Table of Contents


```
clc
clear
close all
```

ELEC 4700 -- Assignment #4 Submission

Name: Kian Molani, Student Number: 101039806

Part 01 & 02 -- Simulating Current and Resistance Through Bottle-Neck

In this part of the assignment, the solutions from assignment #2/3 will be used to simulate the current through some bottle-neck region, which will then be used to derive the resistance of our device. It should be noted that at the end of assignment #3, the simulated flow of electrons through our bottle-neck was used to derive the current density throughout our region. The results of this simulation are not the ones used here -- instead, as done in assignment #2, the finite difference method will be used to derive various properties of our region, including a spatial map of current densities. Current can be then be extracted from this map by taking the sum of all current densities throughout our region. This will be done for various values of voltage applied across our device, from which a plot of current vs. applied voltage can be produced. Using Ohm's law, the resistance of our device can then be calculated:

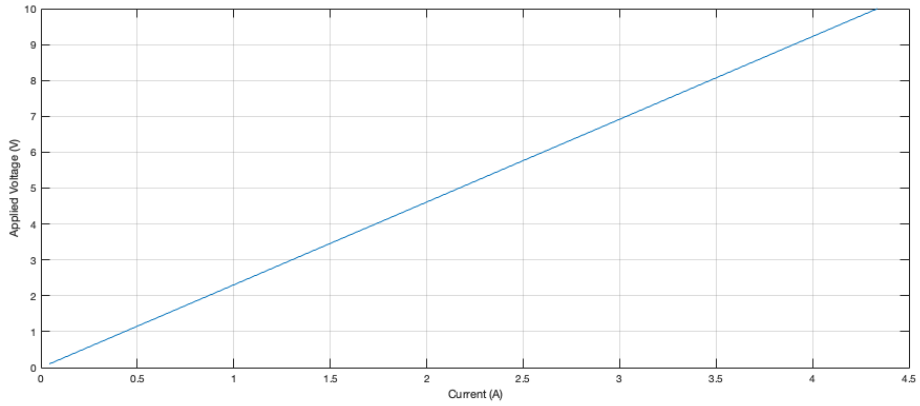
$$R = \frac{V}{I}$$

The value obtained from this simulation will be substituted in for R_3 of our electrical circuit.

```
% initialize parameters

V_applied = linspace(0.1,10,100);
nx = 100; ny = 100;
L_b = 0.2*nx; W_b = 0.4*ny;
sigma_inside = 1e-2; % conductivity inside boxes
sigma_outside = 1; % conductivity outside boxes
bottle_neck_width = 1; % default value (listed as a factor)

% compute currents
```

```
for i = 1:length(V_applied)
    [~,~,~,~,~,~,I(i)] =
 compute_params(nx,ny,L_b,W_b,sigma_inside,sigma_outside,bottle_neck_width,V_appli
end

% plot results and obtain slope

figure;
plot(I,V_applied);
xlabel('Current (A)'); ylabel('Applied Voltage (V)');
grid on;
p = polyfit(I,V_applied,1)
```

```
p =

    2.3060    0.0000
```



The value for the slope of the plot above was obtained from the polyfit function for order 1, and is equal to 2.306. That is, the value for the resistance R_3 that will be used for our simulated electrical circuit is:

$$R_3 = 2.306\Omega$$

Since this value appears to be a little low, we will instead use a resistance value of 10 Ohms.

Part 03 -- Electrical Circuit Simulation: DC and AC Sweeps

In this part of the assignment, we take a provided electrical circuit, describe it via a set of KCL equations, and then use these equations to define C and G matrices for simulation. The KCL/nodal expressions derived across the five nodes of our electrical circuit are listed below:

$$V_1 = V_{in}$$

$$G_1(V_2 - V_1) + \frac{d(V_2 - V_1)}{dt}C - I_E = 0$$

$$G_1(V_2 - V_1) + \frac{d(V_2 - V_1)}{dt}C + G_2V_2 - I_L = 0$$

$$G_3V_3 - I_L = 0$$

$$G_4(V_4 - V_o) + I_\alpha = 0$$

$$V_4 - \alpha G_3V_3 = 0$$

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

$$I_L = \frac{V_2 - V_3}{j\omega L}$$

Our frequency domain expressions are:

