Assignment 4

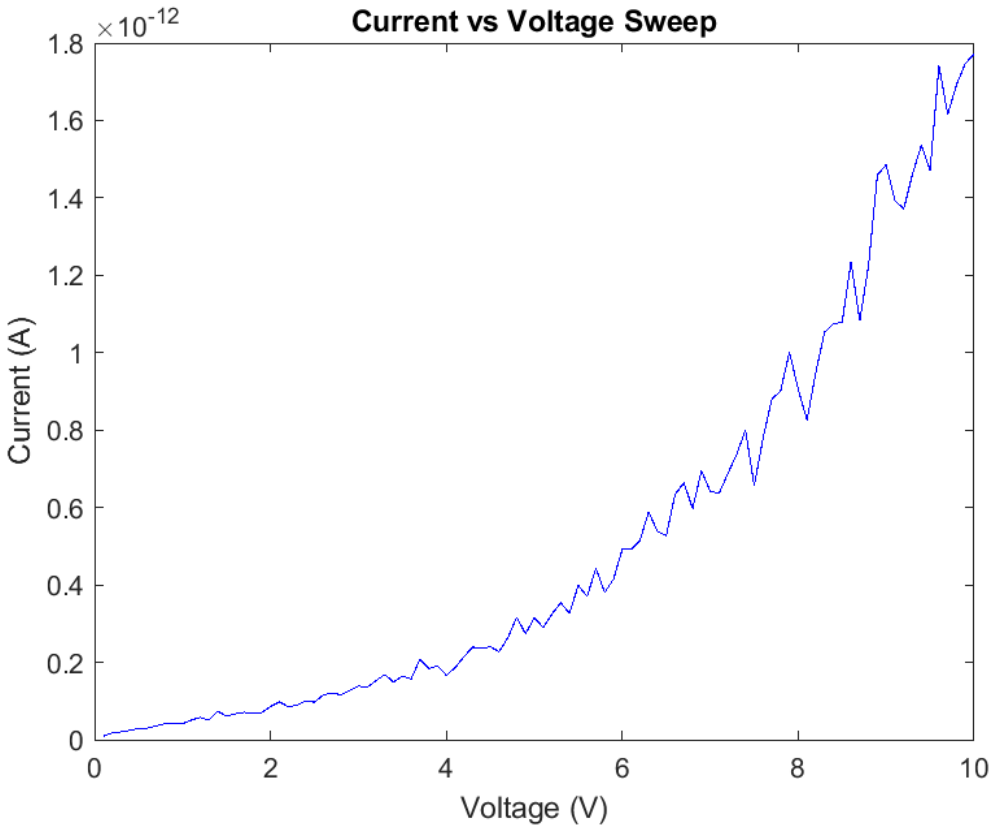
```
% Connor Warden
% 101078296
```

Question 1

In order to model the voltage sweep a simple change had to be made to the code from lab 3.

```
for vsweep = 0.1:0.1:10 % sweep from 0.1 to 10 in 0.1 increments
    ...
    ...
    ...

    itr = int16(vsweep*10); % current_mean needs a integer for iteration
                            % This makes the current sweep value equal to a
                            % whole number. E.G. vsweep = 0.1 => itr = 1
    current_mean(itr) = mean(current_current);  % Finds average current at each
                                                % voltage value
end

voltage = linspace(0.1,10,100);     % Represents all of the voltages swept
figure(1)
plot(voltage, current_mean, 'b')    % Plots current at each voltage
xlabel("Voltage (V)")
ylabel("Current (A)")
title("Current vs Voltage Sweep")
```

The result of this was a plot containing the average current at each voltage, shown on the following page.


Current vs Voltage Sweep
×10^-12
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Current (A)
0
2
4
6
8
10
Voltage (V)

Question 2

Question 3

Part A

The matrices for C G and F are shown below.

```
% R1 = 1;  %Ohms
% R2 = 2;  %Ohms
% R3 = 10;  %Ohms
% R4 = 0.1;  %Ohms
% Ro = 1000;  %Ohms
% C1 = 0.25;  %Farads
% L1 = 0.2;  %Henry
% alpha = 100;
% G1 = 1/R1;
% G2 = 1/R2;
% G3 = 1/R3;
% G4 = 1/R4;
% G = [G1 -G1 0 0 0 1 0 0;
%     -G1 G1+G2 0 0 0 0 1 0;
%     0 0 G3 0 0 0 -1 0;
%     0 0 0 G4 -G4 0 0 1;
%     0 0 0 -G4 G4+G5 0 0 0;
%     1 0 0 0 0 0 0 0;
%     0 1 -1 0 0 0 0 0;
%     0 0 -alpha*G3 1 0 0 0 0];
% C = [C1 -C1 0 0 0 0 0 0;
%     -C1 C1 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 0 0 0 0 0 0;
%     0 0 0 0 0 0 -L1 0;
%     0 0 0 0 0 0 0 0];
% F = [0 0 0 0 0 vin 0 vout]';
```

These were implemented using pre built functions rather than written as shown above, however the result still matches that of those shown above.

Part B

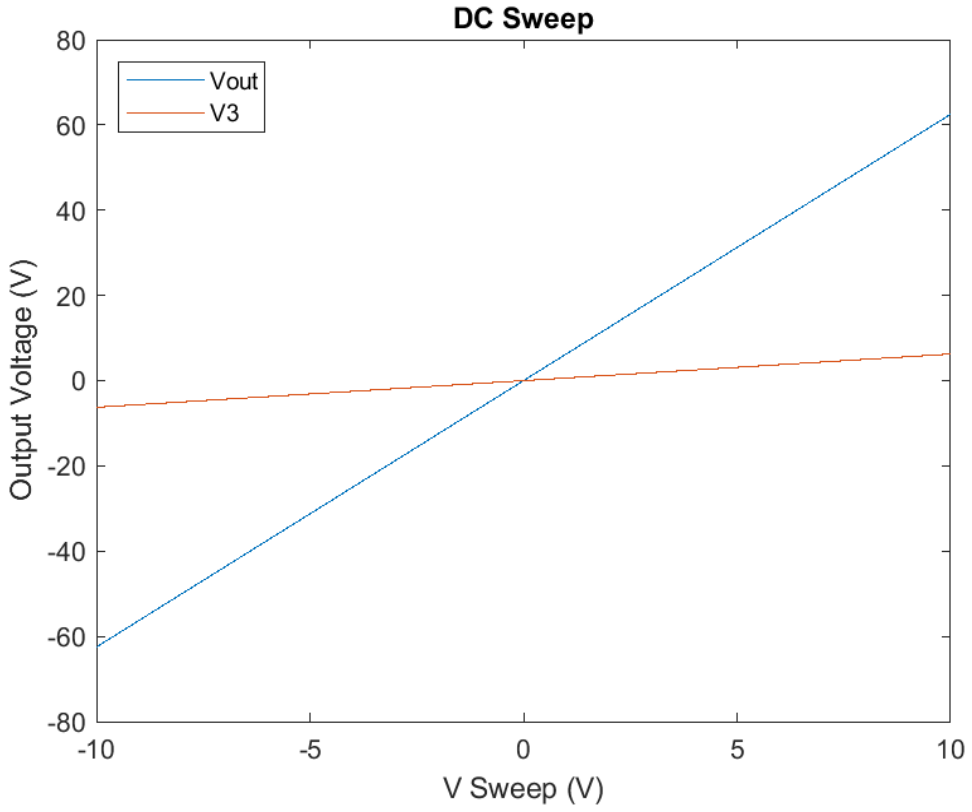
DC Case

The following code was used for for a DC sweep from -10 to 10, with 20 steps.

```
clc; close all;  clear all;  %initialization of the matlab environment
NrNodes = 5;

global G C b L; %define global variables

G = zeros(NrNodes,NrNodes);
C = zeros(NrNodes,NrNodes);
b = zeros(NrNodes,1);
L = zeros(NrNodes,NrNodes);

%-------------------------------------------------------------------------
% List of the components (netlist):
%-------------------------------------------------------------------------
R1 = 1;  %Ohms
R2 = 2;  %Ohms
R3 = 10;  %Ohms
R4 = 0.1;  %Ohms
Ro = 1000;  %Ohms
C1 = 0.25;  %Farads
L1 = 0.2;  %Henry
alpha = 100;

% DC CASE 2B
Vout = zeros(21,1);
v3 = zeros(21,1);

vin_a = zeros(21,1);
for i = 1:21
    [vin] = dc_sweep(i);
    G = zeros(NrNodes,NrNodes);
    C = zeros(NrNodes,NrNodes);
    b = zeros(NrNodes,1);
    L = zeros(NrNodes,NrNodes);
    vol(1,0, vin)
    res(1,2,R1)
    cap(1,2,C1)
    res(2,0,R2)
    ind(2,3,L1)
    res(3,0,R3)
    res(4,5,R4)
    res(5,0,Ro)
```

```
    vcvs(4,0,3,0,alpha/R3)

    A =  G;  %Enter A here!
    X = A\b;  % The operator "\" is an efficient way to solve AX=b.
    Vout(i,1) = X(5);
    v3(i,1) = X(3);

    vin_a(i,1) = vin;
end

interval = linspace(-10,10,21)';

figure(1)
plot(interval, Vout)
hold on
plot(interval, v3)
legend({'Vout','V3'},'Location','northwest')
title("DC Sweep")
xlabel("V Sweep (V)")
ylabel("Output Voltage (V)")
```

This was implemented using prebuilt matrix stamps, which are included in the attached code folder. The generated plot is shown below.

