ELEC 4700

ASSIGNMENT 3 – MONTE CARLO/ FINITE DIFFERENCE METHOD

Submitted By:

Jarikre Efe Jeffery

Date Submitted:

05/04/2020

Introduction:

Part 1:

The differential equations that represent the network in the time domain using KCL (Summation of I=0 at the node) can be found below:

Equations:

$$
\begin{aligned}
V_1 &= V_{in} \\
G_1(V_2 - V_1) + C\frac{d(V_2 - V_1)}{dt} + G_2V_2 - I_L &= 0 \\
V_2 - V_3 - L\frac{dI_L}{dt} &= 0 \\
-I_L + G_3V_3 &= 0 \\
V_4 - \alpha I_3 &= 0 \\
G_3V_3 - I_3 &= 0 \\
G_4(V_O - V_4) + G_OV_O &= 0
\end{aligned}
$$

Matrices:

$$
\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -C & C & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -L & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad
\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -G_1 & G_1 + G_2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & G_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\alpha & 1 & 0 \\ 0 & 0 & 0 & G_3 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -G_4 & G_4 + G_O \end{bmatrix}
$$

$$
\mathbf{V} = \begin{bmatrix} V_1 \\ V_2 \\ I_L \\ V_3 \\ I_3 \\ V_4 \\ V_O \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} V_{in} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
$$

Equations in the frequency domain can be found below:

In the frequency domain: Date

$V_1 = V_{in}$

$G_1 (V_2 - V_1) + C j\omega (V_2 - V_1) + G_2 V_2 - I_L = 0$

$V_2 - V_1 - L j\omega I_L = 0$

$-I_L + G_3 V_3 = 0$

$V_4 - \alpha I_3 = 0$

$G_3 V_3 - I_3 = 0$

$G_4 (V_o - V_4) + G_o V_o = 0$

Programming section:

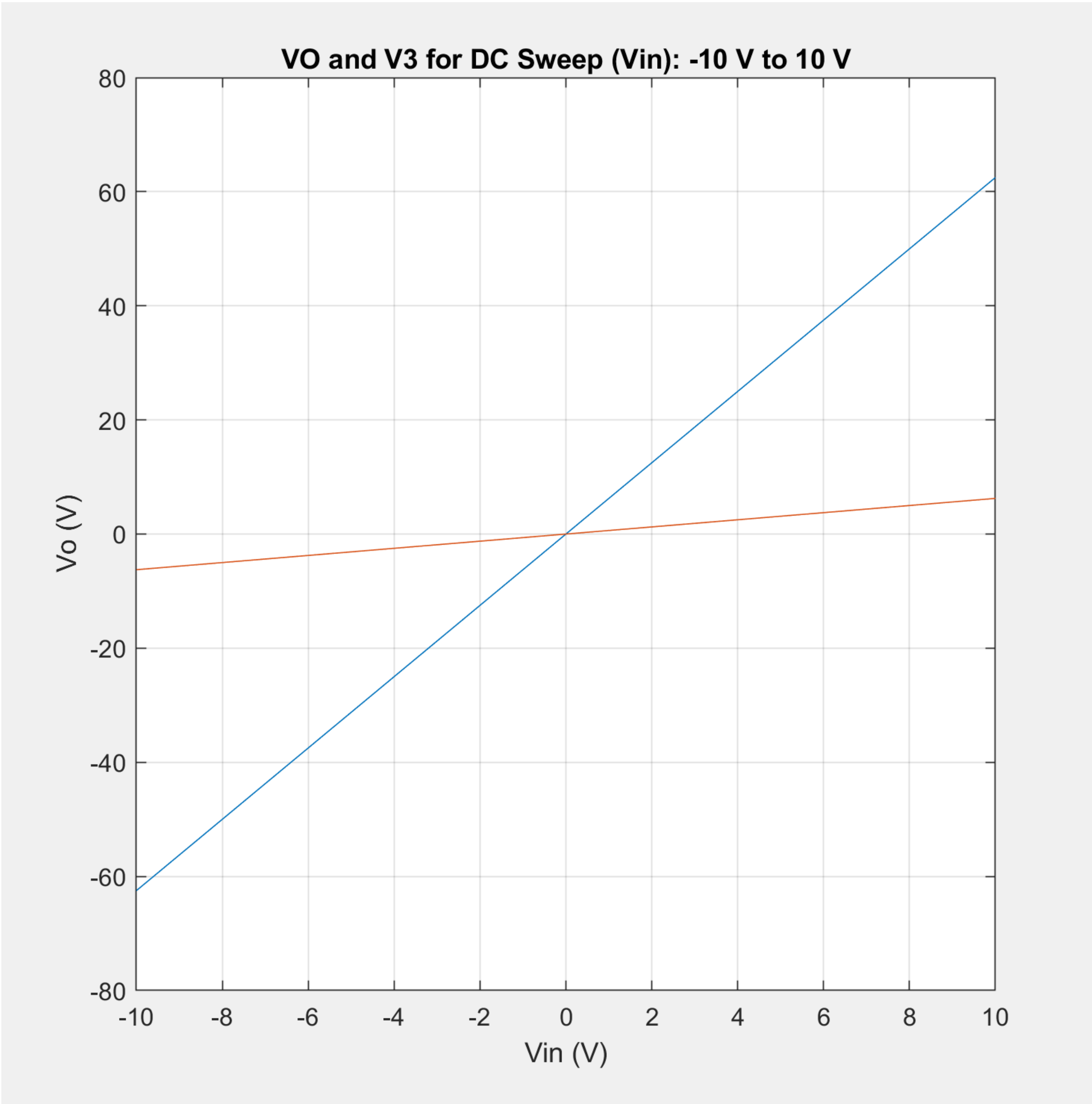

Figure 1: DC Sweep of the input voltage

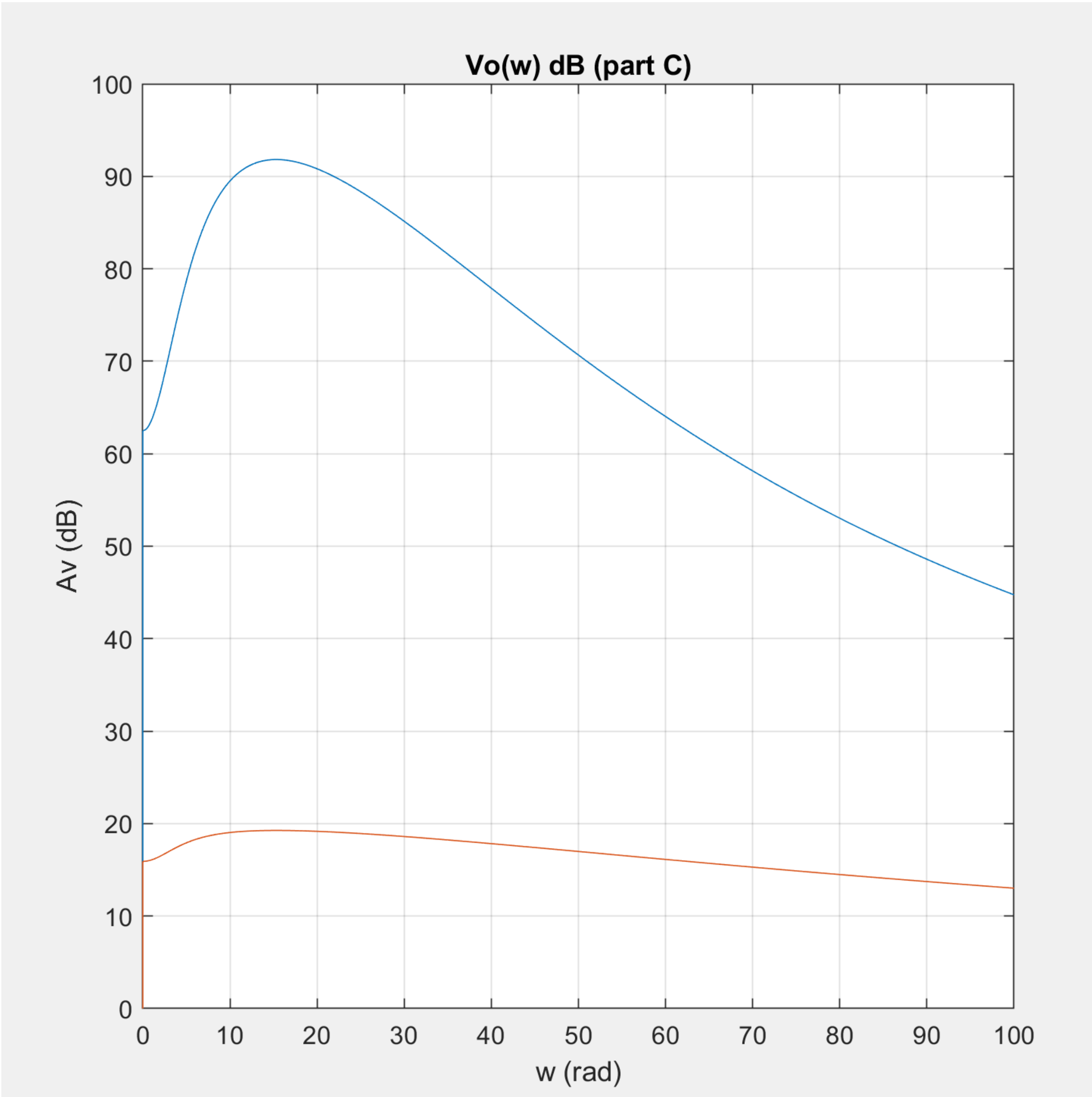

Figure 2: Ac plot of Vo as a fuction of w

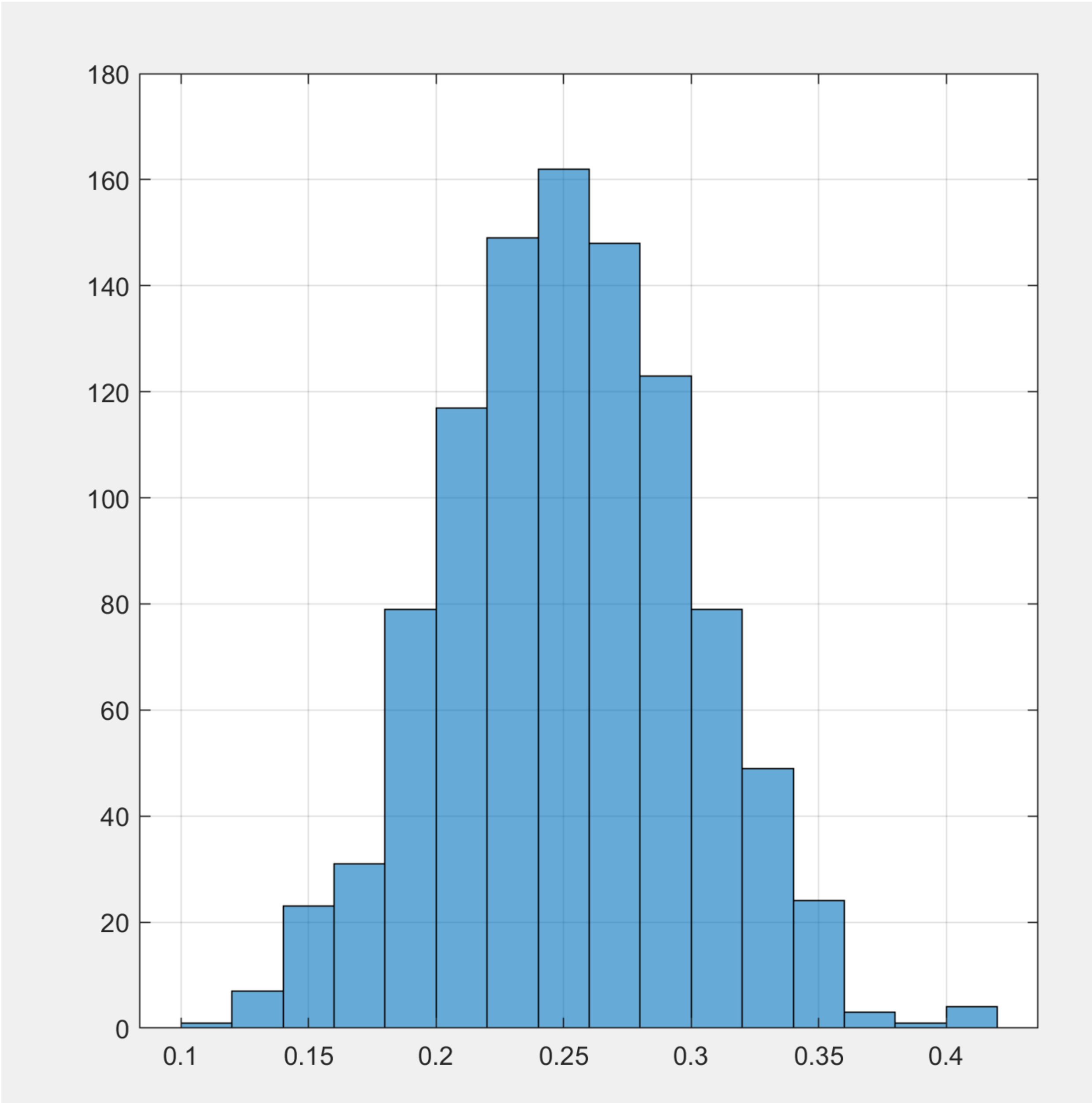

Figure 3: Histogram plot 1

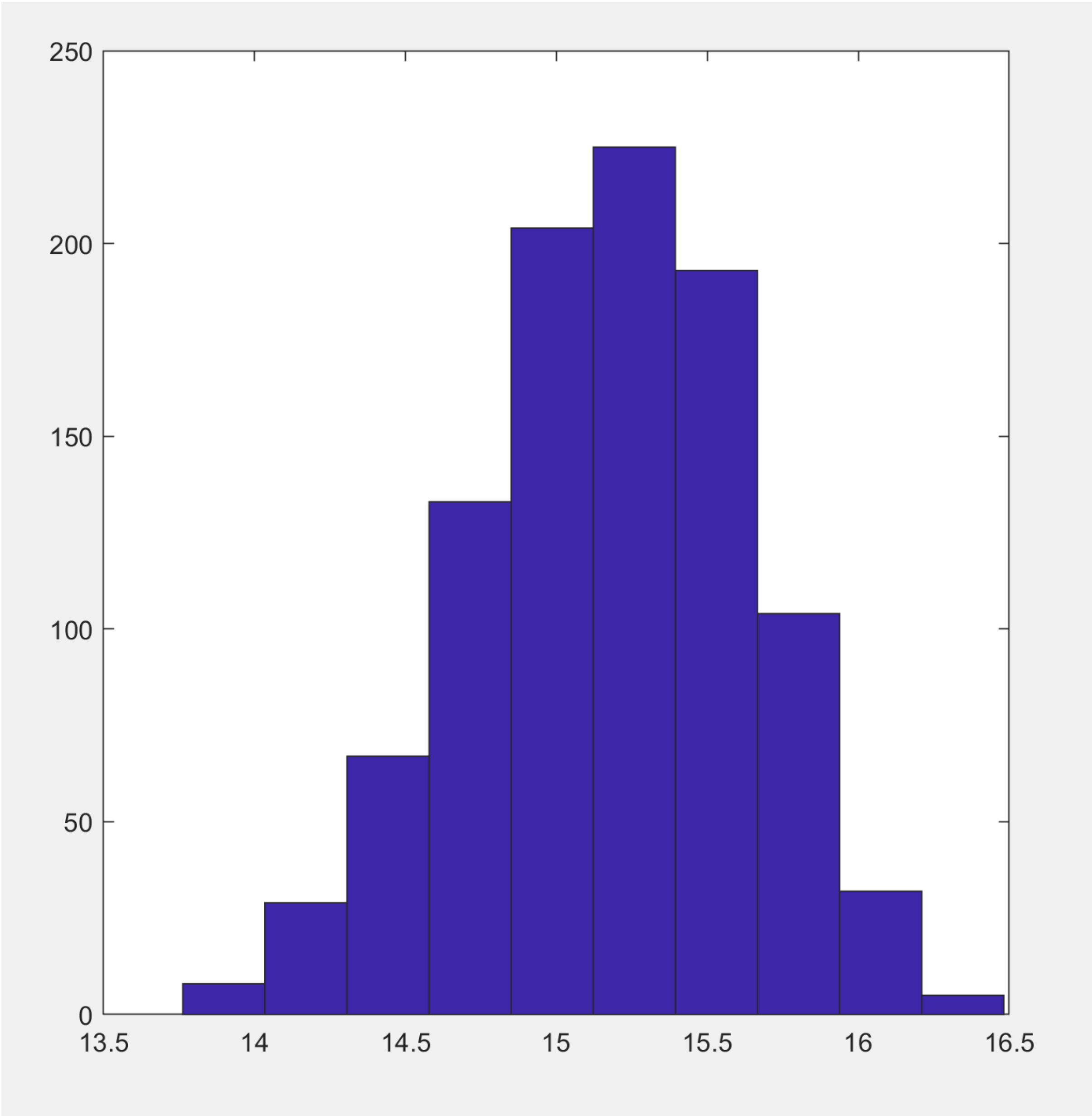

Figure 4: Histogram plot 2

Transient Simulation:

The circuit can be simulated in the time domain by solving the equation of $C dV/dt + GV = F$

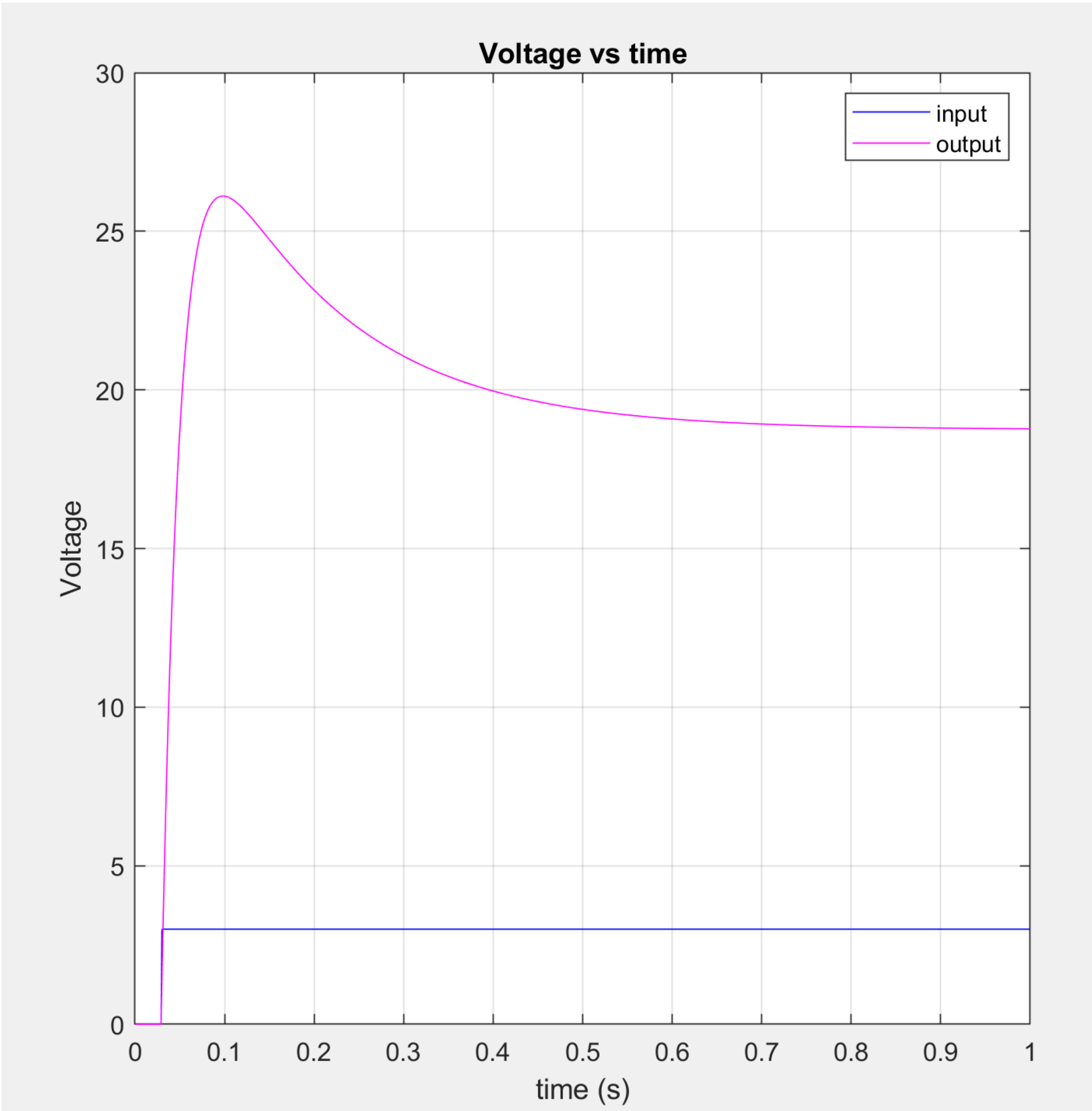

Figure 5: Plot of the voltage vs time

a.) By inspection the circuit is a low pass filter
b.) We should expect the frequency response to cut off high frequencies and allow only the low frequencies to go through.
d.) Part v: As the time step is increased the accuracy of the simulation is decreased.