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

$$G_1(V_2 - V_1) + j\omega C(V_2 - V_1) + G_2V_2 - \frac{V_2 - V_3}{j\omega L} = 0$$

$$G_3V_3 - \frac{V_2 - V_3}{j\omega L} = 0$$

Again, we can use these equations to derive our C and G matrices for circuit simulation. We will do this by looking at the coefficients in front our voltage terms, and in the case of the C-matrix, we will use those coefficients attached to our time-derivatives. We will first work to define basic basic paramaters:

```
% define circuit parameters

R1 = 1;
R2 = 2;
R3 = p(1); R3 = 10; % from part 01
R4 = 0.1;
Ro = 1000;
C1 = 0.25;
L1 = 0.2;
alpha = 100;

% define conductances and C and G matrices

G1 = 1/R1; G2 = 1/R2; G3 = 1/R3; G4 = 1/R4; Go = 1/Ro;
```

Our C and G matrices come out to be:

```
C = [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 L1]

G = [1 0 0 0 0 0;
```

```
    (-1/R1) ((1/R2)+(1/R1)) 0 0 0 0;
    0 0 (1/R3) 0 0 -1;
    0 0 (-1*alpha/R3) 1 0 0;
    0 0 0 (-1/R4) ((1/R4)+(1/Ro)) 0;
    0 -1 1 0 0 0]
```

```
C =

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

```
G =

    1.0000         0         0         0         0         0
   -1.0000    1.5000         0         0         0         0
         0         0    0.1000         0         0   -1.0000
         0         0  -10.0000    1.0000         0         0
         0         0         0  -10.0000   10.0010         0
         0   -1.0000    1.0000         0         0         0
```

The first simulation that we performed was a DC simulation, where we swept our input voltage from minus 10V to plus 10V and then plotted the resulting output voltage and voltage at node 3 of our electrical circuit. The matrix yielding nodal voltages is extracted from the following equation:

$$V = (G + j\omega C) \backslash F$$

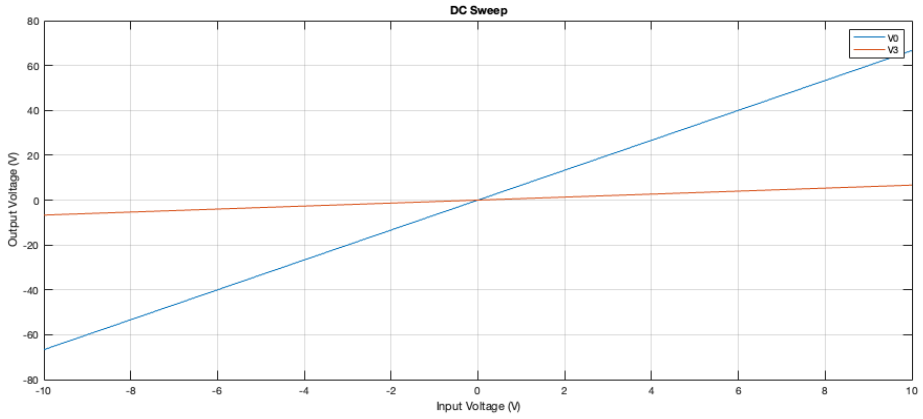
Since this is a DC simulation, our value for ω is zero, which implies that our C-matrix will not be used. Thus, our matrix V becomes:

$$V = G \backslash F$$

To extract our output voltage and voltage at node 3, we will use the element V(5) and V(3) from this matrix, respectively.

