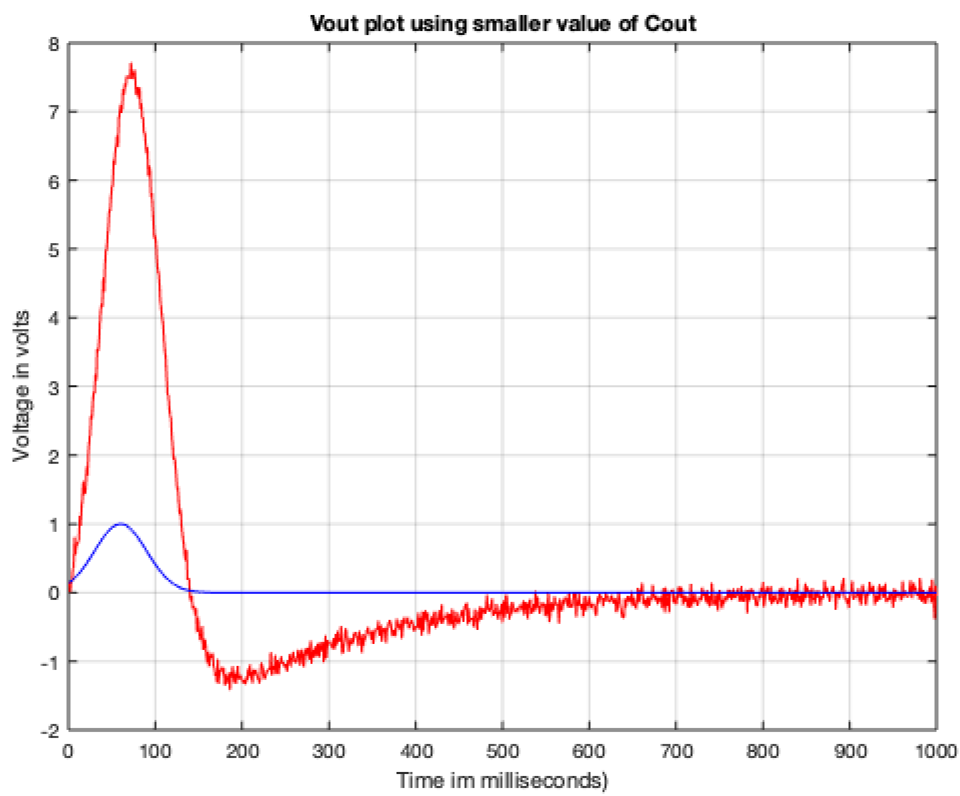
```
    if i_3 == 1

        vol_3(:,i_3) = (C_Matrix3./dt_1+GO)\(Guassian_F
+C_Matrix3*vol_start/dt_1);

    else

        vol_3(:,i_3) = (C_Matrix3./dt_1+GO)\(Guassian_F
+C_Matrix3*vol_old/dt_1);

    end

    vol_old = vol_3(:, i_3);

end


figure(4)
plot(1:step_1, vol_3(7,:), 'r')
hold on
plot(1:step_1, vol_3(1,:), 'b')
title('Vout plot using bigger value of Cout')
xlabel('Time in millseconds')
ylabel('Voltage in volts')
grid on

%Now we will plot Vout using the value of Cout given
for i_4 = 1:step_1

    Guassian_F(1,1) = exp(-1/2*((i_4/step_1-0.06)/(0.03))^2);
    Guassian_F(4,1) = 0.001*randn();
    Guassian_F(7,1) = 0.001*randn();

    if i_4 == 1

        vol_3(:,i_4) = (C_Matrix1./dt_1+GO)\(Guassian_F
+C_Matrix1*vol_start/dt_1);

    else

        vol_3(:,i_4) = (C_Matrix1./dt_1+GO)\(Guassian_F
+C_Matrix1*vol_old/dt_1);

    end

    vol_old = vol_3(:, i_4);

end


figure(5)
plot(1:step_1, vol_3(7,:), 'r')
hold on
```