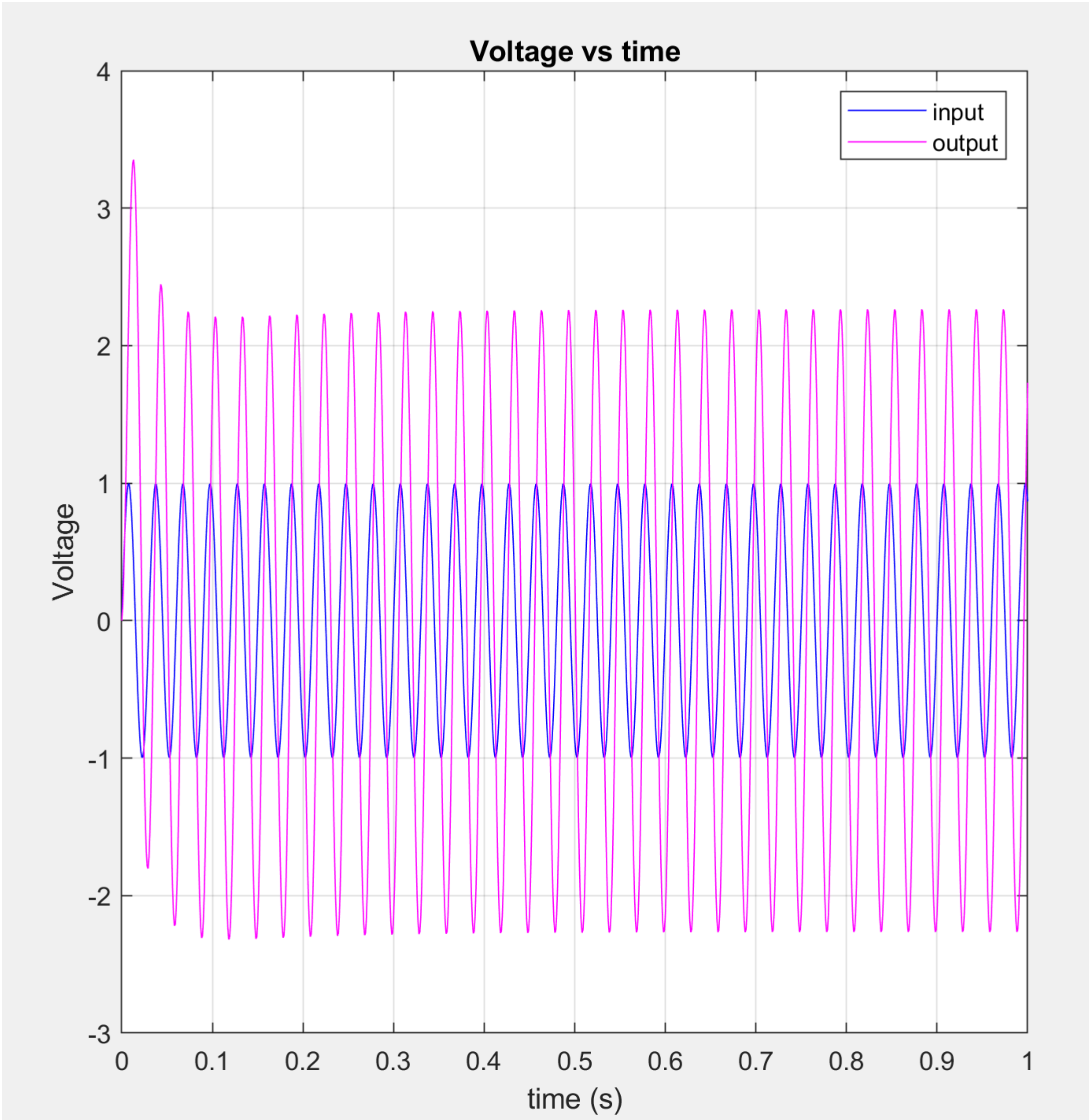

Figure 6: voltage vs time input and output sinusoid

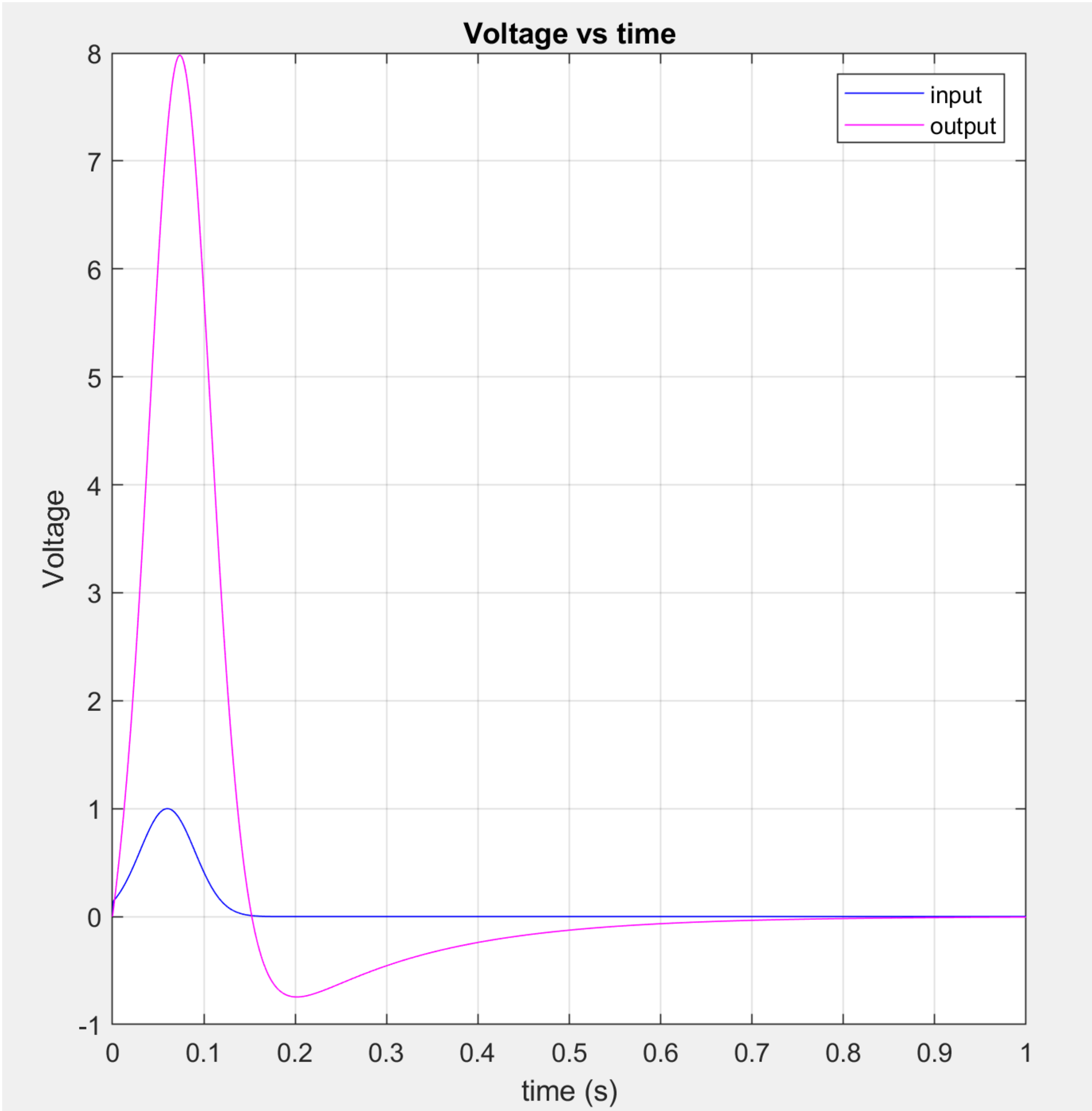

Figure 7: voltage vs time with input and output

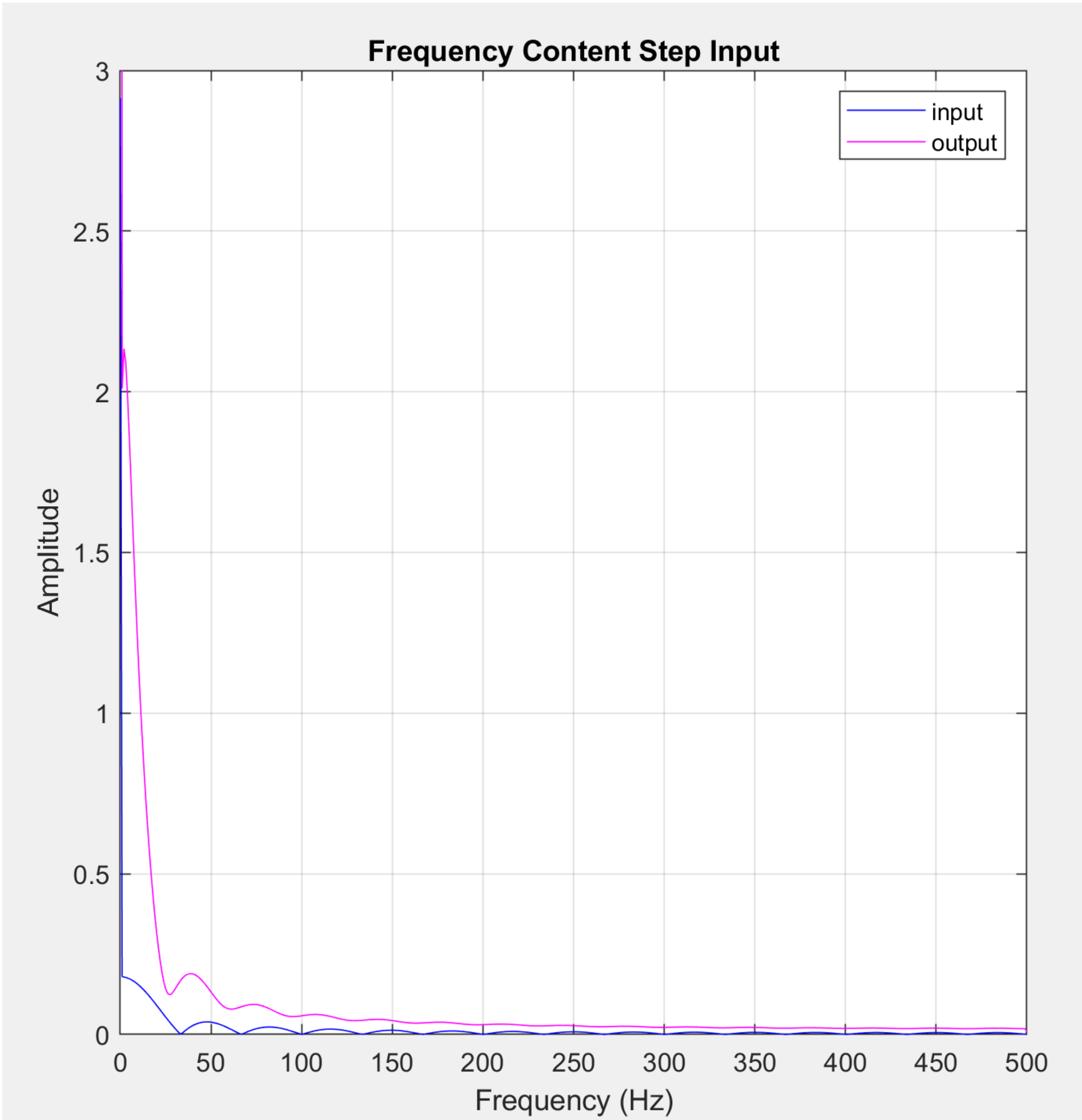

Figure 8: frequency content step input

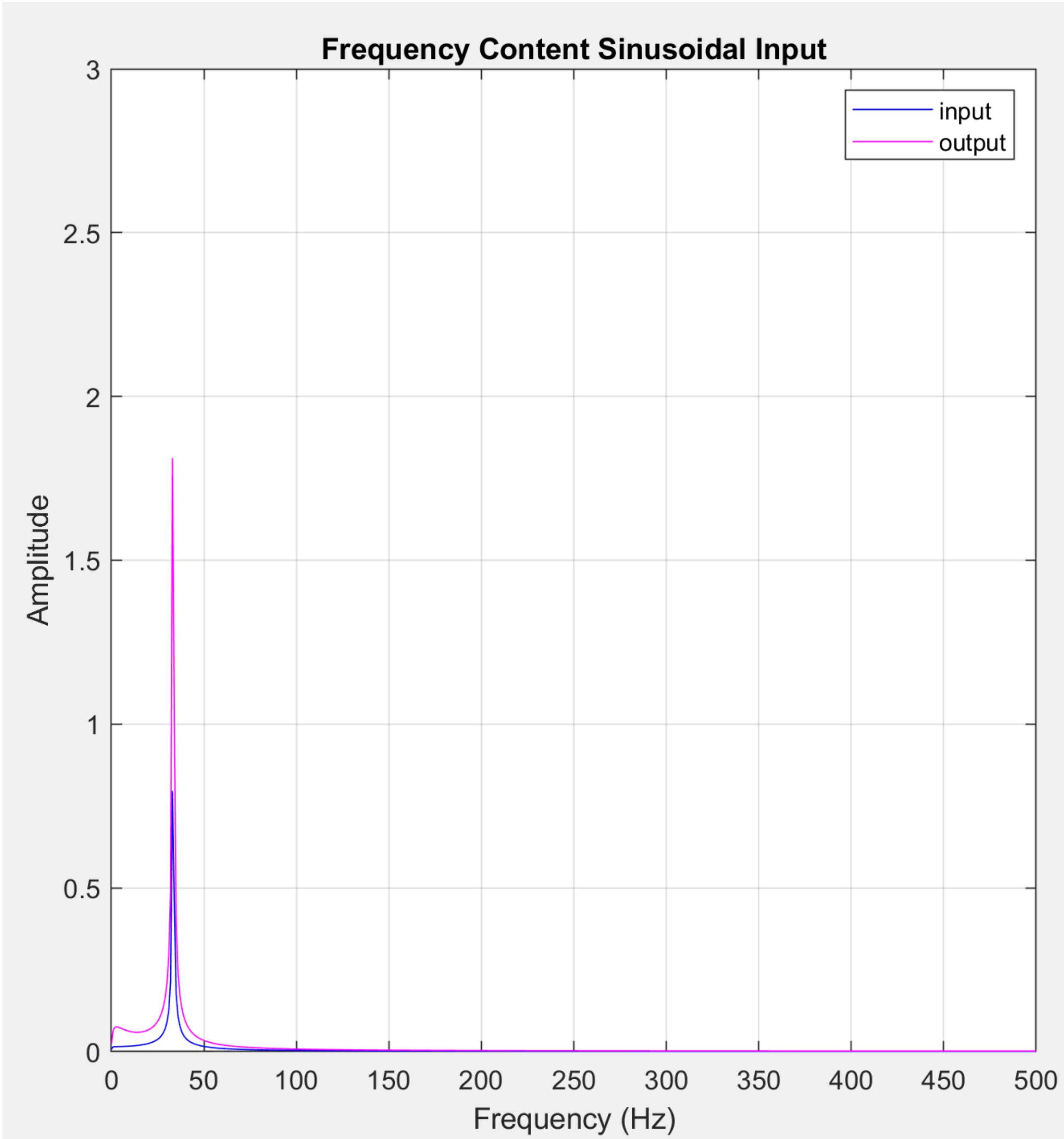

Figure 9: Frequency content sinuisoidal input

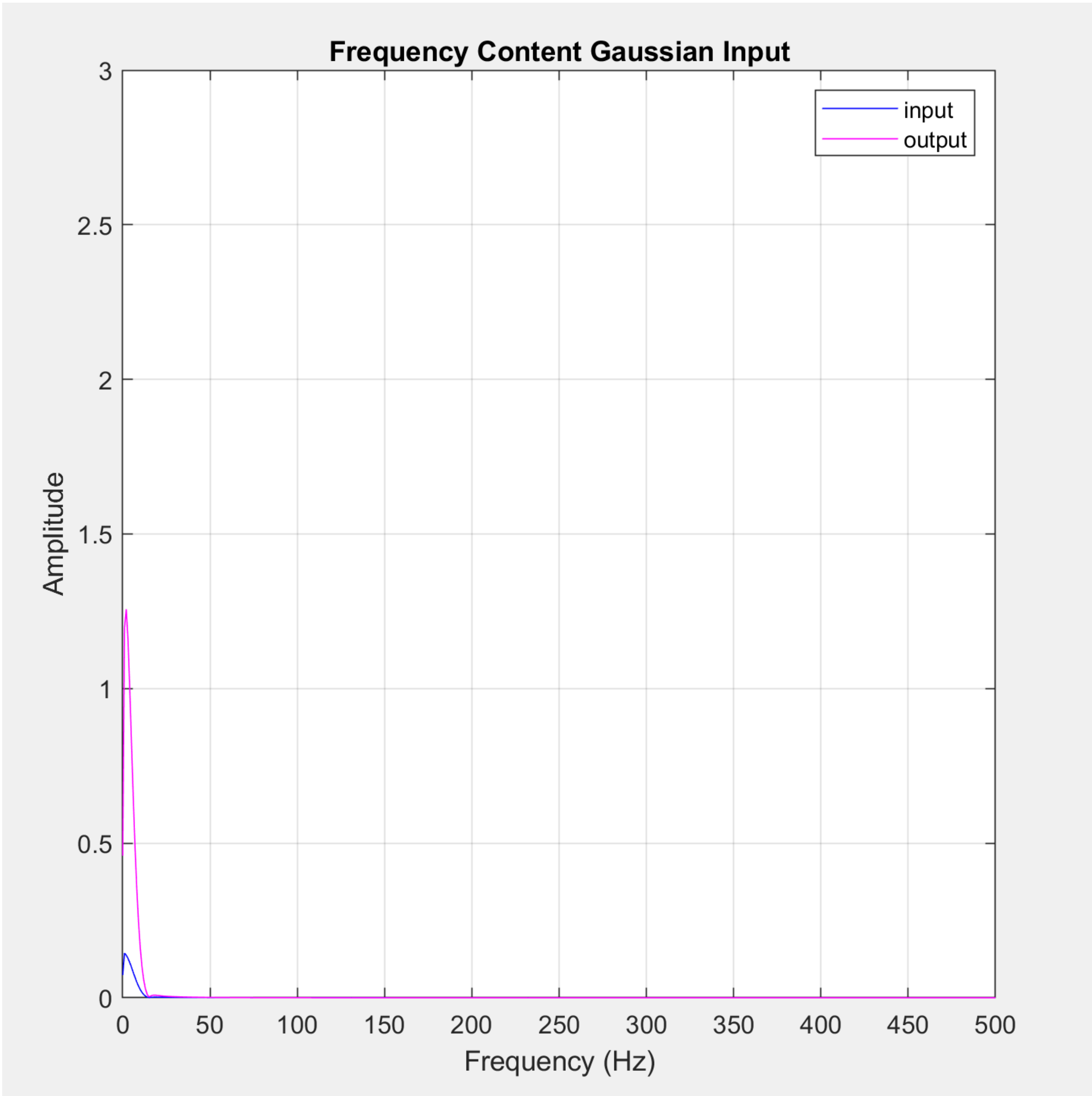

Figure 10: Frequency content Gaussian input

C matrix:

```
C1 =

    0.2500   -0.2500         0         0         0         0         0         0
   -0.2500    0.2500         0         0         0         0         0         0