```
% perform DC sweep

Vin = [-10:0.1:10];

for i=1:length(Vin)
    F = [Vin(i); 0; 0; 0; 0; 0];
    V = G\F;
    Vo(i) = V(5);
    V3(i) = V(3);
end

figure;
plot(Vin,Vo);
hold on;
```

```
plot(Vin,V3);
legend('V0','V3');
title('DC Sweep');
xlabel('Input Voltage (V)');
ylabel('Output Voltage (V)');
grid on;
```



The next simulation that will be performed is an AC sweep to get a frequency response of the gains for our circuit. To do this, we will sweep between frequencies from 0 to 50Hz, calculate angular frequency, and solve our nodal voltages using the equation:

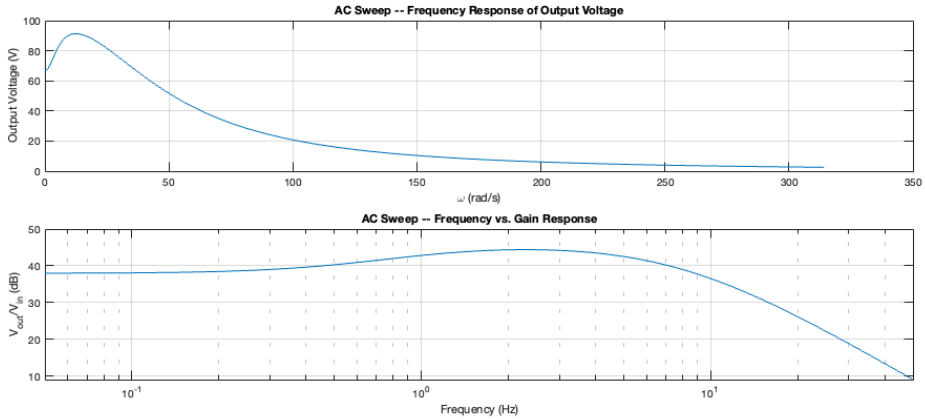
$$V = (G + j\omega C) \backslash F$$

Here, the C-matrix will be used. Note that for our simulation, we have set the input voltage to 10V.

```
Vin = 10;
F = [10;0;0;0;0;0];
freq = linspace(0,50,1000);
w = 2 * pi * freq;
Vo = 0; Vo(1:length(w)) = 0;

for i = 1:length(w)
    V = (G+j*w(i)*C)\F;
    Vo(i) = V(5);
end

Gain = Vo./Vin;

figure;
subplot(2,1,1);
plot(w,Vo);
xlabel('\omega (rad/s)'); ylabel('Output Voltage (V)');
title('AC Sweep -- Frequency Response of Output Voltage');
grid on;
subplot(2,1,2);
semilogx(freq,20*log(abs(Gain)));
xlabel('Frequency (Hz)'); ylabel('V_{out}/V_{in} (dB)');
title('AC Sweep -- Frequency vs. Gain Response');
grid on;
```

```
xlim([-inf inf])
```

Warning: Imaginary parts of complex X and/or Y arguments ignored.

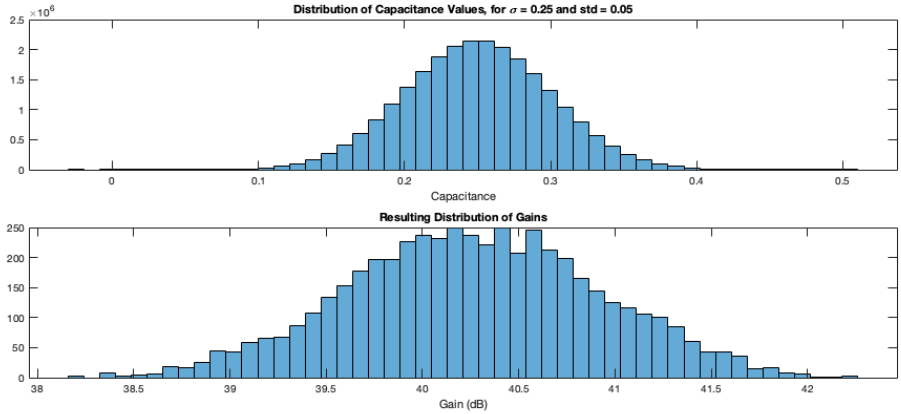


As can be seen from this result, our output voltages are attenuated at higher frequencies -- our electrical circuit acts as a low pass filter that can be characterized with a -3dB drop-off point.

Our final set of results will be to simulation random perturbations in our capacitance value C. To do this, we will model our capacitance C as a normal distribution with mean 0.25 and standard deviation 0.05. By sampling from this distribution, we can then re-compute the gain value from our electrical circuit for some constant frequency (in this case, the value $\omega = \pi$ will be used), and see how variations in capacitance effects the output voltage of our circuit.