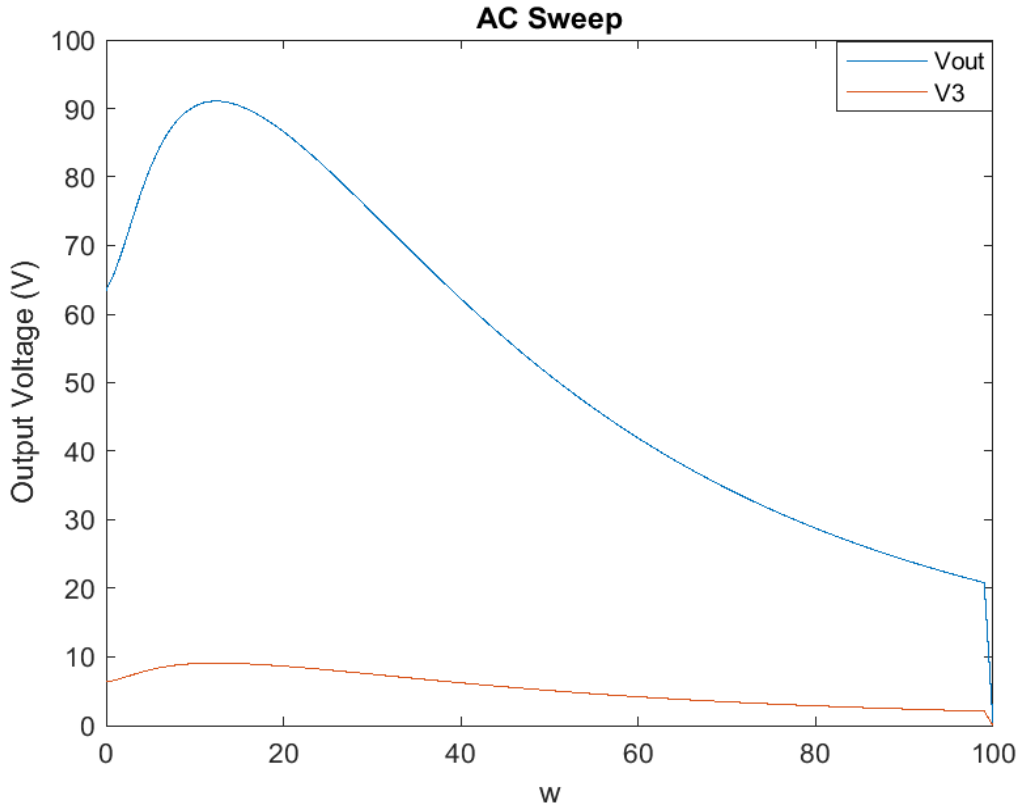


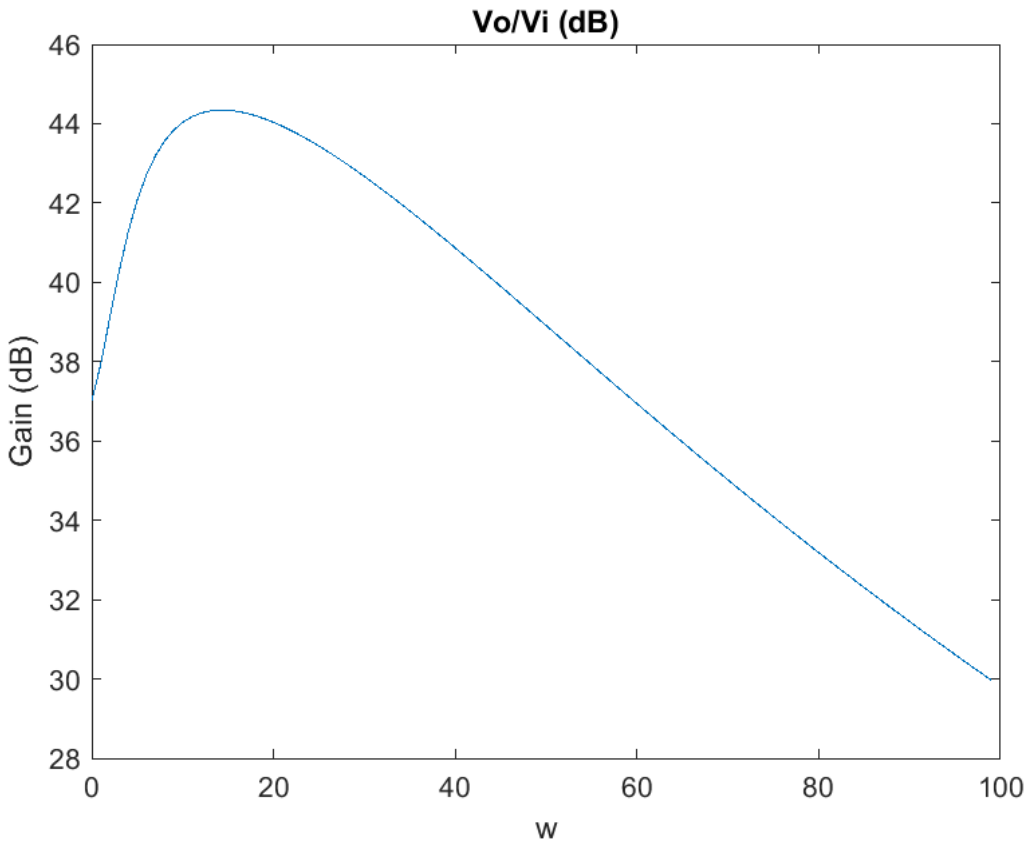
AC Case

The first AC simulation was done by plotting Vo and V3 as function of w. Gain was then calculated and also plotted. All code used is shown below.

```
clc; close all;  clear all;  %initialization of the matlab environment
NrNodes = 5;

global G C b L; %define global variables

G = zeros(NrNodes,NrNodes);
C = zeros(NrNodes,NrNodes);
b = zeros(NrNodes,1);
L = zeros(NrNodes,NrNodes);

%-------------------------------------------------------------------------
% List of the components (netlist):
%-------------------------------------------------------------------------
R1 = 1;  %Ohms
R2 = 2;  %Ohms
R3 = 10;  %Ohms
R4 = 0.1;  %Ohms
Ro = 1000;  %Ohms
C1 = 0.25;  %Farads
L1 = 0.2;  %Henry
alpha = 100;

Vout = zeros(101,1);
v3 = zeros(101,1);
vin_a = zeros(21,1);
for i = 1:21
    [vin] = dc_sweep(i);
    G = zeros(NrNodes,NrNodes);
    C = zeros(NrNodes,NrNodes);
    b = zeros(NrNodes,1);
    L = zeros(NrNodes,NrNodes);
    vol(1,0, vin)
    res(1,2,R1)
    cap(1,2,C1)
    res(2,0,R2)
    ind(2,3,L1)
    res(3,0,R3)
    res(4,5,R4)
    res(5,0,Ro)
    vcvs(4,0,3,0,alpha/R3)
```

```
    for n=1:100
        w = n;
        s = 1i*w;
        A =  G+C*s;  %Enter A here!
        X = A\b;  % The operator "\" is an efficient way to solve AX=b.
        Vout(n,1) = X(5);
        v3(n,1) = X(3);
    end

    vin_a(i,1) = vin;
end


%FOR AC CASE 2C
interval = linspace(0,100,101)';

plot(interval, Vout)
hold on
plot(interval, v3)
legend({'Vout','V3'},'Location','northwest')
title("AC Sweep")
xlabel("w")
ylabel("Output Voltage (V)")

figure(2)
plot(interval, 20*log(Vout/vin_a));
title("Vo/Vi (dB)")
xlabel("w")
ylabel("Gain")
```

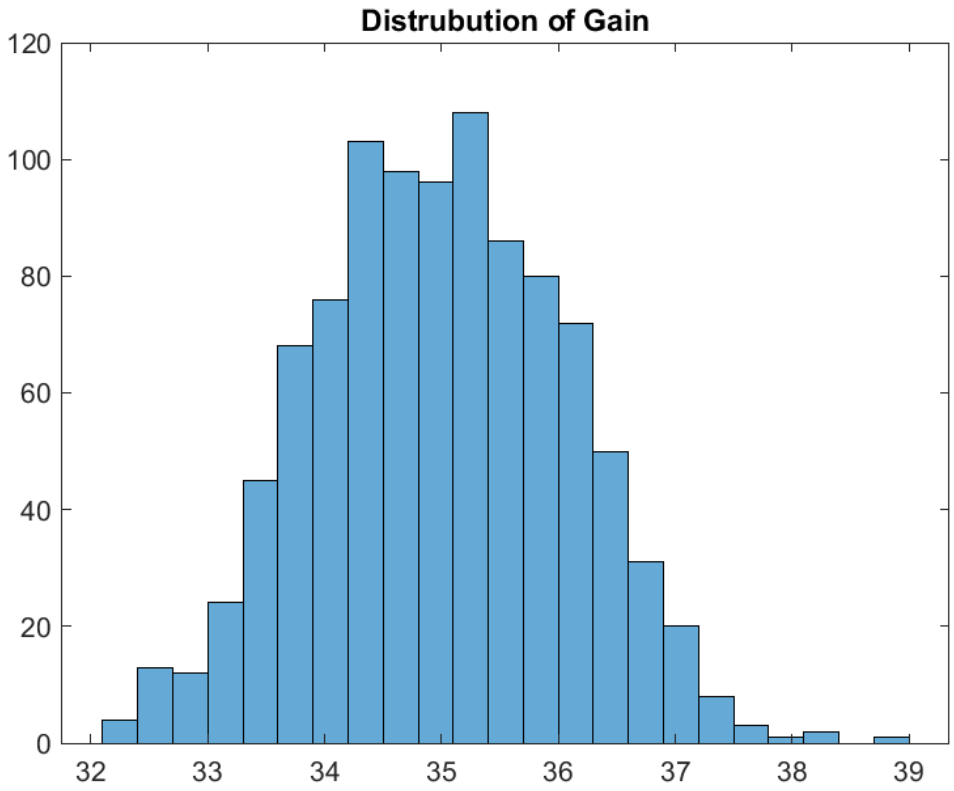

AC Sweep
Vout
V3
Output Voltage (V)
w
0
10
20
30
40
50
60
70
80
90
100
0
20
40
60
80
100


Vo/Vi (dB)
Gain (dB)
w
28
30
32
34
36
38
40
42
44
46
0
20
40
60
80
100

The second AC case was concerned with the determining gain as a function of random perturbations in the C matrix. This was achieved with the following code, with 1000 iterations done rather than the 100 used for the previous examples.

```
clc; close all;  clear all;  %initialization of the matlab environment
NrNodes = 5;

global G C b L; %define global variables

G = zeros(NrNodes,NrNodes);
C = zeros(NrNodes,NrNodes);
b = zeros(NrNodes,1);
L = zeros(NrNodes,NrNodes);

%-------------------------------------------------------------------------
% List of the components (netlist):
%-------------------------------------------------------------------------
R1 = 1;  %Ohms
R2 = 2;  %Ohms
R3 = 10;  %Ohms
R4 = 0.1;  %Ohms
Ro = 1000;  %Ohms
C1 = 0.25;  %Farads
L1 = 0.2;  %Henry
alpha = 100;

Vout = zeros(1001,1);
vin_a = zeros(1001,1);
c = zeros(1001,1);
for i = 1:1001
    C1 = 0.05.*randn + 0.25;
    vin = 10; % set Vin as 10 for this part
    G = zeros(NrNodes,NrNodes);
    C = zeros(NrNodes,NrNodes);
    b = zeros(NrNodes,1);
    L = zeros(NrNodes,NrNodes);
    vol(1,0, vin)
    res(1,2,R1)
    cap(1,2,C1)
    res(2,0,R1)
    ind(2,3,L1)
    res(3,0,R3)
    res(4,5,R4)
    res(5,0,Ro)
    vcvs(4,0,3,0,alpha/R3)
    % The following 8 lines are for AC case 2C
```

```
    w = pi;
    s = 1i*w;
    A =  G+C*s;  %Enter A here!
    X = A\b;  % The operator "\" is an efficient way to solve AX=b.
    Vout(i,1) = X(5);

    c(i,:) = C1; % stores the value of C at each iteration, for plotting
    vin_a(i,1) = vin;

end

interval = linspace(0,1000,1001)';

gain = 20*log(Vout./vin);

figure(1)
histogram(abs(gain))
title('Distrubution of Gain')
```



Question 4

This portion of the assignment dealt with determining the transient response of the same circuit as part 3. Three cases needed to be considered, those being:

- A step that transitions from 0 to 1 at 0.03s.
- A sin(2πft) function with f = 1/(0.03) 1/s.
- A gaussian pulse with a magnitude of 1, std dev. of 0.03s and a delay of 0.06s.