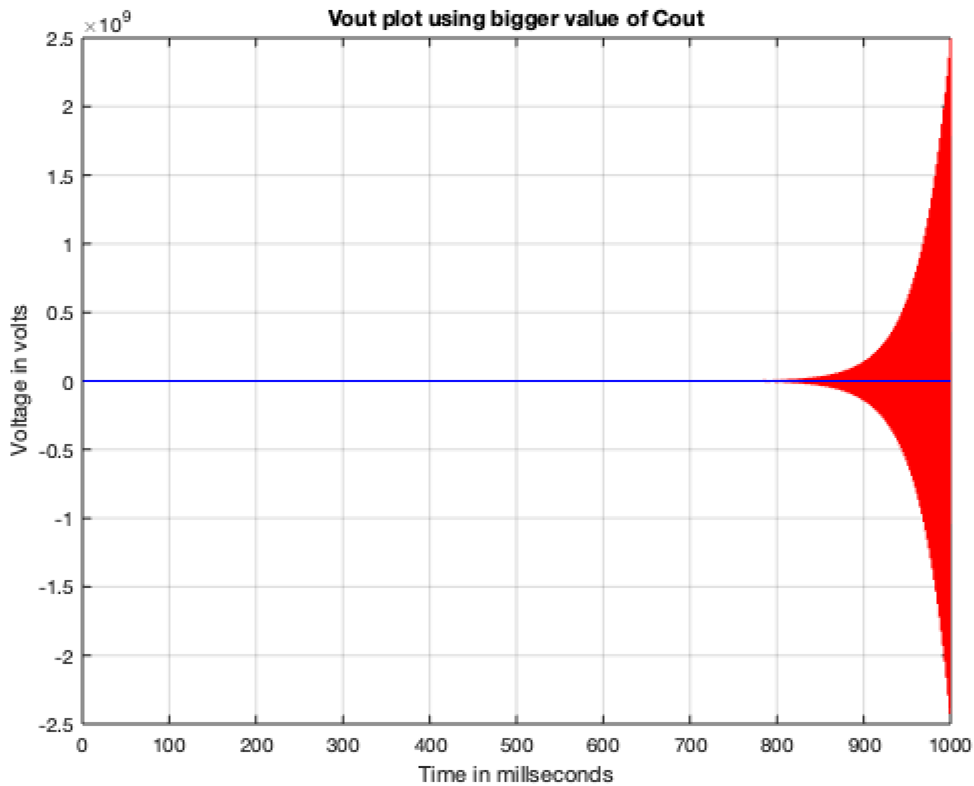
```
plot(1:step_1, vol_3(1,:), 'b')
title('Vout plot using original of Cout of 0.00001')
xlabel('Time in millseconds')
ylabel('Voltage in volts')
grid on

vol_7 = zeros(7, step_1);
Guassian_F = zeros(7,1);

%In part f), we are observing the effects of plotting Vout with
 differet
%time steps. A discussion on my findings will take place at the end of
 this
%document.

%Note that the original timestep was used in part b), and will not be
%replotted to avoid redundancy.

%using the timestep given

for i_5 = 1:step_1

    Guassian_F(1,1) = exp(-1/2*((i_5/step_1-0.06)/(0.03))^2);
    Guassian_F(4,1) = 0.001*randn();
    Guassian_F(7,1) = 0.001*randn();

    if i_5 == 1
        vol_7(:,i_5) = (C_Matrix1./dt_1+GO)\(Guassian_F
+C_Matrix1*vol_start/dt_1);

    else
        vol_7(:,i_5) = (C_Matrix1./dt_1+GO)\(Guassian_F
+C_Matrix1*vol_old/dt_1);

    end

    vol_old = vol_7(:, i_5);

end

figure(6)
plot(1:step_1, vol_7(7,:), 'r')
hold on
plot(1:step_1, vol_7(1,:), 'b')
title('Vout plot using original timestep of 10^-3')
xlabel('Time in picoseconds')
ylabel('Voltage in volts')
grid on

%using a smaller timestep

for i_6 = 1:step_2

    Guassian_F(1,1) = exp(-1/2*((i_6/step_2-0.06)/(0.03))^2);
```