         0         0    0.0000         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   -0.2000         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
```

Note That Cn1 = 0.00001

```
C1(1,:)=[C -C 0 0 0 0 0 0];
C1(2,:)=[-C C 0 0 0 0 0 0];
C1(3,:)=[0 0 cn1 0 0 0 0 0];
C1(4,:)=[0 0 0 0 0 0 0 0];
C1(5,:)=[0 0 0 0 0 0 0 0];
C1(6,:)=[0 0 0 0 0 -L 0 0];
C1(7,:)=[0 0 0 0 0 0 0 0];
C1(8,:)=[0 0 0 0 0 0 0 0];
```

Varying Cn:

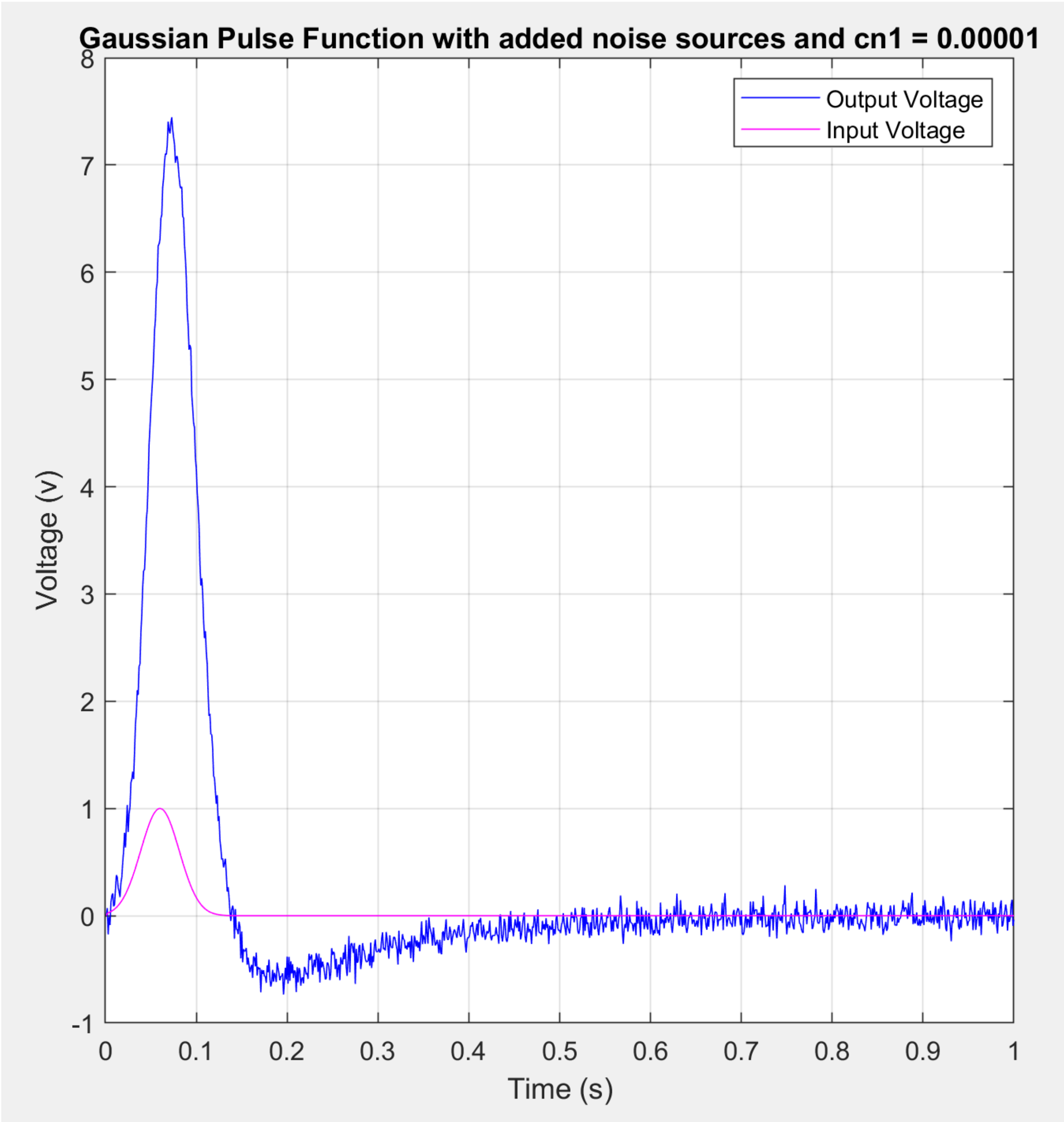

Figure 11: Gaussian Pulse function with added noise sources and cn1 = 0.00001

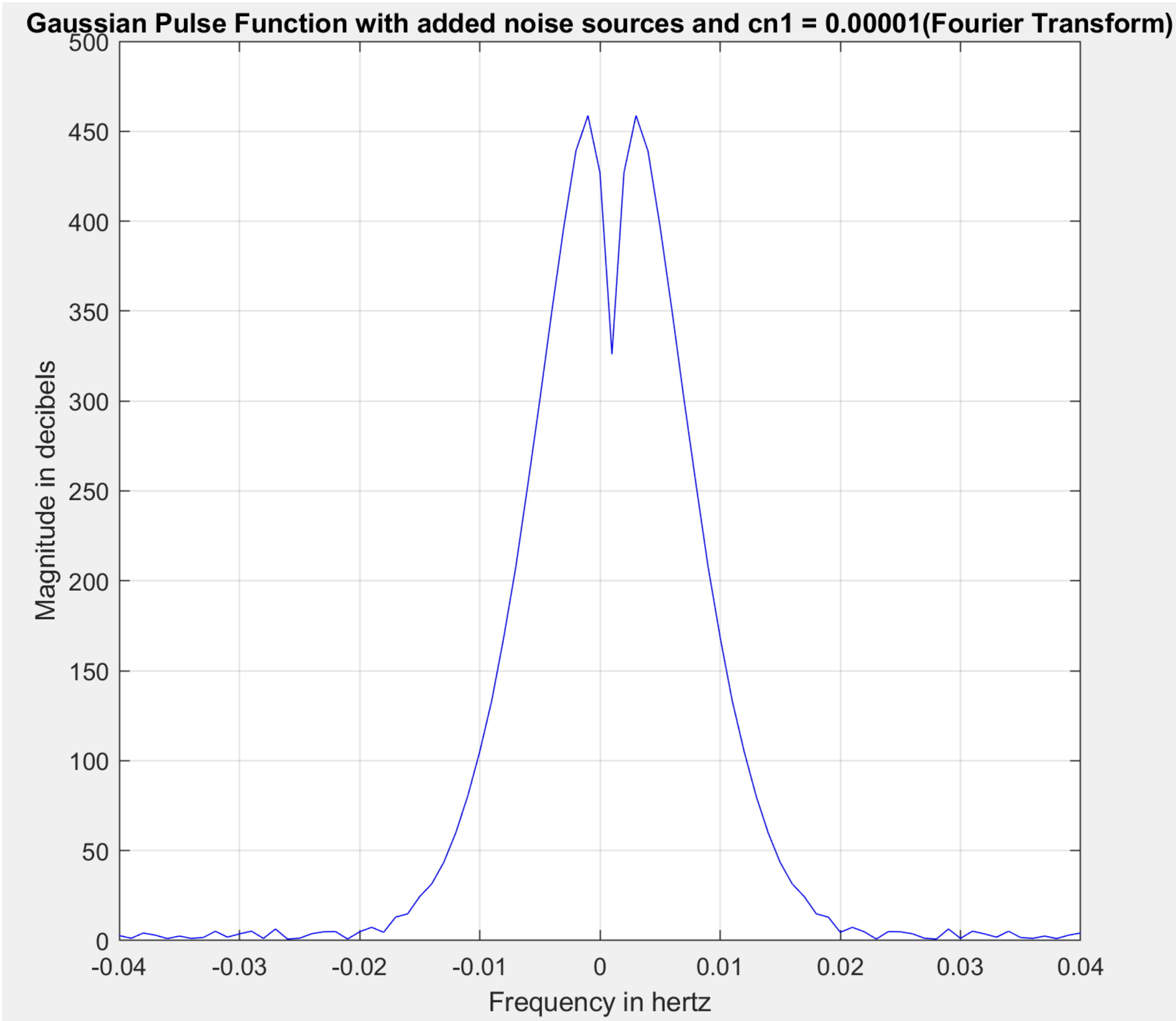

Figure 12: Gaussian Pulse function with added noise sources and cn1=0.00001 fourier transform