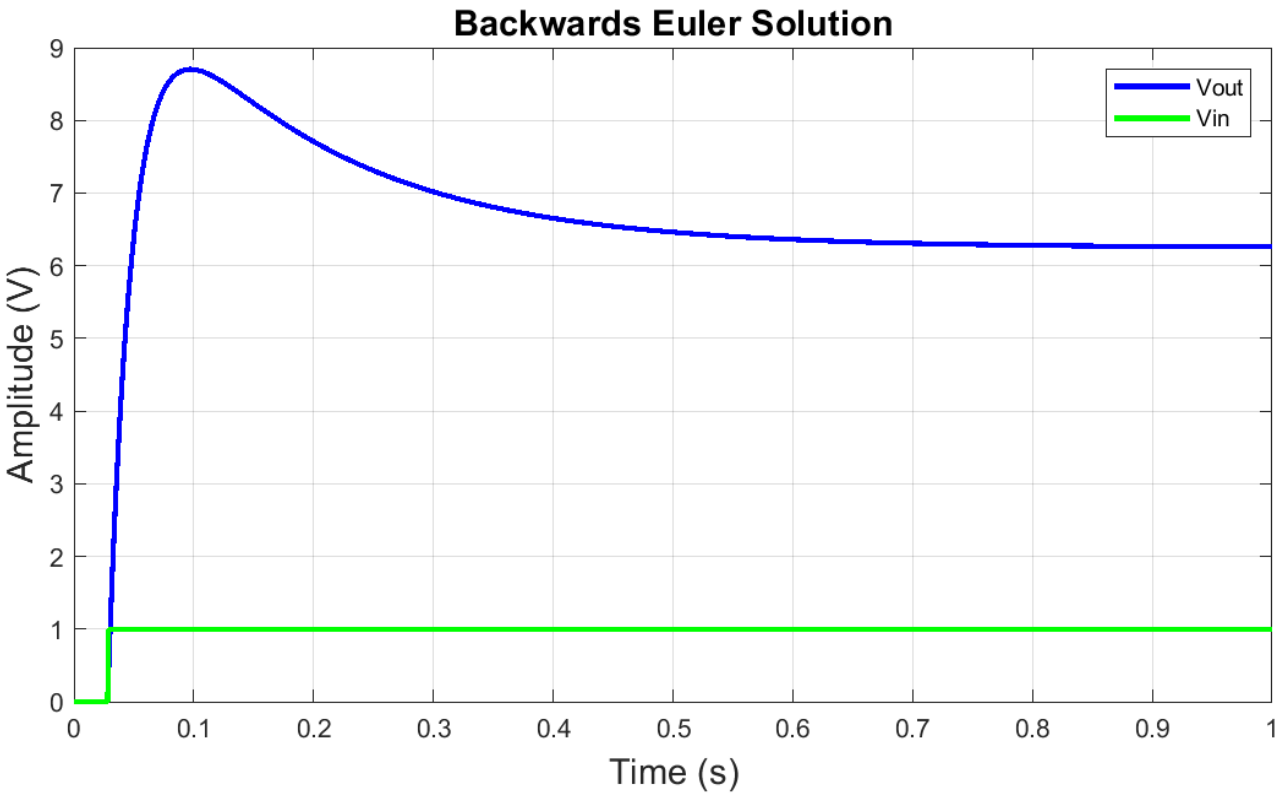
This was simulated using the following code.

```
clc; close all; clear all;

NrNodes = 5;

global G C b L; %define global variables

X(:,1) = zeros(8,1);
v_in(:,1) = 0;
v_out(:,1) = 0;



%Netlist
G = zeros(NrNodes,NrNodes);
C = zeros(NrNodes,NrNodes);
b = zeros(NrNodes,1);
L = zeros(NrNodes,NrNodes);
R1 = 1;  %Ohms
R2 = 2;  %Ohms
R3 = 10;  %Ohms
R4 = 0.1;  %Ohms
Ro = 1000;  %Ohms
C1 = 0.25;  %Farads
L1 = 0.2;  %Henry
alpha = 100;

type = 1;  % for step
% type =2;    % for sine
%type =3;   % for pulse

f = 1/0.03;
h = 0.001;
t = linspace(0, 1, 1/h);

vin = PWL(1,type, f);
```

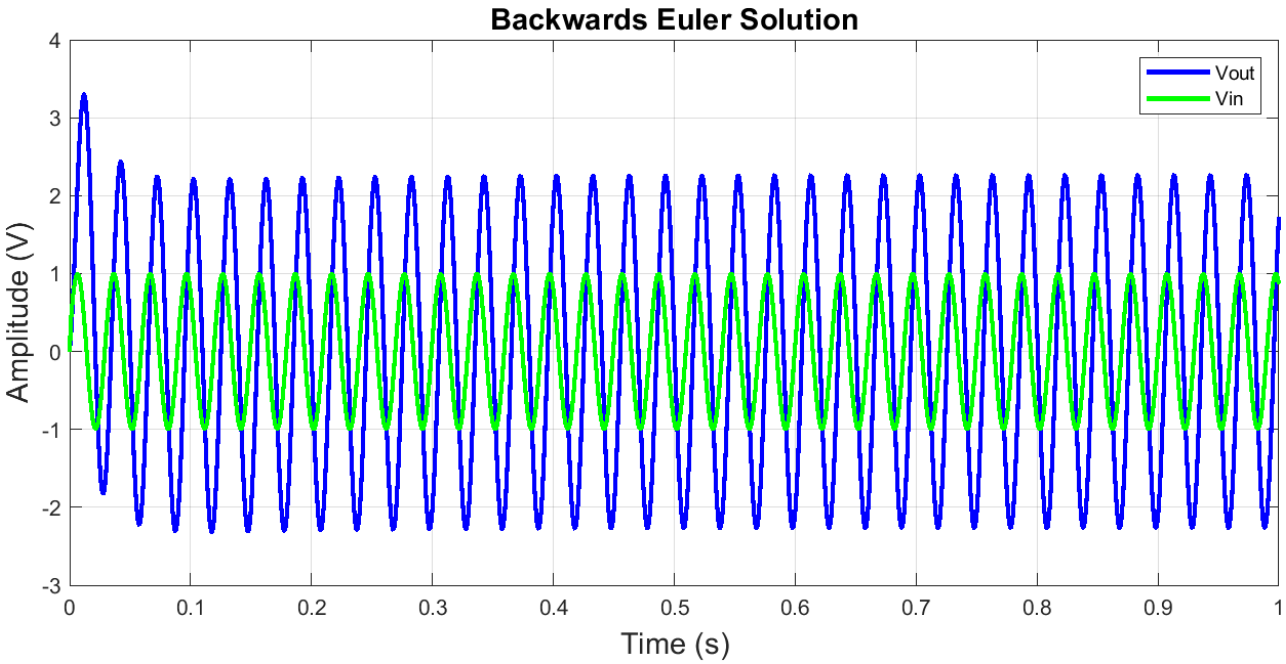
```
vol(1,0, vin)
res(1,2,R1)
cap(1,2,C1)
res(2,0,R2)
ind(2,3,L1)
res(3,0,R3)
res(4,5,R4)
res(5,0,Ro)
vcvs(4,0,3,0,alpha/R3)
X(:,1) = 0;

Vout = zeros(1001,1);
vin_a = zeros(1001,1);
for i = 2:(numel(t))
    vin = PWL(i, type, f); % determines the vin based on the current time,
defined by numel(t)
    b(6) = vin; % corresponds to the location of the voltage source within my
calculations
    X(:,i) = ((C/h) + G) \ ((C/h)*X(:,i-1) + b(:,1)); % euler implementation
    v_out(i) = X(5,i); % V5 is the same as Vout in this circuit
    v_in(i) = vin;
end

figure(1);
plot(t, v_out,'b','LineWidth',2);
hold on
plot(t, v_in,'g','LineWidth',2);
grid;
title('Backwards Euler Solution', 'FontSize',14);
xlabel('Time (s)','FontSize',14);
ylabel('Amplitude (V)','FontSize',14);
legend({'Vout','Vin'},'Location','northeast')

function [val] = PWL(t,type,f)

if type == 1
    t1 = 30;
    if t < t1
        val = 0;
    elseif t == t1
        val = 1;
    else
        val = 1;
    end
end
```

```
 if type == 2
     val = sin(2 * pi * f * t/1000);
 end

 if type == 3
        val = 37.6 * (1/(15*sqrt(2*pi))*exp((-1/2)*((t-90)/15))^2); %
multiplying by 37.6 gives the proper pulse
 end

 end
```

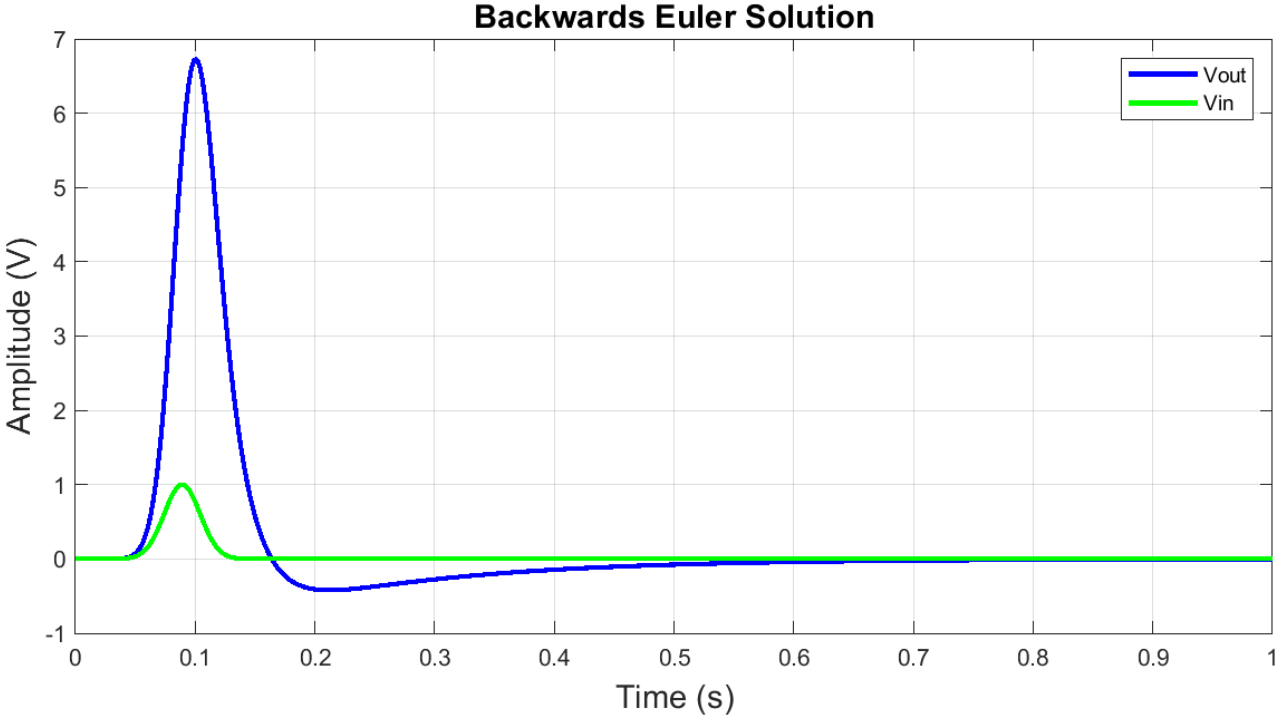
With this code the circuit could be simulated for each test case by changing the value of 'type'. The step result is shown below.



The sine input response is shown on the following page. Lowering the frequency resulted in a larger period of the sine response, and the opposite held true when the frequency was increased. The following plot is shown with a frequency of approximately 33 Hz.