```
C1 = normrnd(0.25,0.05,5000);
figure;
subplot(2,1,1)
histogram(C1,50)
xlabel('Capacitance');
title('Distribution of Capacitance Values, for \sigma = 0.25 and std =
 0.05')

w = pi;
Vo(1:length(C1)) = 0;
for i = 1:length(C1)
    C1_temp = C1(i);

    C = [0 0 0 0 0 0;
    -C1_temp C1_temp 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];

    V = (G+j*w.*C)\F;
    Vo(i) = V(5);
end

Gain = Vo./Vin; Gain = 20*log(abs(Gain));

subplot(2,1,2)
```

```
histogram(Gain,50);
xlabel('Gain (dB)');
title('Resulting Distribution of Gains')
```



We can see from our simulation results that real capacitance values can vary quite significantly from their listed nominal values, and that the resulting output voltage swing is also non-negligible (the gains encapsulated from plus or minus two standard deviations varies from about 68 to 70). Though this may be tolerable for general purpose applications, this amount of variation of output voltage gain might be unacceptable for some.

Part 04 -- Transient Circuit Simulation

In part 04 of this assignment, we will perform transient analysis of our circuit. From inspection of our previous work, we can see that our circuit is an amplifier. From the circuit's frequency response, we can also see that the circuit acts as a low pass filter, where signals are amplified in a relatively stable manner over a range of frequencies, after which point the magnitude of our signal drops substantially for higher frequencies.

In order to perform nodal analysis of our circuit in the time domain, we will employ Backwards-Euler finite difference method. In this method, we will iterate and calculate new solutions for our matrix at each time interval, and will make use of values calculated from previous iterations. We start with the following equation:

$$C\frac{dV(t)}{dt} + GV(t) = F(t)$$

From which we may perform the following manipulations:

$$C(\frac{V_i - V_{i-1}}{dt}) + GV_i(t) = F(t)$$

$$V_i(\frac{C}{dt} + G) = F + V_{i-1}(\frac{C}{dt})$$

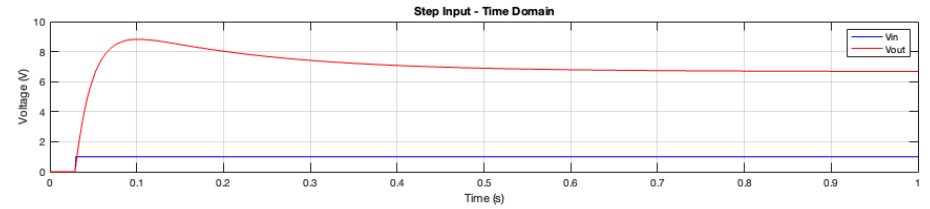
$$V_i = (\frac{C}{dt} + G)\backslash(F + V_{i-1}(\frac{C}{dt}))$$

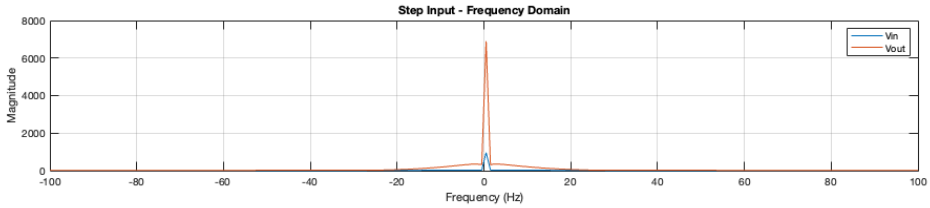
We will perform our transient simulation for 4 different voltage inputs: a step input, two sinusoidal inputs with different frequencies, and a Gaussian pulse.

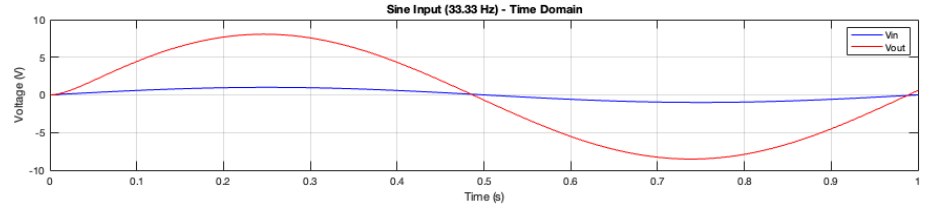
```
% re-initialize circuit parameters