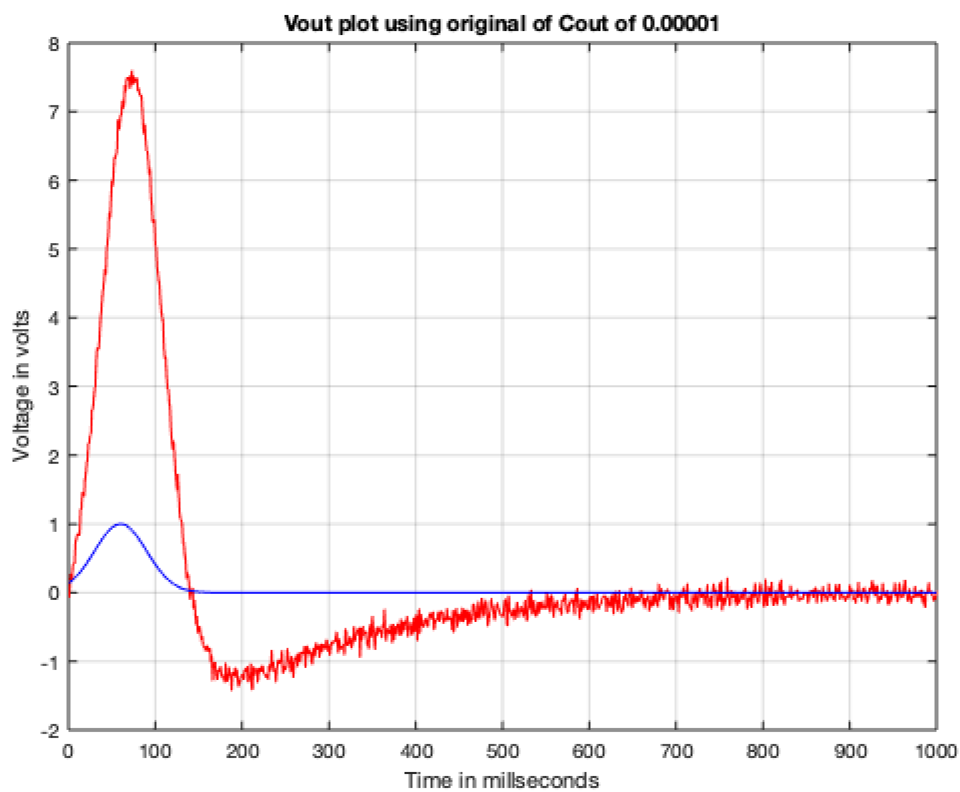
```
    Guassian_F(4,1) = 0.001*randn();
    Guassian_F(7,1) = 0.001*randn();

    if i_6 == 1
        vol_4(:,i_6) = (C_Matrix1./dt_2+GO)\(Guassian_F
+C_Matrix1*vol_start/dt_2);

    else
        vol_4(:,i_6) = (C_Matrix1./dt_2+GO)\(Guassian_F
+C_Matrix1*vol_old/dt_2);

    end

    vol_old = vol_4(:, i_6);

end


figure(7)
plot(1:step_2, vol_4(7,:), 'r')
hold on
plot(1:step_2, vol_4(1,:), 'b')
title('Vout plot using a smaller timestep')
xlabel('Time in picoseconds')
ylabel('Voltage in volts')
grid on
```