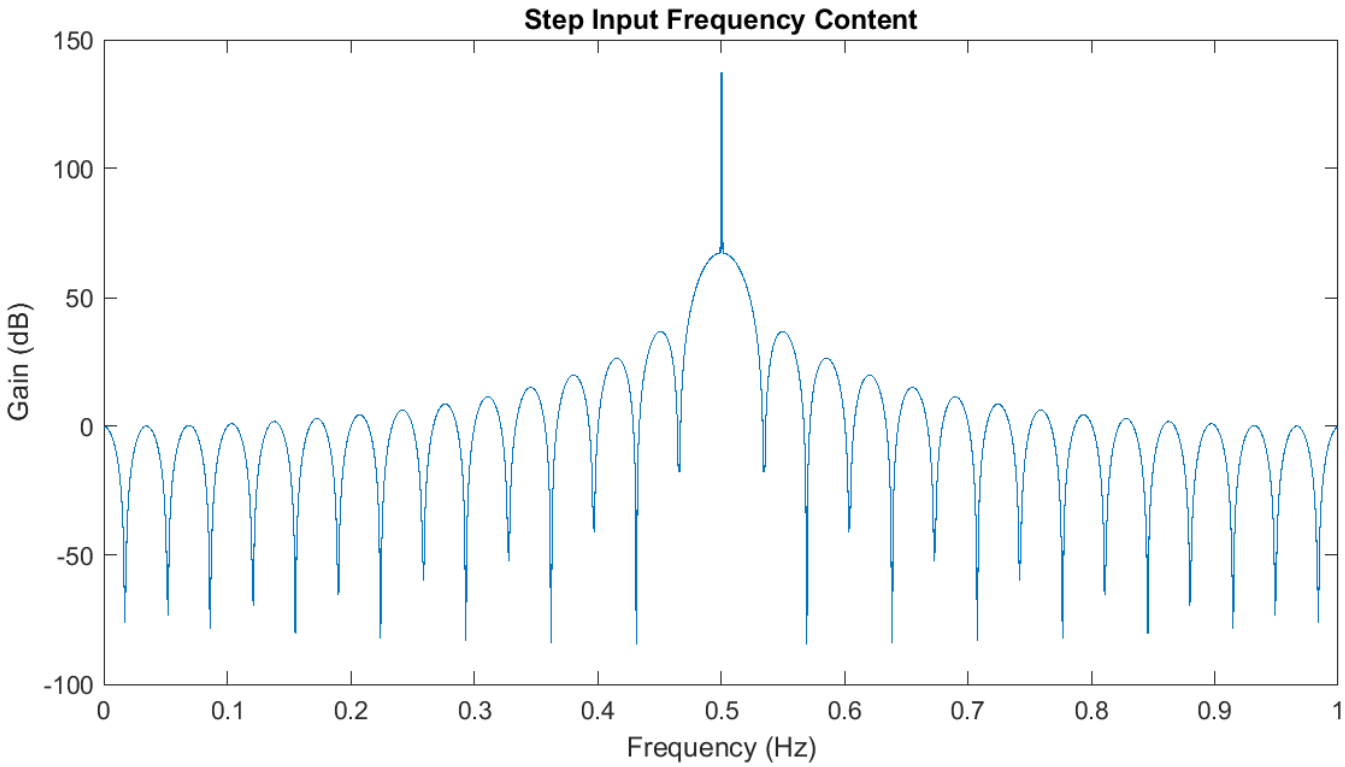
The final case involved simulating with a pulse containing a delay and standard deviation of 0.03 seconds. The result is shown below.

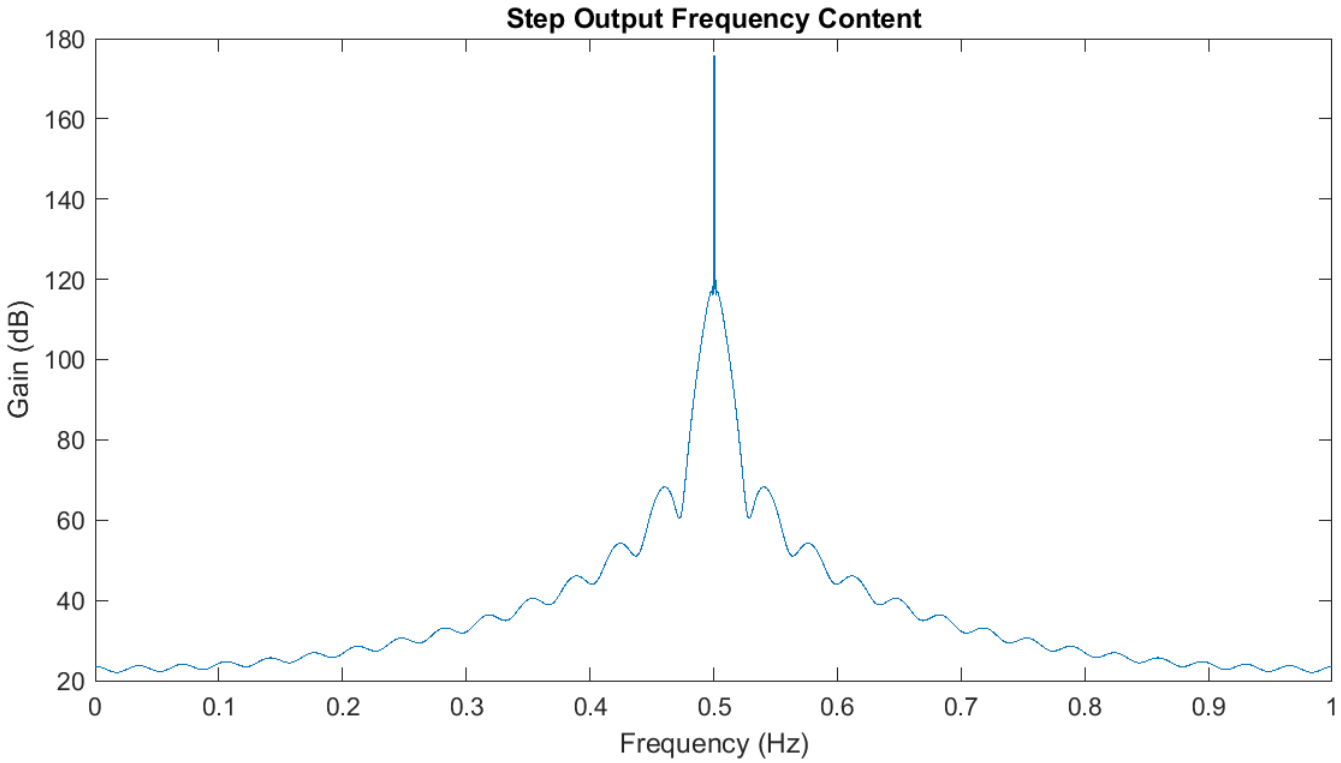


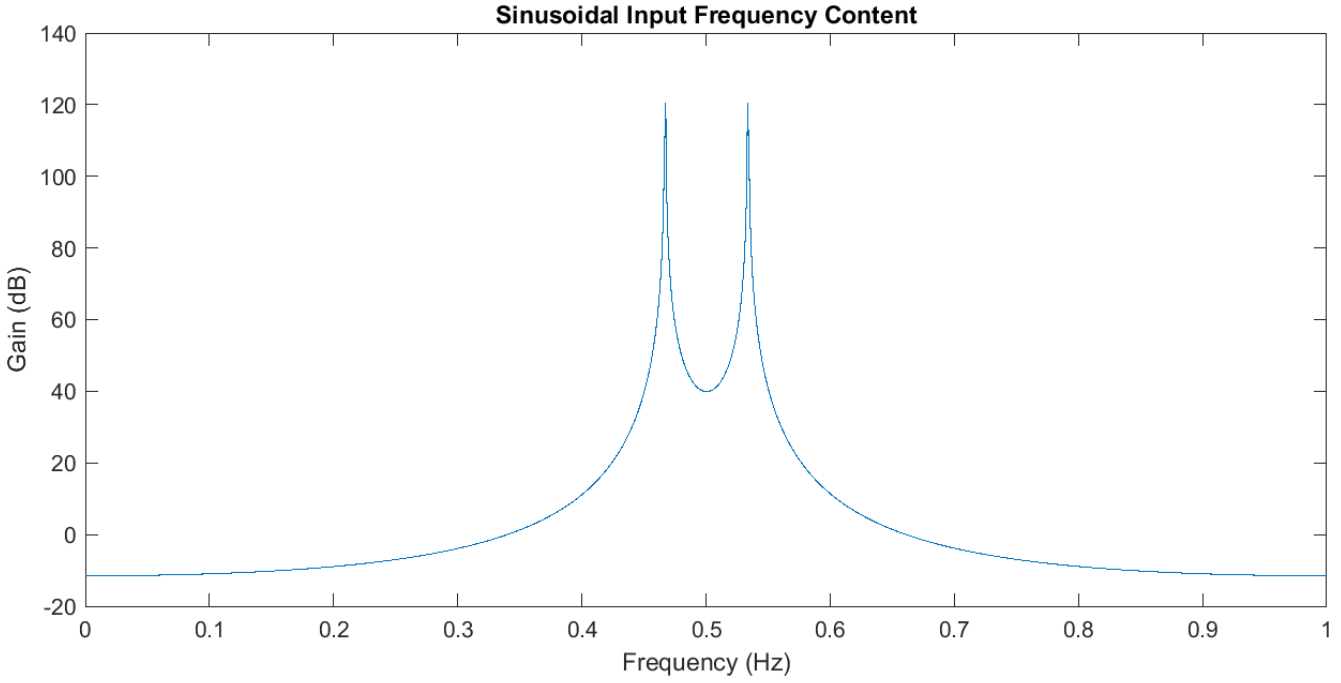
After this was complete the Fourier transforms of all three signals were plotted using the following code.

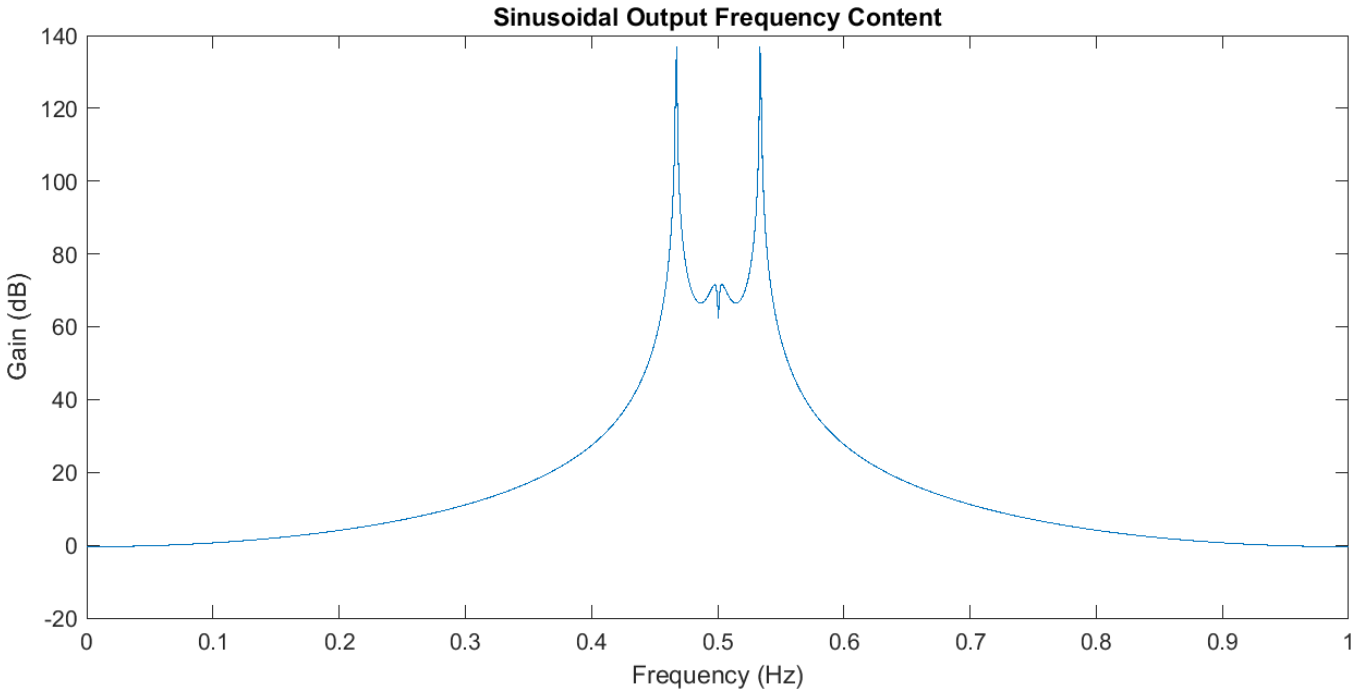
```
 f_in =  fft(v_in);
 f_out = fft(v_out);
 f_in_shift =  fftshift(f_in);
 f_out_shift =  fftshift(f_out);
 figure(6)
 plot(t, 20*log(f_in_shift))
 ylabel("Gain (dB)")
 xlabel("Frequency (Hz)")
 title("Step Input Frequency Content") % naming convention changes depending on
what input is set
 figure(7)
 plot(t, 20*log(f_out_shift))
 ylabel("Gain (dB)")
 xlabel("Frequency (Hz)")
 title("Step Output Frequency Content")
```