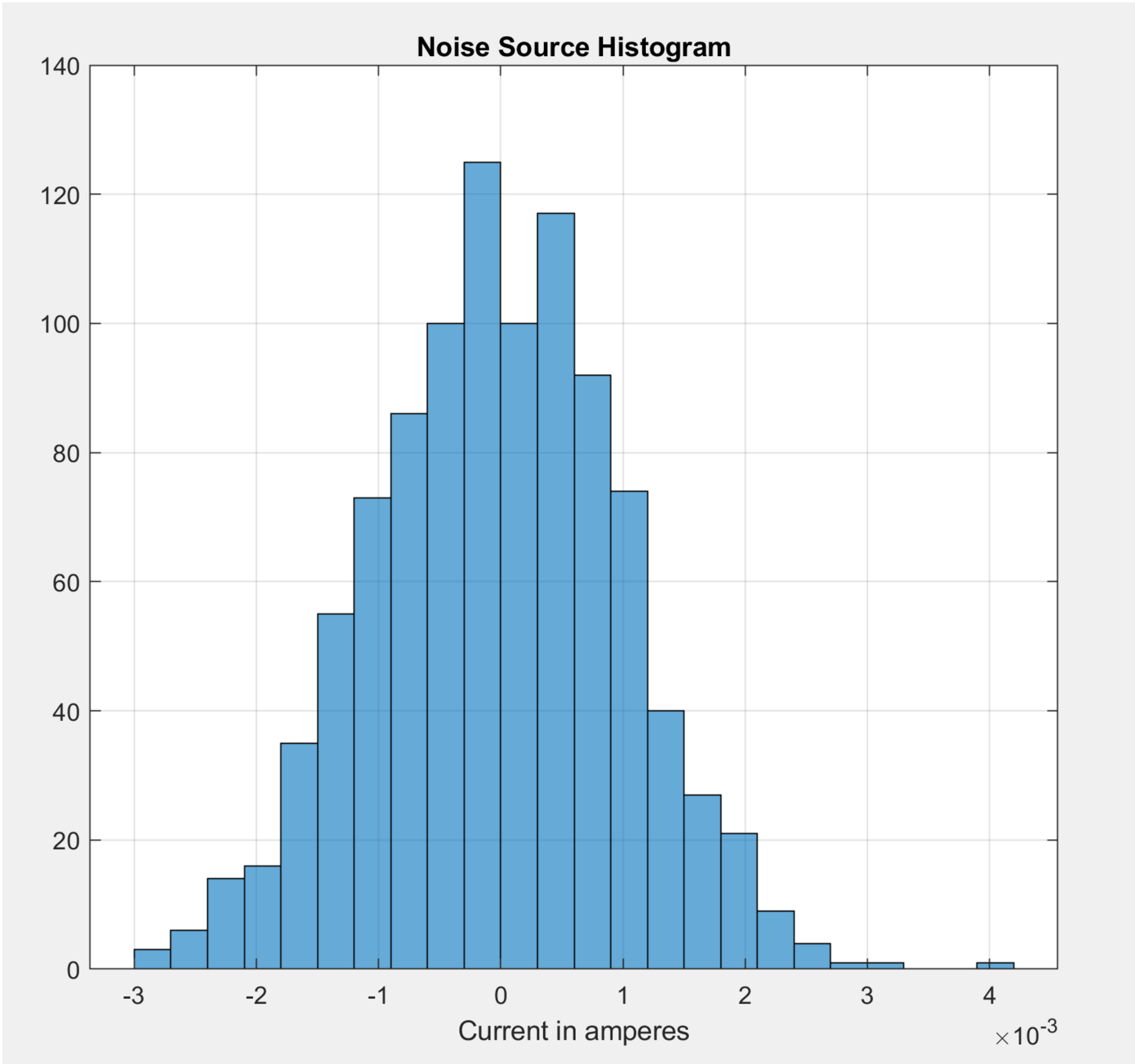

Figure 13: Noise Source Histogram

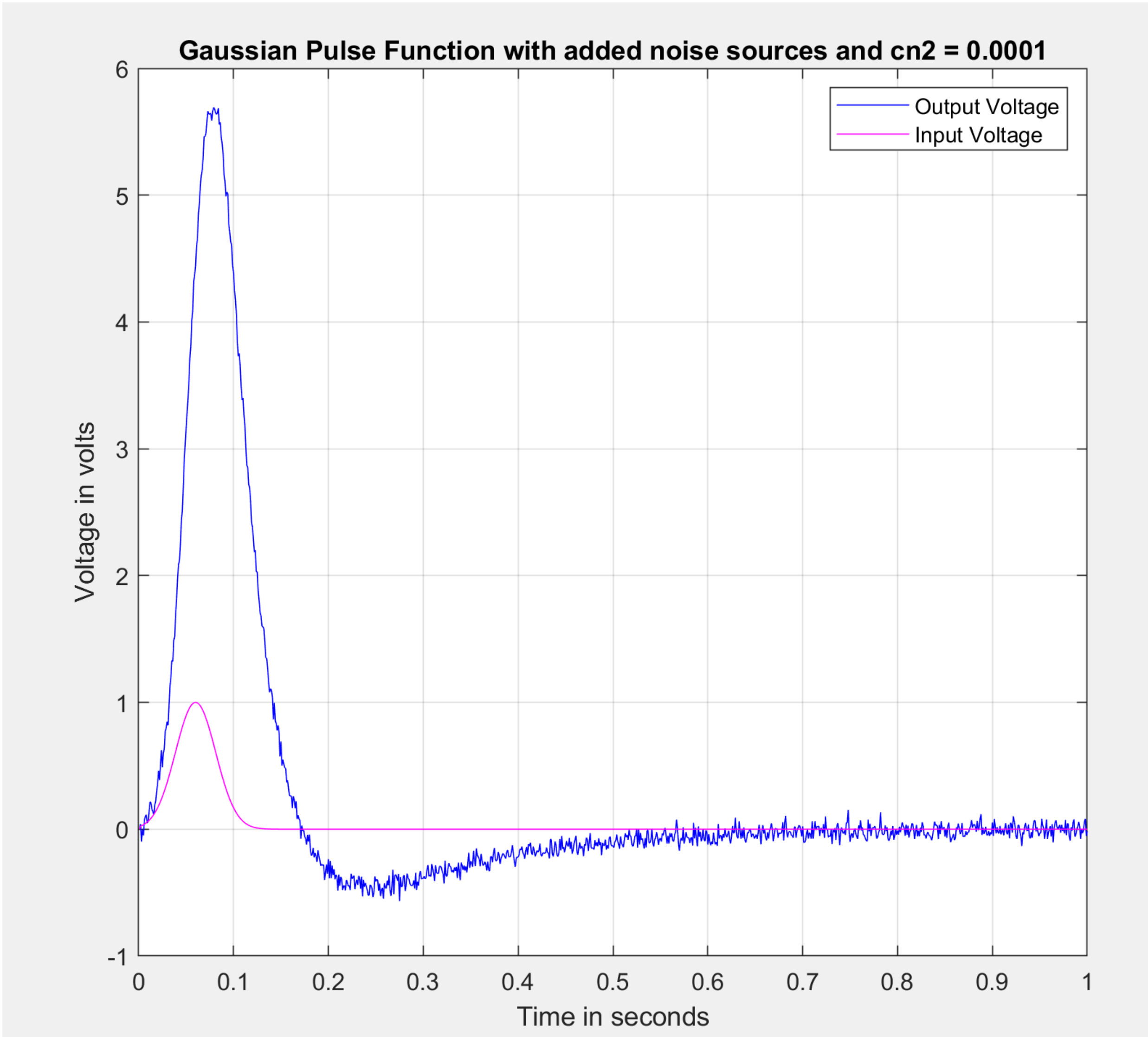

Figure 14: Gaussian Pulse function with added noise sources and cn2=0.0001

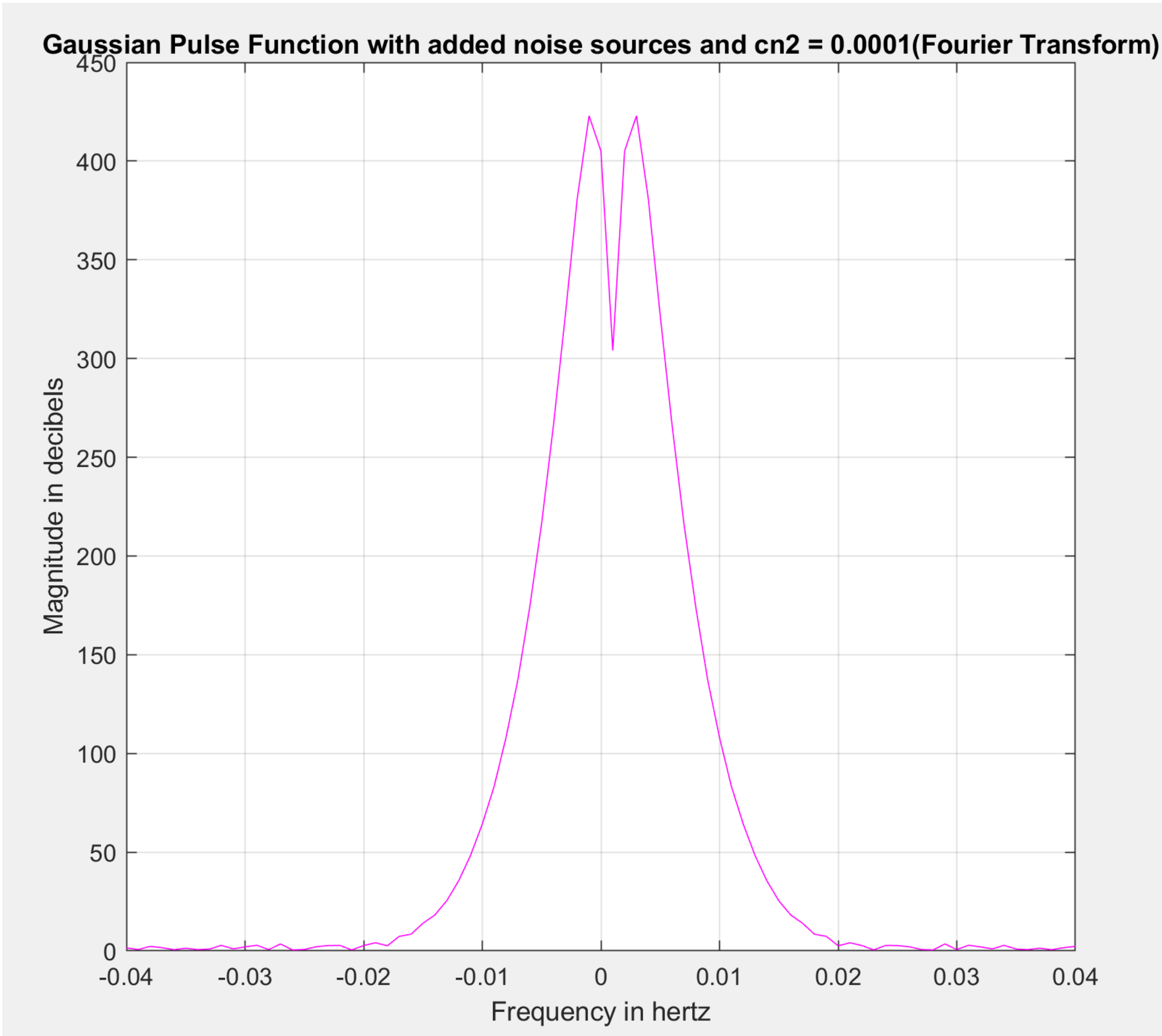

Figure 15: Gaussian Pulse Function with added noise sources and cn2 =0.0001 fourier transform