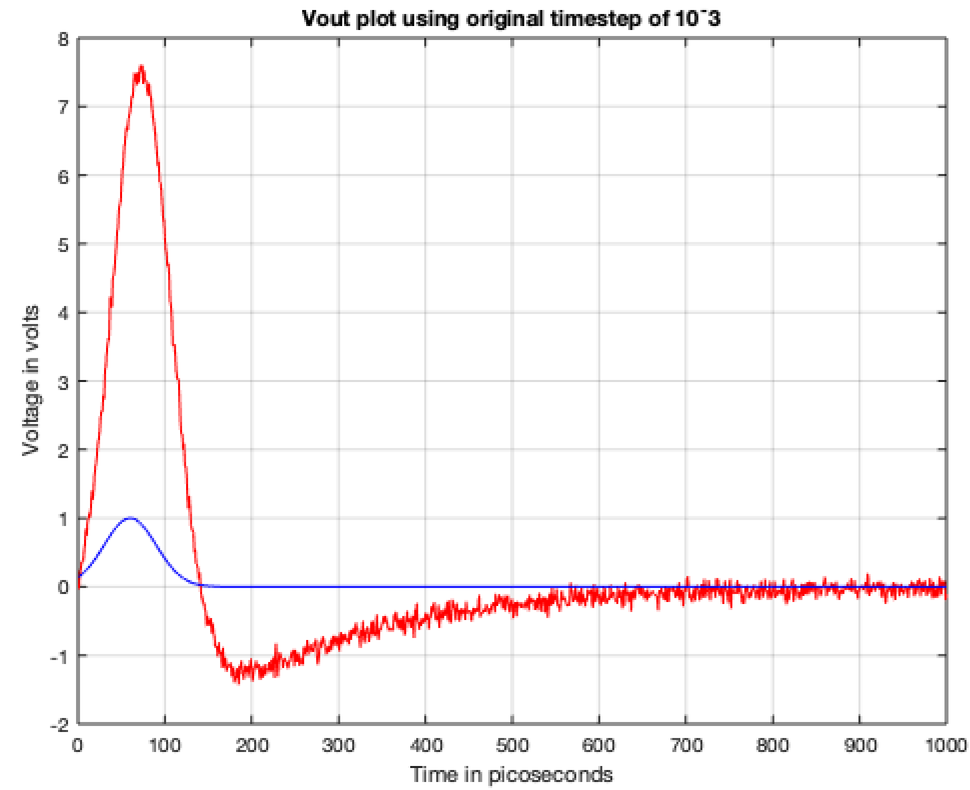


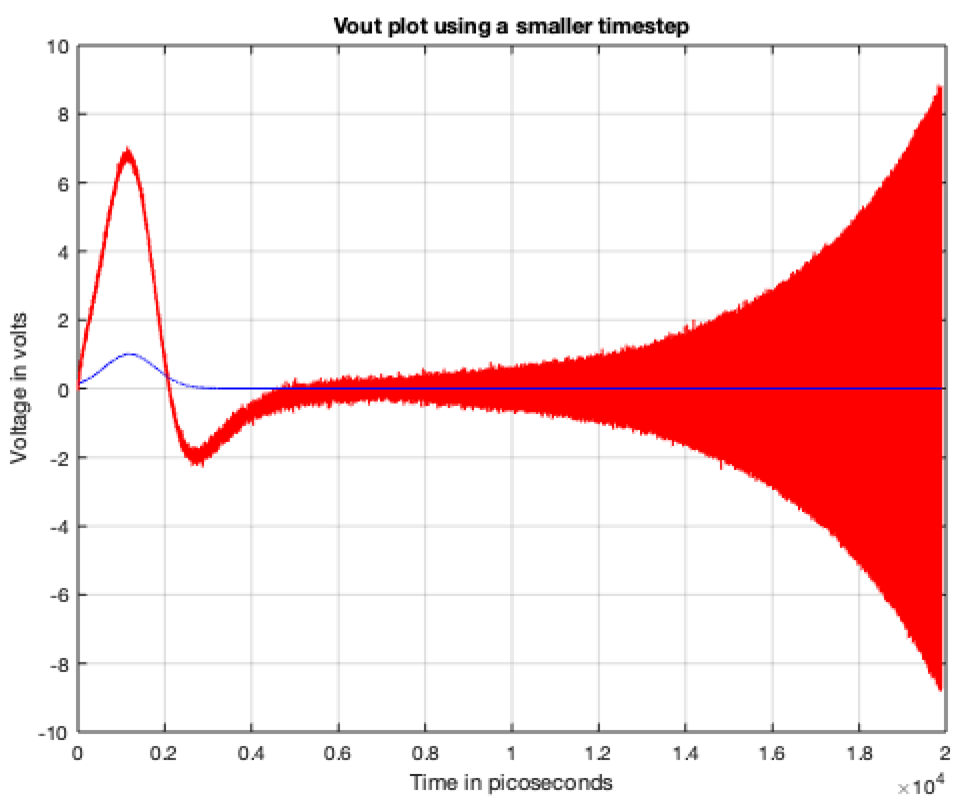

Fourier-Transform Plot of Vout
Voltage in volts
frequency in 1/ms
70
60
50
40
30
20
10
0
-500
-400
-300
-200
-100
0
100
200
300
400
500


Vout plot using smaller value of Cout
Voltage in volts
Time im milliseconds)
8
7
6
5
4
3
2
1
0
-1
-2
0
100
200
300
400
500
600
700
800
900
1000


Vout plot using bigger value of Cout
×10^9
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
Voltage in volts
0
100
200
300
400
500
600
700
800
900
1000
Time in millseconds


Vout plot using original of Cout of 0.00001
8
7
6
5
4
3
2
1
0
-1
-2
Voltage in volts
0
100
200
300
400
500
600
700
800
900
1000
Time in millseconds


Vout plot using original timestep of 10^3
Voltage in volts
Time in picoseconds
8
7
6
5
4
3
2
1
0
-1
-2
0
100
200
300
400
500
600
700
800
900
1000


Vout plot using a smaller timestep
Voltage in volts
Time in picoseconds
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
×10^4

Discussion

```
% In part e), 3 plots of Vout were made using 3 values of Cout. It is
% noticed that using a smaller value of Cout does not change the plot.
% However, a larger value of Cout causes the simulation to break down.
 This
% is because the circuit becomes trapped in a feedback loop.

% In part f), 2 plots of Vout were made using different timesteps.
 Using a
% smaller timestep than the original timestep of 10^-3 causes the
% simulation to break down because the circuit becomes trapped in a
% feedback loop.
```

PART 4

```
% If the e voltage source on the output stage described by the
 transconductance equation
% V = ?I3 was instead modeled by V = alpha*I3 +beta*I2^2 + gamma*I3^3,
 the changes
% required in my simulator are more than just simply changing the
 matrices
% we used in the simulation for part 3. This new equation would need
 to be
% fitted, And new matrices would have to be created for the Jacobian
 method
% for the simulation of the new equation in the circuit. Note that the
% inclusion of this equation would in turn increase the size of the
 matrix
% and the iterations required to traverse through it. The values of
 alpha
% and beta that we define will have to be considerably large, as if i3
 is smaller
% than 1 this will be too small to have a noticeable effect on the
% simulation.
```

Published with MATLAB® R2019b