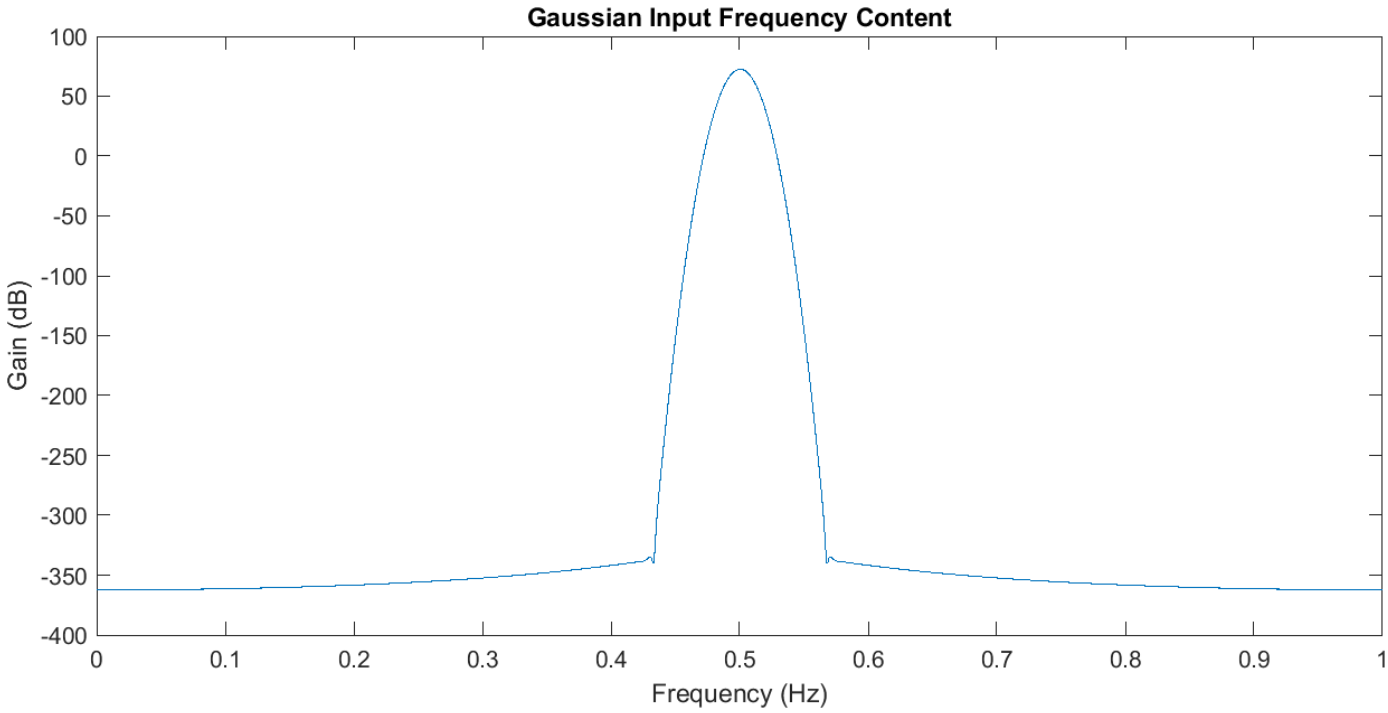
After running this code for all three cases the following plots were generated.

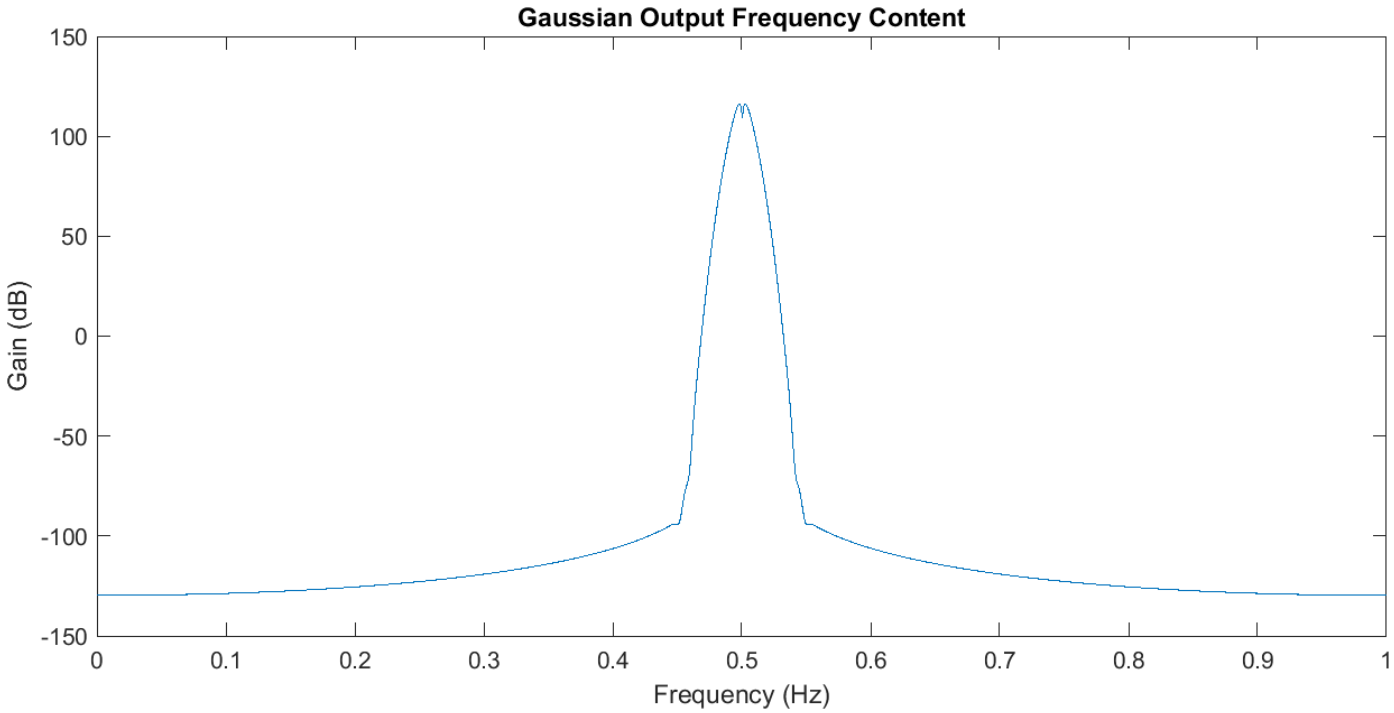



Step Output Frequency Content
Gain (dB)
Frequency (Hz)
180
160
140
120
100
80
60
40
20
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Sinusoidal Input Frequency Content
Gain (dB)
Frequency (Hz)
140
120
100
80
60
40
20
0
-20
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Sinusoidal Output Frequency Content
Gain (dB)
Frequency (Hz)
140
120
100
80
60
40
20
0
-20
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Gaussian Input Frequency Content
Gain (dB)
Frequency (Hz)
100
50
0
-50
-100
-150
-200
-250
-300
-350
-400
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



Question 5

The updated C matrix is as follows.

C

8x8 double

	1	2	3	4	5	6	7	8
1	0.2500	-0.2500	0	0	0	0	0	0
2	-0.2500	0.2500	0	0	0	0	0	0
3	0	0	1.0000e-05	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	-0.2000	0
8	0	0	0	0	0	0	0	0

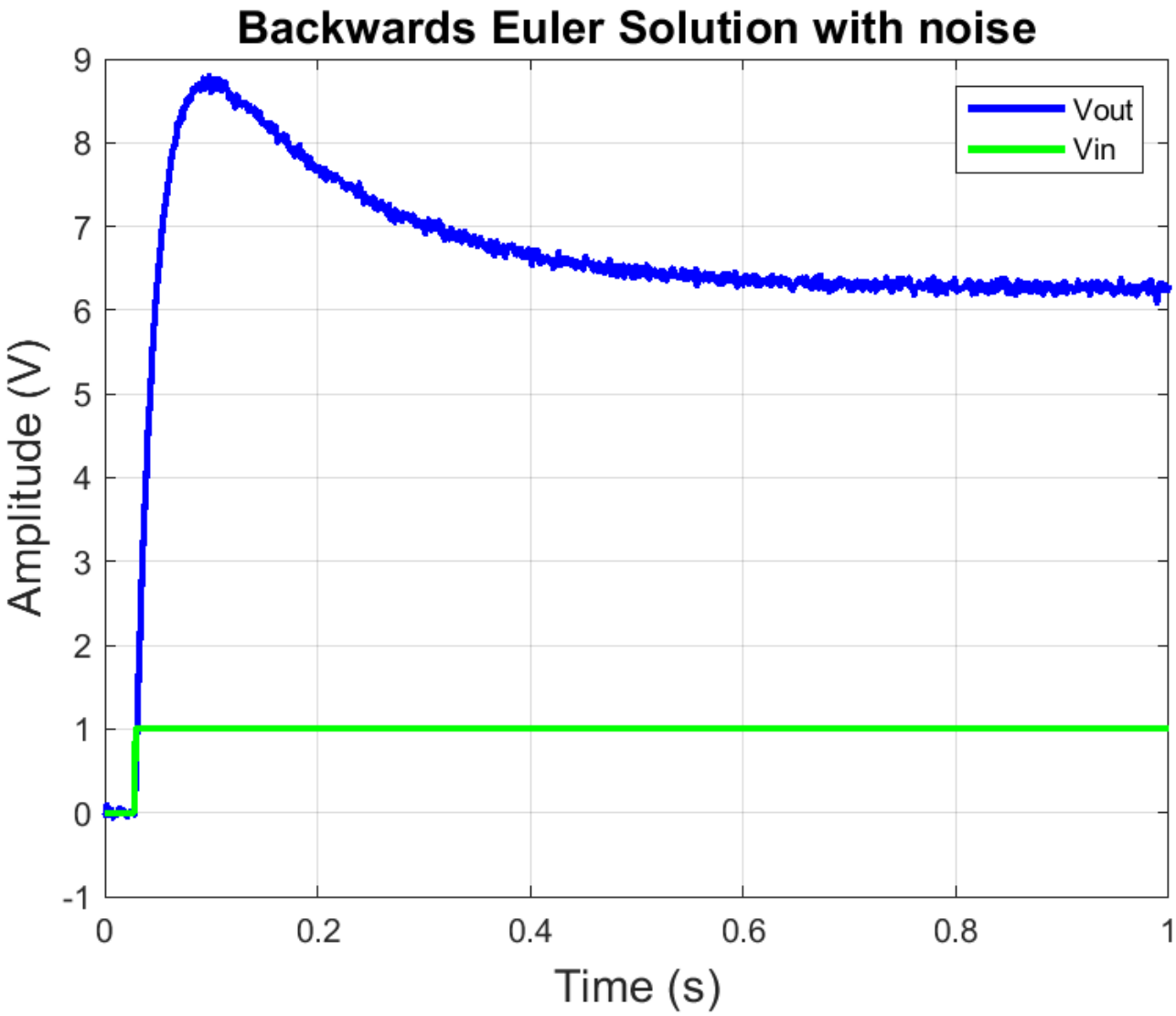
The code used to generate the plots with noise only required slight modification.