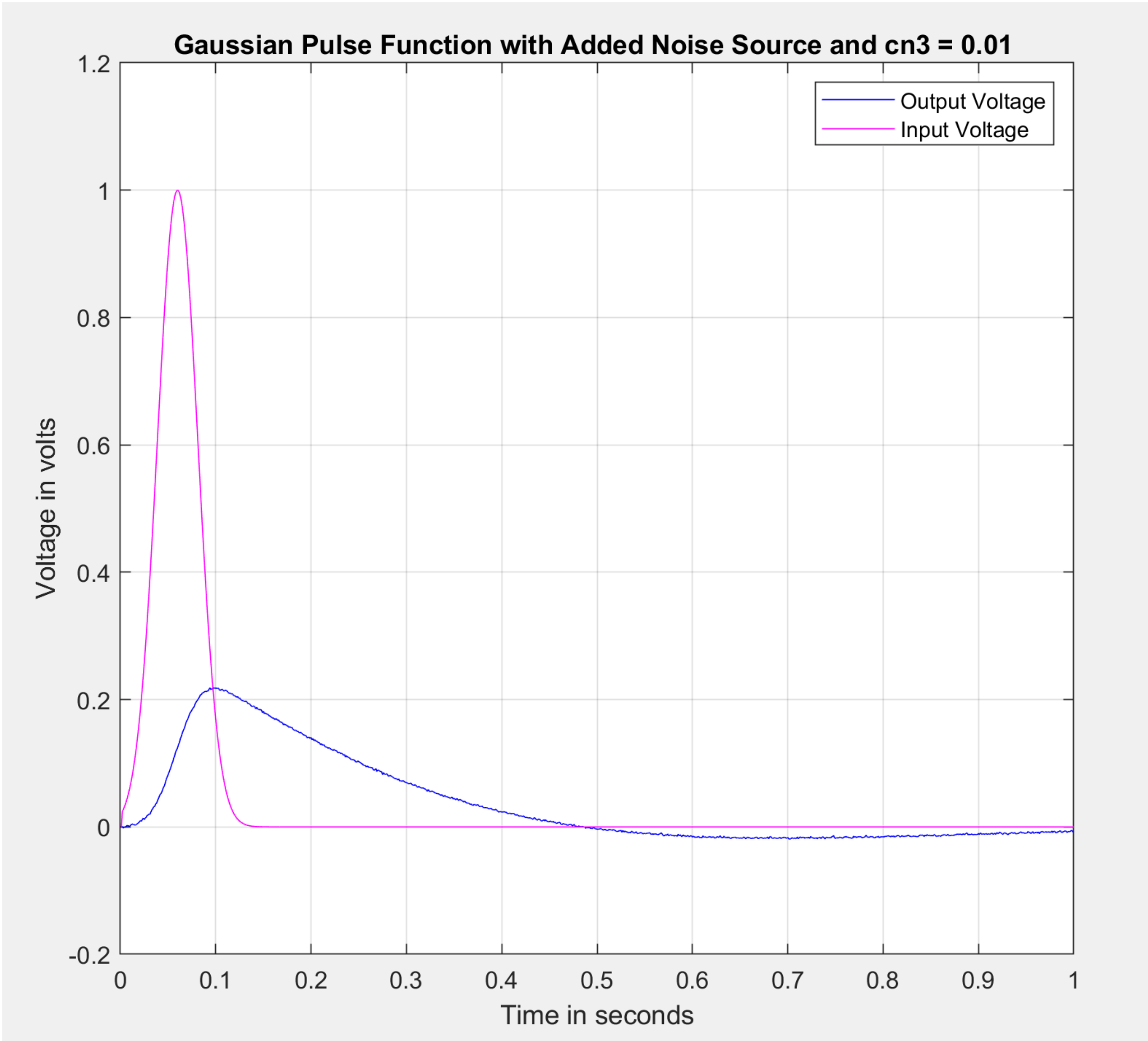

Figure 16: Gaussian Pulse function with added noise sources and cn3=0.01

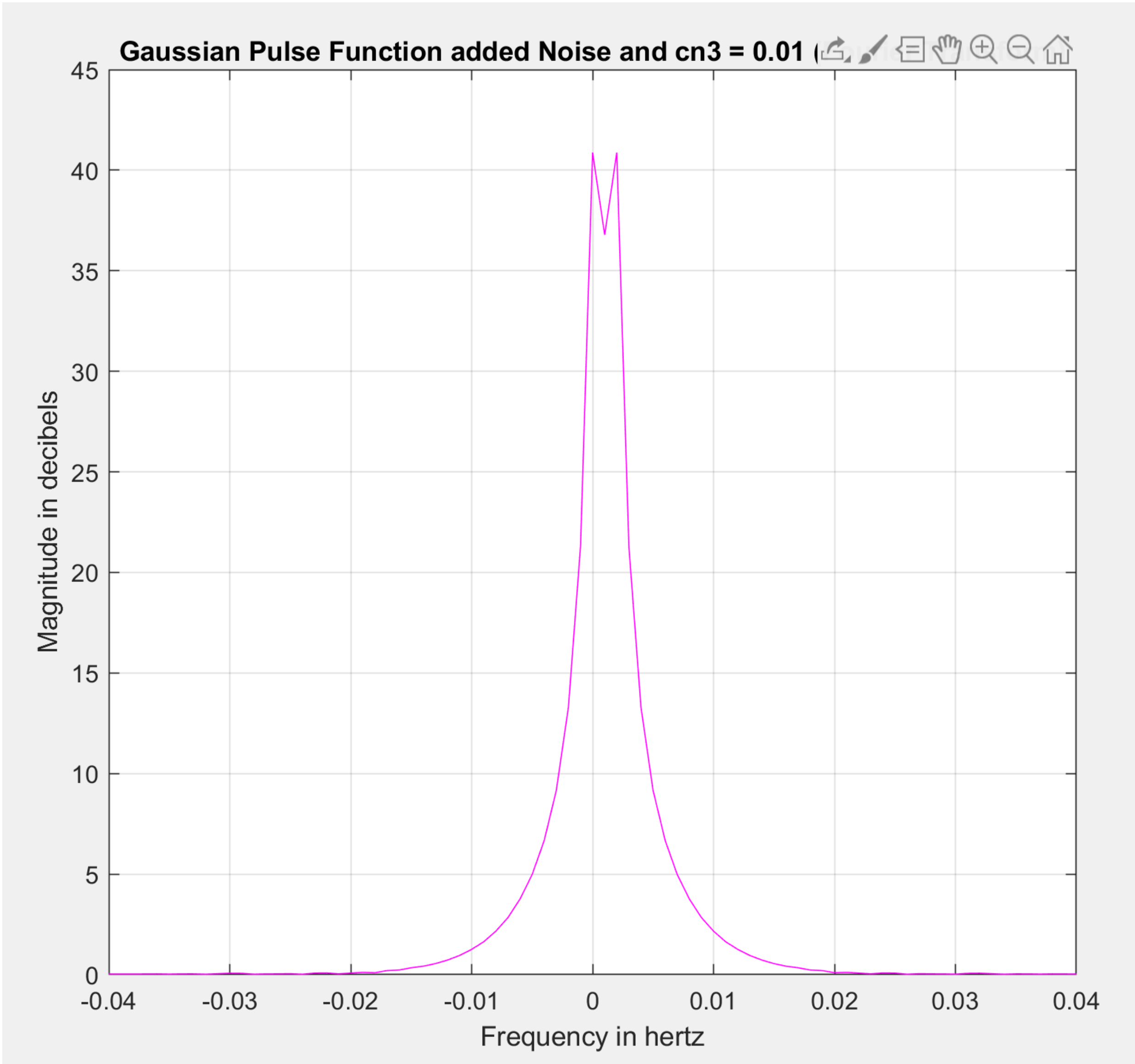

Figure 17: Gaussian Pulse function with added noise sources and cn3=0.01 fourier transform

We can conclude that as cn gets larger the bandwidth tends to get smaller.

Varying the timestep:

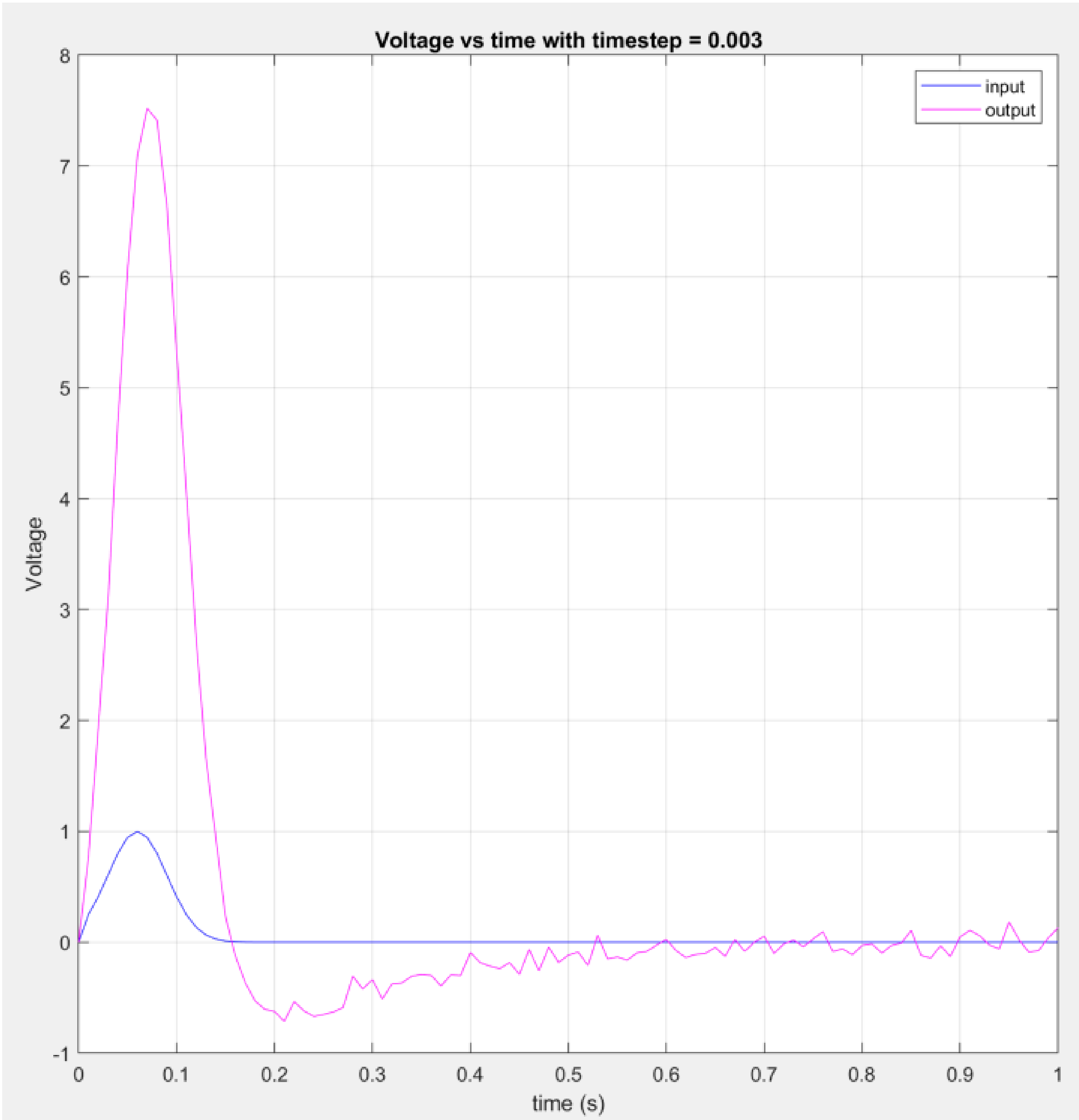

Figure 18: Plot when time step equals 0.003

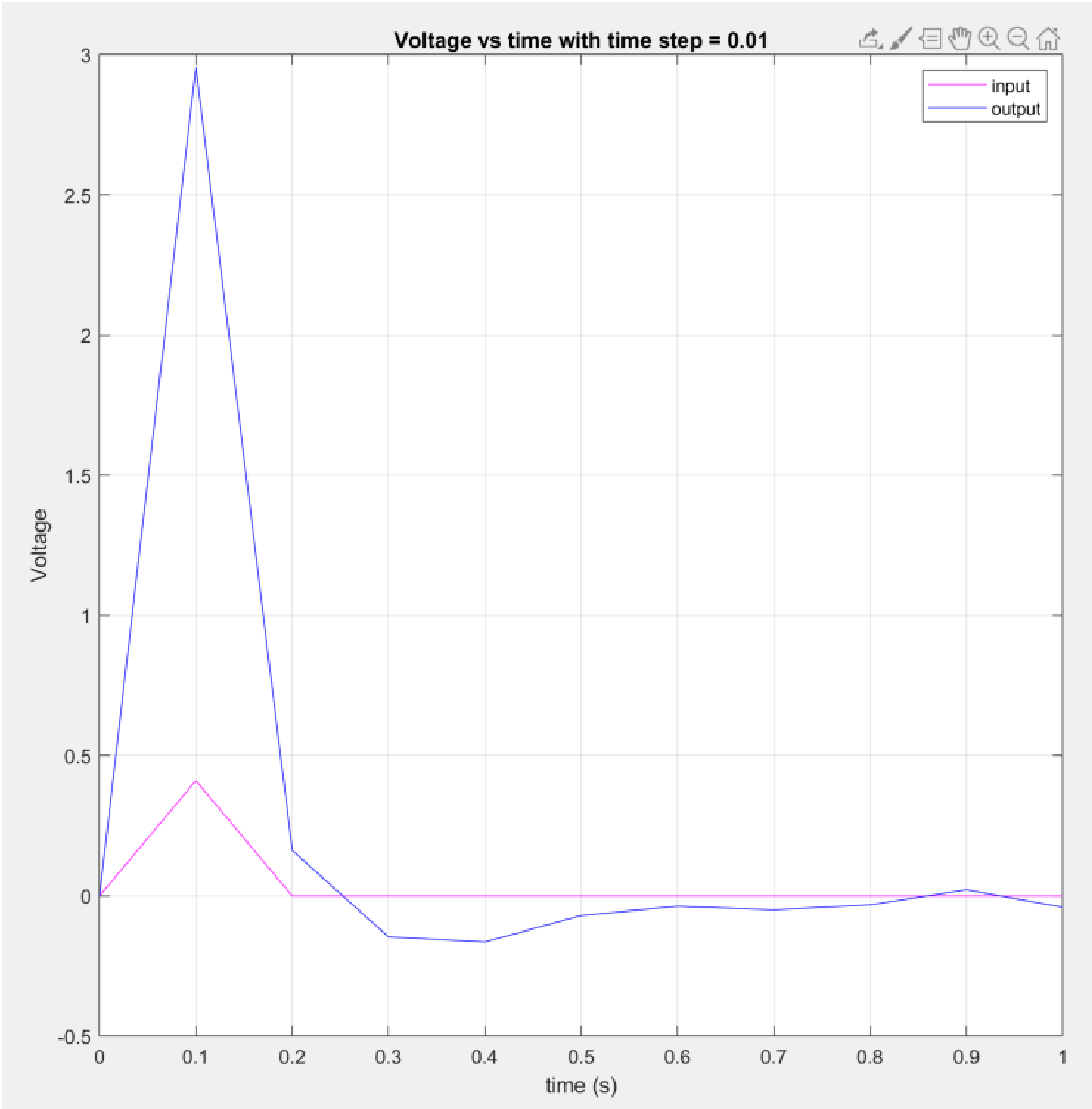

Figure19: Plot when timestep equals 0.01

What I seemed to have observed was that as the time step is increased the accuracy of the simulation decreases.

Non-Linearity:

In order to implement this in MATLAB we would need to add another matrix to represent or contain the equations for the non-linear elements. A column vector B(V) would need to be added. When that has been added, because the system is now non-linear it can't be solved by simple gaussian elimination and instead the newton Raphson numerical method would need to be used instead.