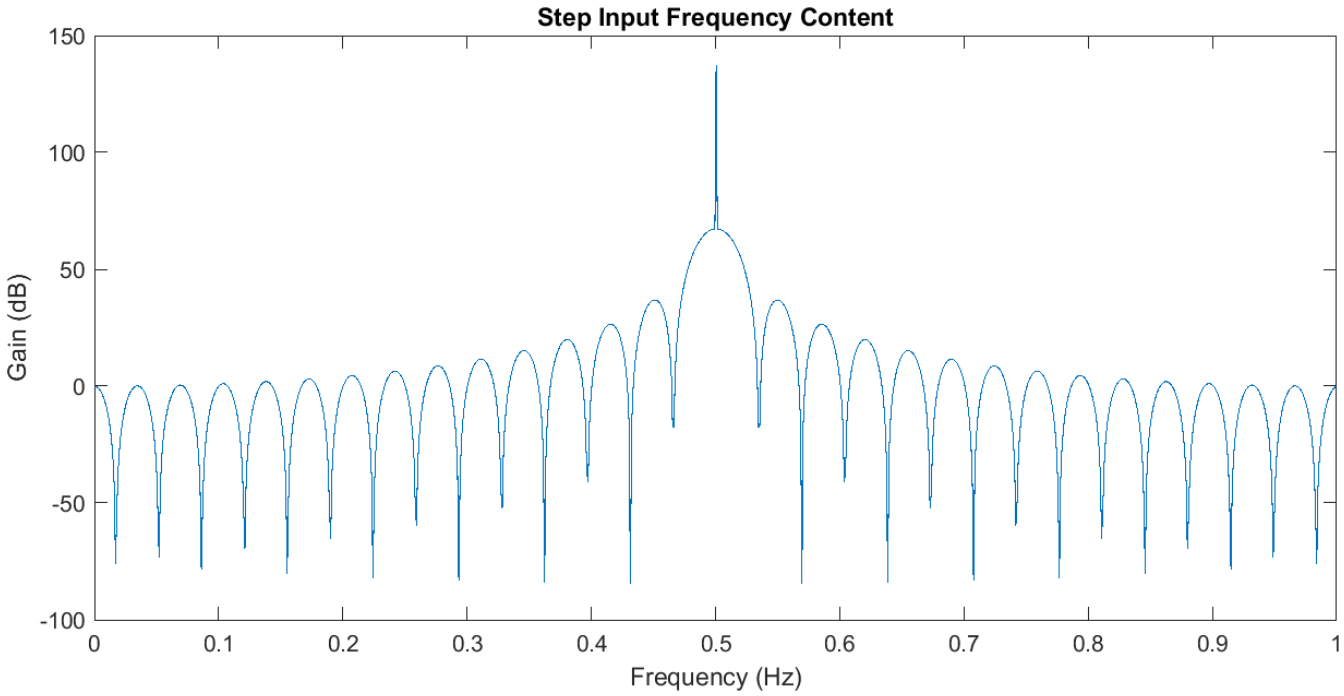
```
Vout = zeros(1001,1);
vin_a = zeros(1001,1);
for i = 2:(numel(t))
    [vin, curr] = PWL(i, type, f);
    b(6) = vin;
    b(3) = curr; % this line was added for noise
    X(:,i) = ((C/h) + G) \ ((C/h)*X(:,i-1) + b(:,1));
```

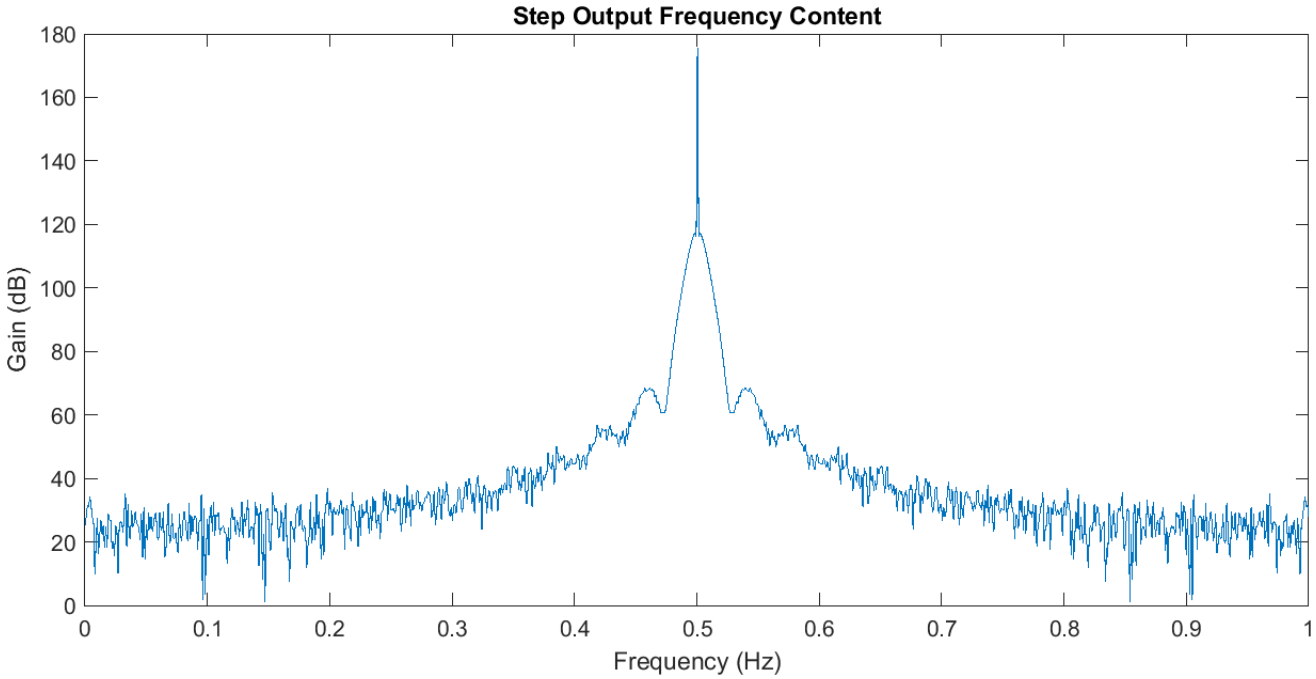
```
    v_out(i) = X(5,i);
    v_in(i) = vin;
end
...
...
...
function [val, current] = PWL(t,type,f)


if type == 1
    t1 = 30;
    if t < t1
        val = 0;
    elseif t == t1
        val = 1;
    else
        val = 1;
    end
    current = 0.0005.*randn + 0.0001; % added for noise
end

if type == 2
    val = sin(2 * pi * f * t/1000);
    current = 0.0005.*randn + 0.0001; % added for noise
end

if type == 3
    val = 37.6* (1/(15*sqrt(2*pi))*exp((-1/2)*(((t-90)/15))^2));
    current = 0.0005.*randn + 0.0001; % added for noise
end

end
```

The following plots are each of the different input signals along with their associated frequency plots.