Appendix section:

```
clc
clear
clearvars
set(0,'DefaultFigureWindowStyle','docked')
% Name:Jarikre Efe Jeffery
% Student Number: 101008461
% Part 3: Programming

G = zeros(6, 6);

%Resistances:
R1 = 1;
R2 = 2;
R3 = 10;
R4 = 0.1;
R0 = 1000;

%Conductances:
G1 = 1/R1;
G2 = 1/R2;
G3 = 1/R3;
G4 = 1/R4;
G0 = 1/R0;

%Additional Parameters:
a = 100;
Cval = 0.25;
L = 0.2;
vi = zeros(100, 1);
vo = zeros(100, 1);
v3 = zeros(100, 1);

G(1, 1) = 1;
G(2, 1) = G1; G(2, 2) = -(G1 + G2); G(2, 6) = -1;
```

```
G(3 ,3) = -G3; G(3, 6) = 1;
G(4, 3) = -a*G3; G(4, 4) = 1;
G(5, 5) = -(G4+G0); G(5, 4) = G4;
G(6, 2) = -1; G(6, 3) = 1;


C = zeros(6, 6);

C(2, 1) = Cval; C(2, 2) = -Cval;
C(6, 6) = L;

F = zeros(6, 1);
v = 0;

for vin = -10:0.1:10
   v = v + 1;
   F(1) = vin;
   Vm = G\F;
   vi(v) = vin;
   vo(v) = Vm(5);
   v3(v) = Vm(3);
end

figure(1)
plot(vi, vo);
hold on;
plot(vi, v3);
title('VO and V3 for DC Sweep (Vin): -10 V to 10 V');
xlabel('Vin (V)')
ylabel('Vo (V)')
grid on


vo2 = zeros(1000, 1);
W = zeros(1000, 1);

Avlog = zeros(1000, 1);

for freq = linspace(0, 100, 1000)
   v = v+1;
```

```
    Vm2 = (G+1j*freq*C)\F;
    W(v) = freq;
    vo2(v) = norm(Vm2(5));

    Avlog(v) = 20*log10(norm(Vm2(5))/10);
end

figure(3)
plot(W, vo2)
hold on;
plot(W, Avlog)
grid on
title('Vo(w) dB (part C)')
xlabel('w (rad)')
ylabel('Av (dB)')

figure(4)
semilogx(W,vo2)
title('Vo(w) dB (part C)')
xlabel('w (rad)')
ylabel('Av (dB)')
grid on

w = pi;
CC = zeros(1000,1);
GG = zeros(1000,1);

for i = 1:1000
    crand = Cval + 0.05*randn();
    C(2, 1) = crand;
    C(2, 2) = -crand;
    C(3, 3) = L;
    Vm3 = (G+(1i*w*C))\F;
    CC(i) = crand;
    GG(i) = 20*log10(abs(Vm3(5))/10);
end

figure(5)
histogram(CC)
grid on
```

```
figure(6)
grid on
hist(GG)

clc
clear
clearvars
set(0,'DefaultFigureWindowStyle','docked')
% Transient simulation
R1 = 1;
R2 = 2;
C = 0.25;
L = 0.2;
R3 = 10;
a = 100;
R4 = 0.1;
R0 = 1000;
G = zeros(7,7);
Cm = zeros(7,7);
% Conductance value
G(1,1) = 1;
G(2,1) = -1/R1;
G(2,2) = 1/R1 + 1/R2;
G(2,6) = 1;
G(3,3) = 1/R3;
G(3,6) = -1;
G(4,3) = 1/R3;
G(4,7) = -1;
G(5,4) = -1/R4;
G(5,5) = 1/R4 + 1/R0;
G(6,2) = 1;
G(6,3) = -1;
G(7,4) = 1;
G(7,7) = -a;
% Capacitance value
Cm(2,1) = -C;
Cm(2,2) = C;
Cm(6,6) = -L;
V1 = zeros(7,1);
V2 = zeros(7,1);
```

```
V3 = zeros(7,1);
V_node1(1) = 0;
V_node2(1) = 0;
V_node3(1) = 0;
delta = 0.001;
A = (Cm/delta) + G;
F_index1 = zeros(7,1);
F_index2 = zeros(7,1);
F_index33 = zeros(7,1);
Vo_node1(1) = 0;
Vo_node2(1) = 0;
Vo_node3(1) = 0;

i = 1;
for j = delta:delta:1
    if j >= 0.03
        F_index1(1) = 3;
    end
    F_index2(1) = sin(2*pi*j/0.03);
    F_index33(1) = exp(-0.5*((j - 0.06)/0.03)^2);
    V1 = A\(Cm*V1/delta + F_index1);
    V2 = A\(Cm*V2/delta + F_index2);
    V3 = A\(Cm*V3/delta + F_index33);
    V_node1(i+1) = V1(1);
    V_node2(i+1) = V2(1);
    V_node3(i+1) = V3(1);
    Vo_node1(i+1) = V1(5);
    Vo_node2(i+1) = V2(5);
    Vo_node3(i+1) = V3(5);
    i = i+1;
end

figure(7)
plot(0:delta:1,V_node1,'b')
hold on
plot(0:delta:1,Vo_node1,'m')
title('Voltage vs time')
xlabel('time (s)')
ylabel('Voltage')
legend('input','output')
```

```
grid on
figure(8)
plot(0:delta:1,V_node2,'b')
hold on
plot(0:delta:1,Vo_node2,'m')
title('Voltage vs time')
xlabel('time (s)')
ylabel('Voltage')
legend('input','output')
grid on
figure(9)
plot(0:delta:1,V_node3,'b')
hold on
plot(0:delta:1,Vo_node3,'m')
title('Voltage vs time')
xlabel('time (s)')
ylabel('Voltage')
legend('input','output')
grid on

% Convert to frequency domain by taking the fourier transform
step_in = fft(V_node1); %fft -> Fast fourier transform
P_mag2_in = abs(step_in/1000);
P_1_in = P_mag2_in(1:1000/2+1);
P_1_in(2:end-1) = 2*P_1_in(2:end-1);
sample_f = (1/delta)*(0:(1000/2))/1000;
% Plot figure
figure(10)
plot(sample_f,P_1_in,'b')
step_out = fft(Vo_node1);
P2_out = abs(step_out/1000);
P1_out = P2_out(1:1000/2+1);
P1_out(2:end-1) = 2*P1_out(2:end-1);
sample_f = (1/delta)*(0:(1000/2))/1000;
hold on
plot(sample_f,P1_out,'m')
title('Frequency Content Step Input')
xlabel('Frequency (Hz)')
ylabel('Amplitude')
ylim([0 3])
```

```
legend('input','output')
grid on
% The fourier transform of the step input signal givess us a sinc function.
% This makes sense as we know that by taking the fourier transform of a step signal we
should get a sinc function.
% The High Frequency components are attenuated due to the fact that the filter
% is a low pass filter

step_in = fft(V_node2); %Take fourier transform
P_mag2_in = abs(step_in/1000);
P_1_in = P_mag2_in(1:1000/2+1);
P_1_in(2:end-1) = 2*P_1_in(2:end-1); % Calculate singel ended spectrum
figure(11)
plot(sample_f,P_1_in,'b')
grid on
step_out = fft(Vo_node2);
P2_out = abs(step_out/1000);
P1_out = P2_out(1:1000/2+1);
P1_out(2:end-1) = 2*P1_out(2:end-1);
hold on
plot(sample_f,P1_out,'m')
title('Frequency Content Sinusoidal Input')
xlabel('Frequency (Hz)')
ylabel('Amplitude')
ylim([0 3])
legend('input','output')
step_in = fft(V_node3); %Take fourier transform
P_mag2_in = abs(step_in/1000);
P_1_in = P_mag2_in(1:1000/2+1);
P_1_in(2:end-1) = 2*P_1_in(2:end-1); % Calculate singel ended spectrum
figure(12)
plot(sample_f,P_1_in,'b')
grid on
step_out = fft(Vo_node3);
P2_out = abs(step_out/1000);
P1_out = P2_out(1:1000/2+1);
P1_out(2:end-1) = 2*P1_out(2:end-1);
hold on
plot(sample_f,P1_out,'m')
title('Frequency Content Gaussian Input')
```

```
xlabel('Frequency (Hz)')
ylabel('Amplitude')
ylim([0 3])
legend('input','output')

clc
clearvars
set(0,'DefaultFigureWindowStyle','docked')

R1 = 1;
R2 = 2;
R3 = 10;
R4 = 0.1;
Ro = 1000;
C = 0.25;
L = 0.2;
a = 100;
cn1 = 0.00001;
cn2 = 0.0001;
cn3 = 0.01;

C1(1,:)=[C -C 0 0 0 0 0 0];
C1(2,:)=[-C C 0 0 0 0 0 0];
C1(3,:)=[0 0 cn1 0 0 0 0 0];
C1(4,:)=[0 0 0 0 0 0 0 0];
C1(5,:)=[0 0 0 0 0 0 0 0];
C1(6,:)=[0 0 0 0 0 -L 0 0];
C1(7,:)=[0 0 0 0 0 0 0 0];
C1(8,:)=[0 0 0 0 0 0 0 0];

C1

C2(1,:)=[C -C 0 0 0 0 0 0];
C2(2,:)=[-C C 0 0 0 0 0 0];
C2(3,:)=[0 0 cn2 0 0 0 0 0];
C2(4,:)=[0 0 0 0 0 0 0 0];
C2(5,:)=[0 0 0 0 0 0 0 0];
C2(6,:)=[0 0 0 0 0 -L 0 0];
C2(7,:)=[0 0 0 0 0 0 0 0];
C2(8,:)=[0 0 0 0 0 0 0 0];
```

```
C2

C3(1,:)=[C -C 0 0 0 0 0 0];
C3(2,:)=[-C C 0 0 0 0 0 0];
C3(3,:)=[0 0 cn3 0 0 0 0 0];
C3(4,:)=[0 0 0 0 0 0 0 0];
C3(5,:)=[0 0 0 0 0 0 0 0];
C3(6,:)=[0 0 0 0 0 -L 0 0];
C3(7,:)=[0 0 0 0 0 0 0 0];
C3(8,:)=[0 0 0 0 0 0 0 0];

C3

G(1,:)=[1 -1 0 0 0 0 0 1];
G(2,:)=[-1 1.5 0 0 0 1 0 0];
G(3,:)=[0 0 0.1 0 0 -1 0 0];
G(4,:)=[0 0 0 10 -10 0 1 0];
G(5,:)=[0 0 0 -10 10.0010 0 0 0];
G(6,:)=[0 1 -1 0 0 -L 0 0];
G(7,:)=[0 0 -10 1 0 0 0 0];
G(8,:)=[1 0 0 0 0 0 0 0];

G

F = [
    0   ;
    0   ;
    0   ;
    0   ;
    0   ;
    0   ;
    0   ;
    0   ;
    ];

% Given these values from lab manual
delta = 0.001;
standard_deviation = 0.03;
d = 0.06;
```

```
m = 1;
c_s = zeros(1,1000);
f_l = zeros(8,1,1000);

for i=1:1:1000
    f_l(8,1,i) = exp(-((i*delta - d)/standard_deviation)^2);
    f_l(3,1,i) = -0.001*randn;
    c_s(i) = f_l(3,1,i);
end

VL_1 = zeros(8,1,1000);
VL_2 = zeros(8,1,1000);
VL_3 = zeros(8,1,1000);

for i = 2:1:1000
    index1 = C1/delta + G;
    index2 = C2/delta + G;
    index3 = C3/delta + G;

    VL_1(:,:,i) = index1\(C1*VL_1(:,:,i-1)/delta +f_l(:,:,i));
    VL_2(:,:,i) = index2\(C1*VL_2(:,:,i-1)/delta +f_l(:,:,i));
    VL_3(:,:,i) = index3\(C1*VL_3(:,:,i-1)/delta +f_l(:,:,i));
end


VoL_1(1,:) = VL_1(5,1,:);
ViL_1(1,:) = VL_1(1,1,:);

VoL_2(1,:) = VL_2(5,1,:);
ViL_2(1,:) = VL_2(1,1,:);

VoL_3(1,:) = VL_3(5,1,:);
ViL_3(1,:) = VL_3(1,1,:);


figure(13)
plot((1:1000).*delta, VoL_1(1,:),'b')
hold on
plot((1:1000).*delta, ViL_1(1,:),'m')
title('Gaussian Pulse Function with added noise sources and cn1 = 0.00001')
```

```
xlabel('Time (s)')
ylabel('Voltage (v)')
grid on
legend('Output Voltage','Input Voltage')
hold off

figure(14)
histogram(c_s)
title('Noise Source Histogram')
grid on
xlabel('Current in amperes')

figure(15)
FF = abs(fftshift(fft(VoL_1(1,:))));
plot(((1:length(FF))/1000)-0.5,FF,'b')
xlabel('Frequency in hertz')
ylabel('Magnitude in decibels')
grid on
xlim([-0.04 0.04])
title('Gaussian Pulse Function with added noise sources and cn1 = 0.00001(Fourier
Transform)')

% CN 2 ----------------------------------------

figure(16)
plot((1:1000).*delta, VoL_2(1,:),'b')
hold on
plot((1:1000).*delta, ViL_2(1,:),'m')
title('Gaussian Pulse Function with added noise sources and cn2 = 0.0001')
xlabel('Time in seconds')
ylabel('Voltage in volts')
grid on
legend('Output Voltage','Input Voltage')
hold off

figure(17)
FF = abs(fftshift(fft(VoL_2(1,:))));
plot(((1:length(FF))/1000)-0.5,FF,'m')
xlabel('Frequency in hertz')
ylabel('Magnitude in decibels')
```

```
grid on
xlim([-0.04 0.04])
title('Gaussian Pulse Function with added noise sources and cn2 = 0.0001(Fourier
Transform)')

figure(18)
plot((1:1000).*delta, VoL_3(1,:),'b')
hold on
plot((1:1000).*delta, ViL_3(1,:),'m')
title('Gaussian Pulse Function with Added Noise Source and cn3 = 0.01 ')
xlabel('Time in seconds')
ylabel('Voltage in volts')
grid on
legend('Output Voltage','Input Voltage')
hold off

figure(19)
FF = abs(fftshift(fft(VoL_3(1,:))));
plot(((1:length(FF))/1000)-0.5,FF,'m')
xlabel('Frequency in hertz')
ylabel('Magnitude in decibels')
grid on
xlim([-0.04 0.04])
title('Gaussian Pulse Function added Noise and cn3 = 0.01 (Fourier Transform)')
% Varying timestep
%Varying timestep:
clc
clear
clearvars
set(0,'DefaultFigureWindowStyle','docked')
R1 = 1;
R2 = 2;
C = 0.25;
L = 0.2;
R3 = 10;
a = 100;
R4 = 0.1;
R0 = 1000;

G = zeros(7,7);
```

```
C_mat = zeros(7,7);
G(1,1) = 1;
G(2,1) = -1/R1;
G(2,2) = 1/R1 + 1/R2;
G(2,6) = 1;
G(3,3) = 1/R3;
G(3,6) = -1;
G(4,3) = 1/R3;
G(4,7) = -1;
G(5,4) = -1/R4;
G(5,5) = 1/R4 + 1/R0;
G(6,2) = 1;
G(6,3) = -1;
G(7,4) = 1;
G(7,7) = -a;
C_mat(2,1) = -C;
C_mat(2,2) = C;
C_mat(6,6) = -L;

G
C_mat

F = zeros(7,1);
V = zeros(7,1);

%Circuit with noise
In = 0.001; % as provided in assignment manual
Cn = 0.00001; % as provided in assignment manual
C_mat(3,3) = Cn;
G
C_mat

delta = 0.001;
A_transpse = C_mat/delta + G;

F = zeros(7,1);
V = zeros(7,1);
Vo_index(1) = 0;
Vi_index(1) = 0;
```

```
ii = 1;
for t = delta:delta:1
   F(1) = exp(-0.5*((t - 0.06)/0.03)^2);
   F(3) = In*normrnd(0,1);
    V = A_transpse\(C_mat*V/delta + F);
    Vi_index(ii + 1) = F(1);
    Vo_index(ii + 1) = V(5);
    ii = ii + 1;
end

X_input = fft(Vi_index);
P2_input = abs(X_input/1000);
P1_input = P2_input(1:1000/2+1);
P1_input(2:end-1) = 2*P1_input(2:end-1);
f = (1/delta)*(0:(1000/2))/1000;

X_output = fft(Vo_index);
P2_output = abs(X_output/1000);
P1_output = P2_output(1:1000/2+1);
P1_output(2:end-1) = 2*P1_output(2:end-1);
f = (1/delta)*(0:(1000/2))/1000;

C_sml = C_mat;
C_med = C_mat;
C_big = C_mat;
C_sml(3,3) = 0;
C_med(3,3) = 0.001;
C_big(3,3) = 0.1;
V_sml = zeros(7,1);
V_med = zeros(7,1);
V_big = zeros(7,1);
Vo_sml(1) = 0;
Voutput_med(1) = 0;
Vout_big(1) = 0;
Vi_index(1) = 0;
ii = 1;
for t = delta:delta:1
    F(1) = exp(-0.5*((t - 0.06)/0.03)^2);
    F(3) = In*normrnd(0,1);
    V_sml = (C_sml/delta + G)\(C_sml*V_sml/delta + F);
```

```
    V_med = (C_med/delta + G)\(C_med*V_med/delta + F);
    V_big = (C_big/delta + G)\(C_big*V_big/delta + F);
    Vo_sml(ii + 1) = V_sml(5);
    Voutput_med(ii + 1) = V_med(5);
    Vout_big(ii + 1) = V_big(5);
    Vi_index(ii + 1) = F(1);
    ii = ii + 1;
end

delta1 = 0.01;
Vinput_SmlStep(1) = 0;
Voutput_SmlStep(1) = 0;
V = zeros(7,1);
ii = 1;
for t = delta1:delta1:1
    F(1) = exp(-0.5*((t - 0.06)/0.03)^2);
    F(3) = In*normrnd(0,1);
    V = (C_mat/delta1 + G)\(C_mat*V/delta1 + F);
    Voutput_SmlStep(ii + 1) = V(5);
    Vinput_SmlStep(ii + 1) = F(1);
    ii = ii + 1;
end

delta2 = 0.1;
Vinput_bigStep(1) = 0;
Voutput_bigStep(1) = 0;
V = zeros(7,1);
ii = 1;
for t = delta2:delta2:1
    F(1) = exp(-0.5*((t - 0.06)/0.03)^2);
    F(3) = In*normrnd(0,1);
    V = (C_mat/delta2 + G)\(C_mat*V/delta2 + F);
    Voutput_bigStep(ii + 1) = V(5);
    Vinput_bigStep(ii + 1) = F(1);
    ii = ii + 1;
end

figure(20)
plot(0:delta1:1,Vinput_SmlStep,'b')
hold on
```

```
plot(0:delta1:1,Voutput_SmlStep,'m')
title('Voltage vs time with timestep = 0.003')
xlabel('time (s)')
ylabel('Voltage')
legend('input','output')
grid on

figure(21)
plot(0:delta2:1,Vinput_bigStep,'m')
hold on
plot(0:delta2:1,Voutput_bigStep,'b')
title('Voltage vs time with time step = 0.01')
xlabel('time (s)')
ylabel('Voltage')
legend('input','output')
grid on
```