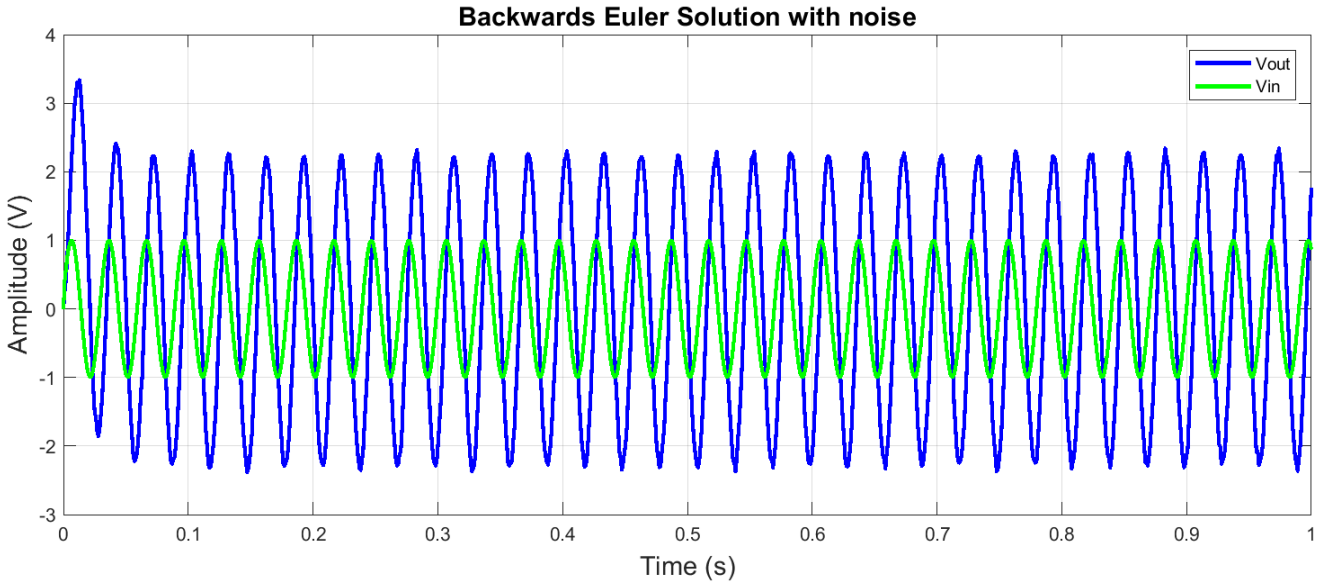
Step

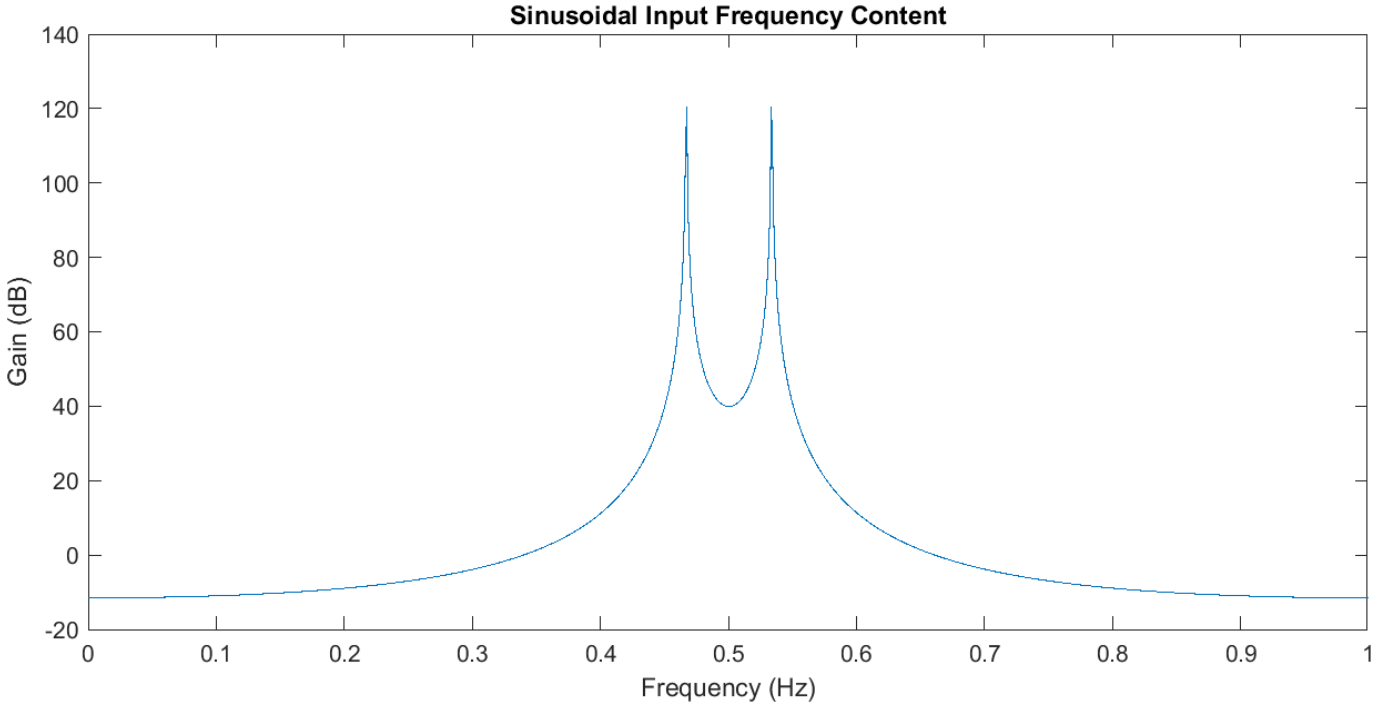


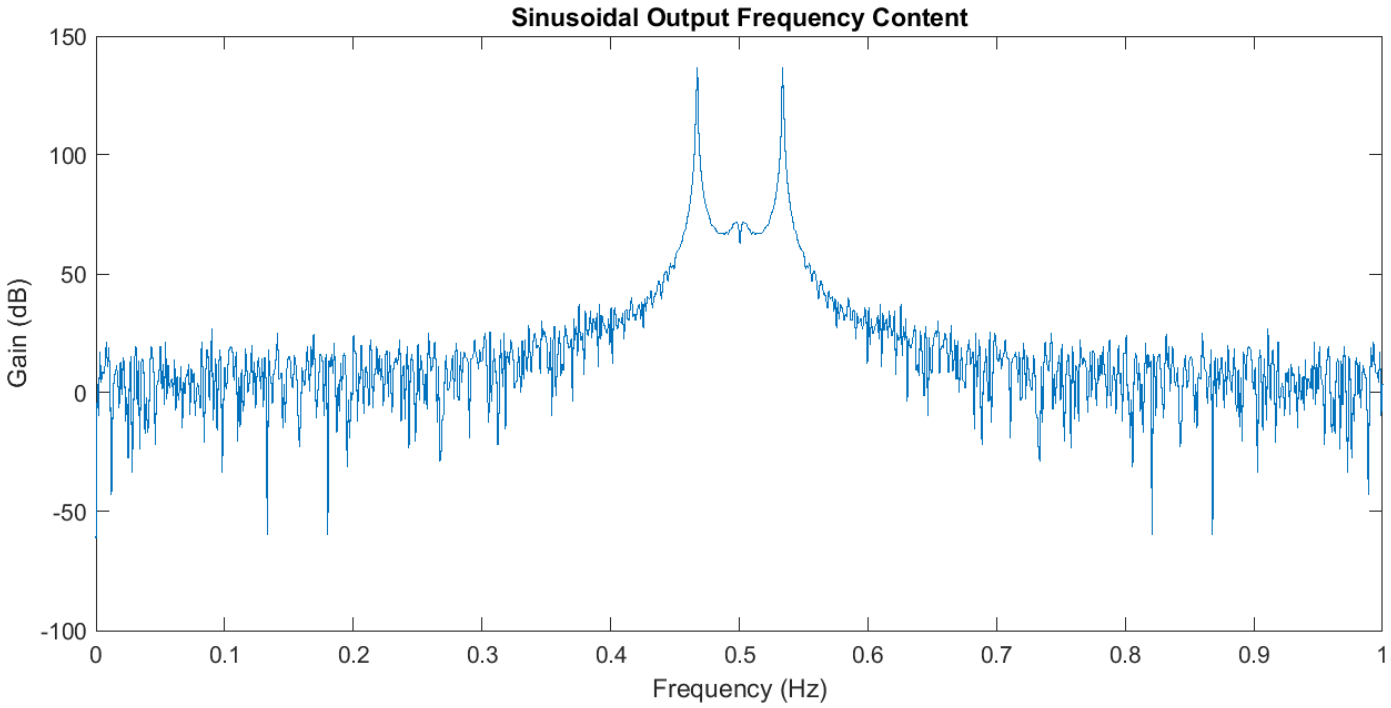

Step Input Frequency Content
Gain (dB)
Frequency (Hz)
150
100
50
0
-50
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Step Output Frequency Content
Gain (dB)
Frequency (Hz)
180
160
140
120
100
80
60
40
20
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

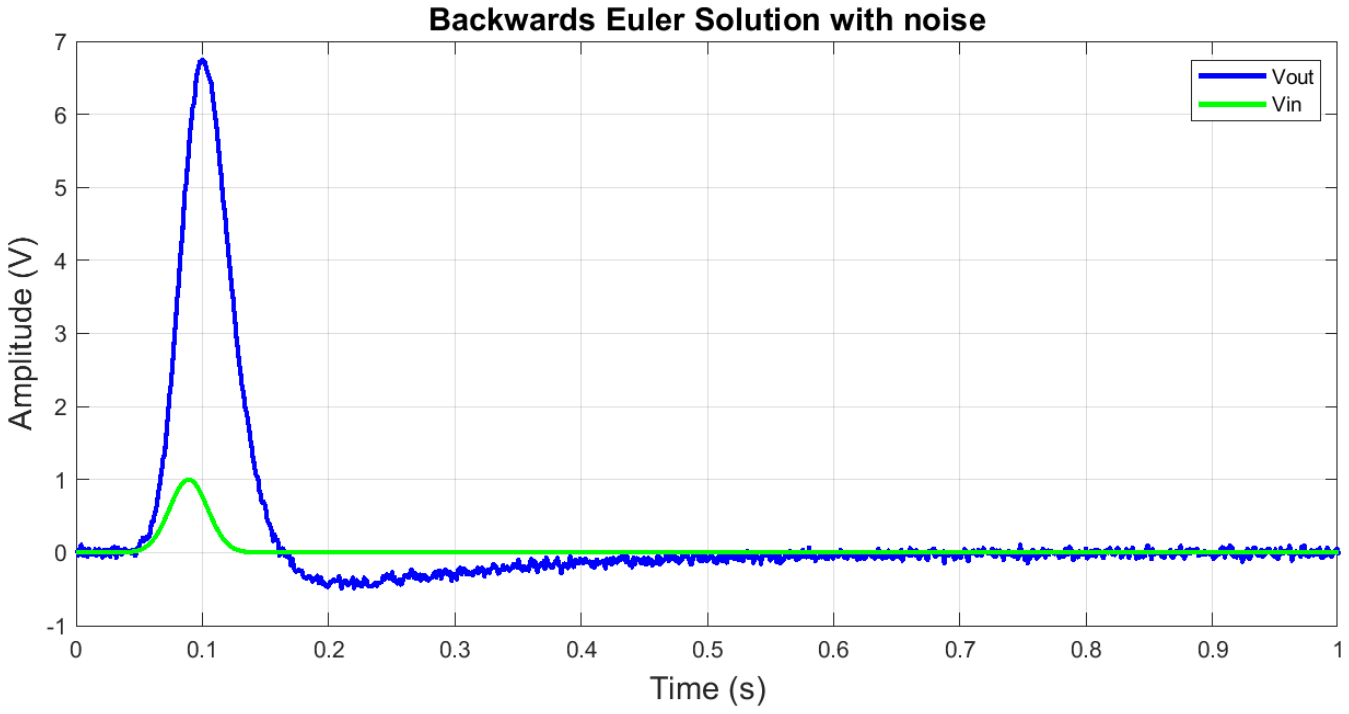
Sinusoidal

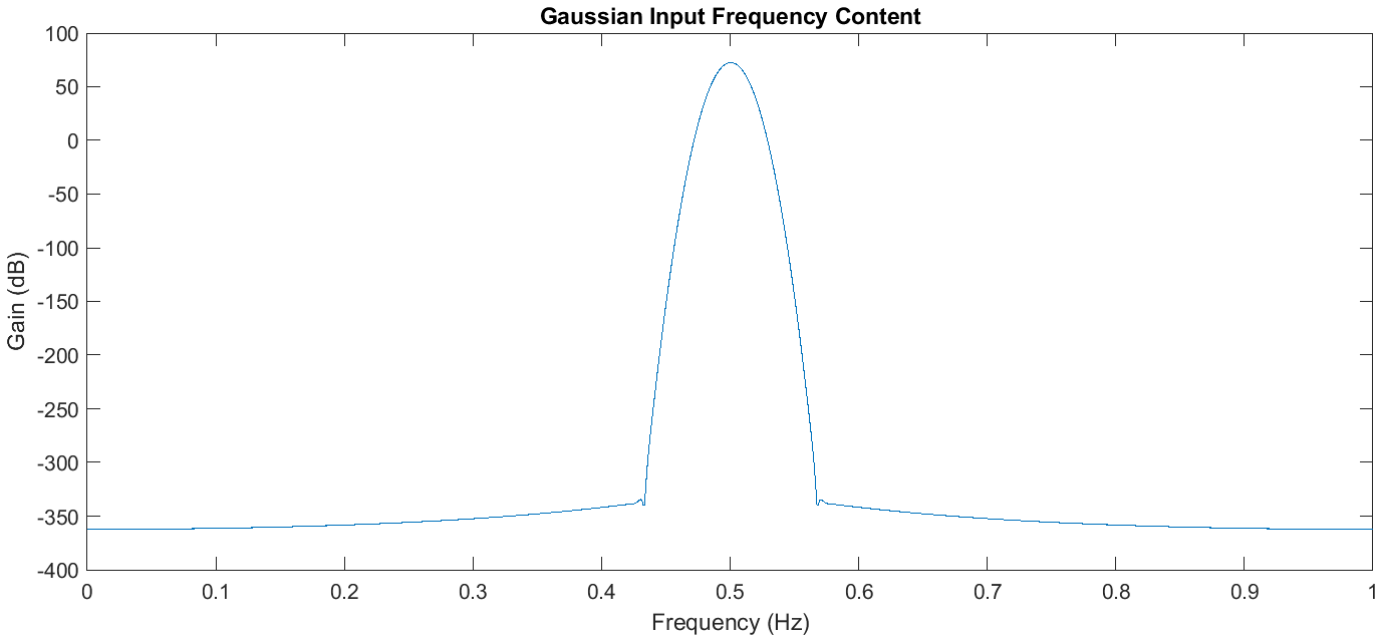

Backwards Euler Solution with noise
Vout
Vin
Amplitude (V)
Time (s)
4
3
2
1
0
-1
-2
-3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

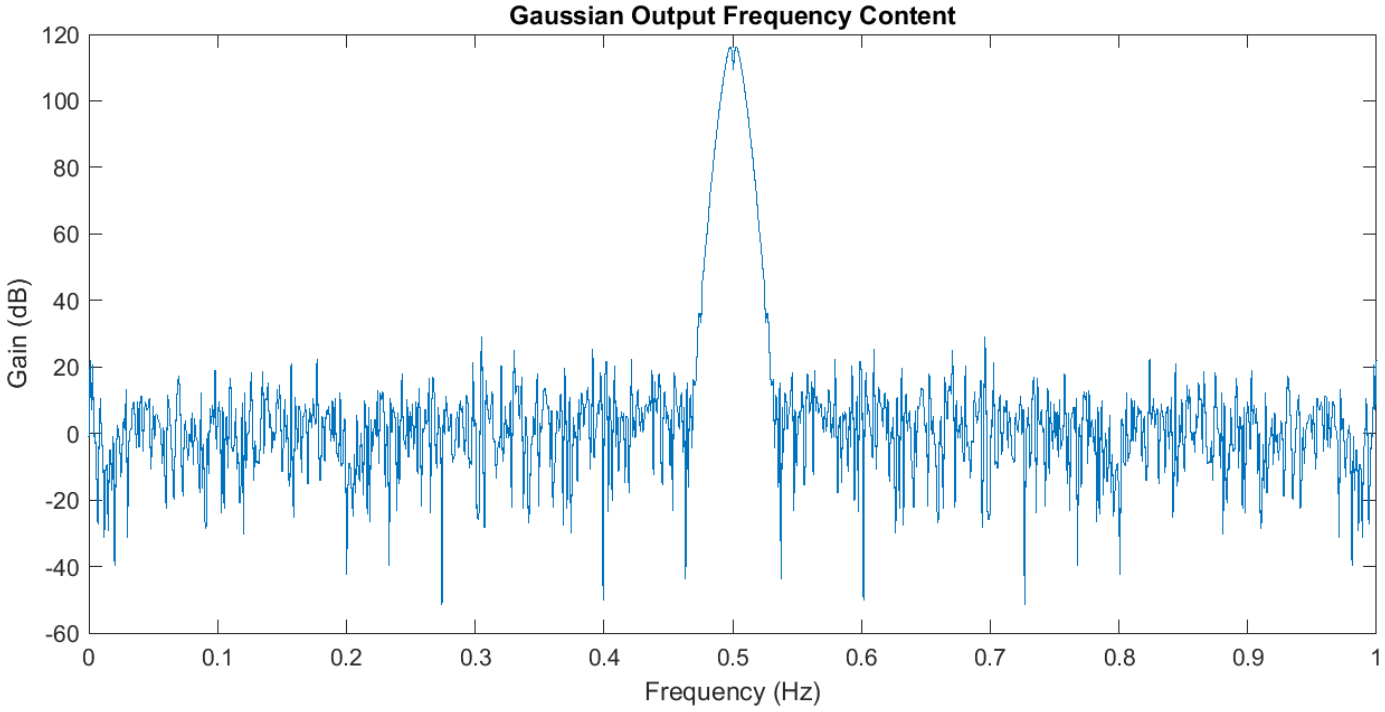

Sinusoidal Input Frequency Content
Gain (dB)
Frequency (Hz)
140
120
100
80
60
40
20
0
-20
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Sinusoidal Output Frequency Content
Gain (dB)
Frequency (Hz)
150
100
50
0
-50
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

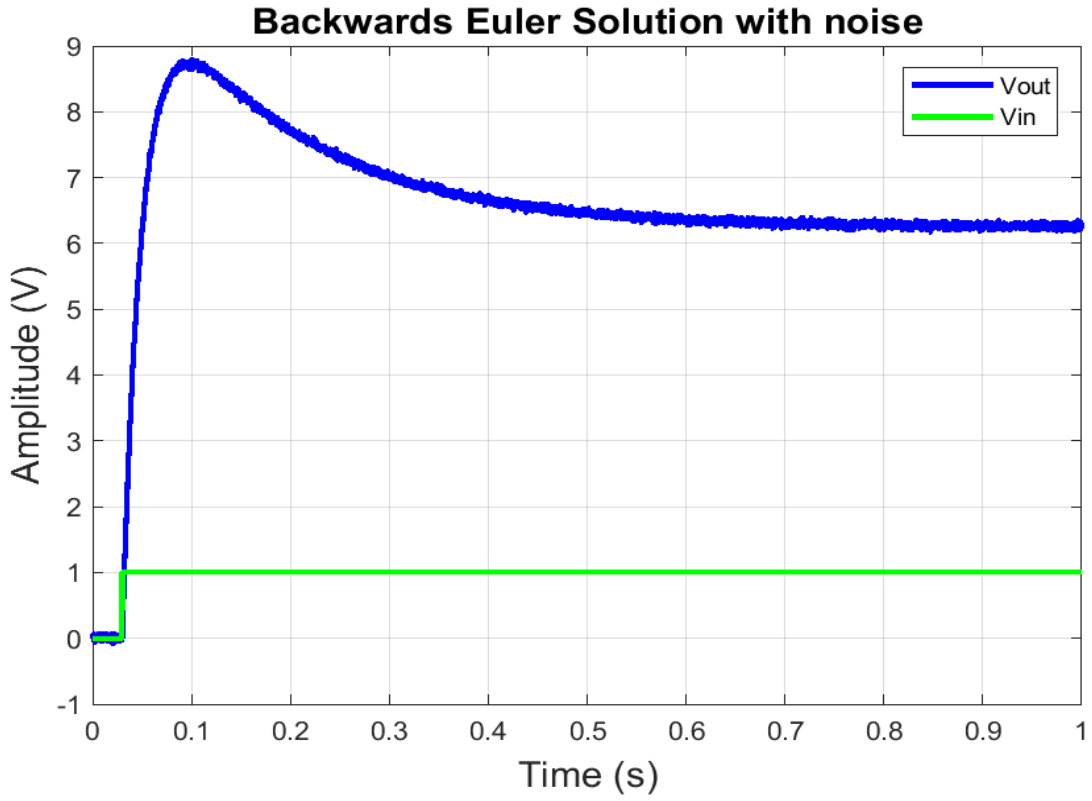
Gaussian

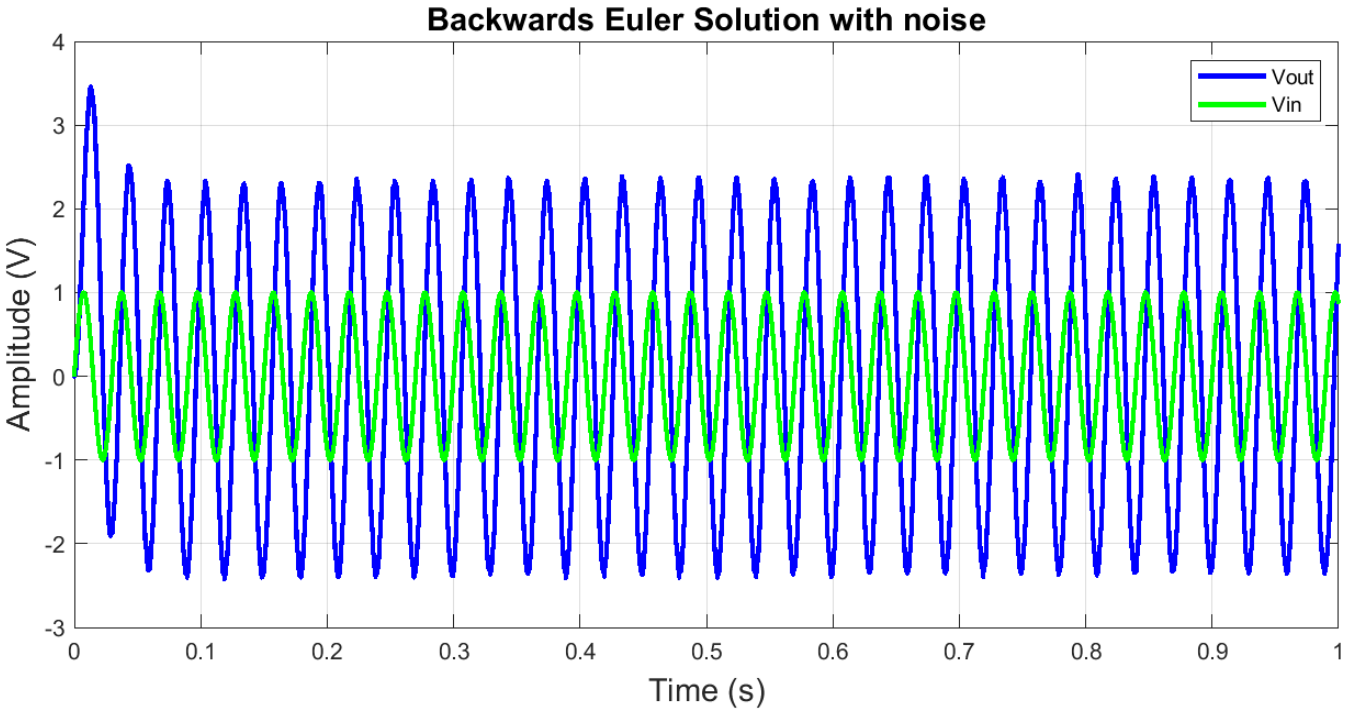

Backwards Euler Solution with noise
Amplitude (V)
Time (s)
Vout
Vin
7
6
5
4
3
2
1
0
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Gaussian Input Frequency Content
Gain (dB)
Frequency (Hz)
100
50
0
-50
-100
-150
-200
-250
-300
-350
-400
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Gaussian Output Frequency Content
Gain (dB)
Frequency (Hz)
120
100
80
60
40
20
0
-20
-40
-60
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

The number of steps was now increased from 1000 to 10000. The following plots of step and sinusoidal input with increased steps are shown below.





As can be seen from the figures above, an increase in the number of steps helps to mitigate some of the effects of noise.

Question 6

Part A

If the output was instead modelled as $V=\alpha I_3+\beta I_3^2+\gamma I_3^3$ then the input into the stamp function would need to change based on a rearranged version of the above equation. This would also make the circuit nonlinear, meaning another vector would need to be introduced to handle this. This new nonlinear vector would account for the change but would also require an adaptation of some of the equations, specifically the backwards Euler implementation used to solve for the transient simulations.

Part B

The new Euler equation would change from

```
X(:,i) = ((C/h) + G) \ ((C/h)*X(:,i-1) + b(:,1));
```

to

```
X(:,i) = ((C/h) + G) \ ((C/h)*X(:,i-1) + b(:,1) -F(x(:,1));
```

Where F represents the new nonlinear vector.