C1 = 0.25;

C = [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 L1];

F = [1;0;0;0;0;0];

V = [0;0;0;0;0;0];

number_of_steps = 1000;
dt = 1/number_of_steps;

% simulate for step input

figure;
subplot(2,1,1)
[Vin,Vout] = bwd_euler("step",number_of_steps,dt,C,G,F,V,4);
xlabel('Time (s)'); ylabel('Voltage (V)'); legend('Vin','Vout');
title('Step Input - Time Domain');
grid on;

subplot(2,1,2)
x = linspace(-length(Vin)/2,length(Vin)/2,length(Vin)); y =
 abs(fftshift(fft(Vin)));
plot(x,y);
hold on
x = linspace(-length(Vout)/2,length(Vout)/2,length(Vout)); y =
 abs(fftshift(fft(Vout)));
plot(x,y);
xlabel('Frequency (Hz)')
ylabel('Magnitude')
grid on;
title('Step Input - Frequency Domain');
legend('Vin','Vout')
xlim([-100 100])

% simulate for sinusoidal input (33.333 Hz)

figure;
subplot(2,1,1)
[Vin,Vout] = bwd_euler("sine01",number_of_steps,dt,C,G,F,V,4);
xlabel('Time (s)'); ylabel('Voltage (V)'); legend('Vin','Vout');
title('Sine Input (33.33 Hz) - Time Domain');
grid on;

subplot(2,1,2)
x = linspace(-length(Vin)/2,length(Vin)/2,length(Vin)); y =
 abs(fftshift(fft(Vin)));
```

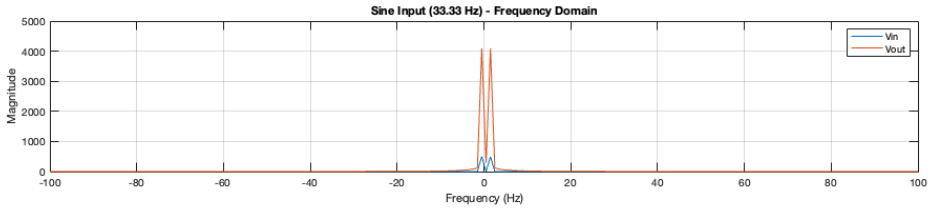
```
plot(x,y);
hold on
x = linspace(-length(Vout)/2,length(Vout)/2,length(Vout)); y =
 abs(fftshift(fft(Vout)));
plot(x,y);
xlabel('Frequency (Hz)')
ylabel('Magnitude')
legend('Vin','Vout')
title('Sine Input (33.33 Hz) - Frequency Domain');
grid on;
xlim([-100 100])

% simulate for sinusoidal input (5 Hz)

figure;
subplot(2,1,1)
[Vin,Vout] = bwd_euler("sine02",number_of_steps,dt,C,G,F,V,4);
xlabel('Time (s)'); ylabel('Voltage (V)'); legend('Vin','Vout');
grid on;
title('Sine Input (5 Hz) - Time Domain');

subplot(2,1,2)
x = linspace(-length(Vin)/2,length(Vin)/2,length(Vin)); y =
 abs(fftshift(fft(Vin)));
plot(x,y);
hold on
x = linspace(-length(Vout)/2,length(Vout)/2,length(Vout)); y =
 abs(fftshift(fft(Vout)));
plot(x,y);
xlabel('Frequency (Hz)')
ylabel('Magnitude')
legend('Vin','Vout')
title('Sine Input (5 Hz) - Frequency Domain');
grid on
xlim([-100 100])

% simulate for Gaussian input

figure;
subplot(2,1,1)
[Vin,Vout] = bwd_euler("gaussian",number_of_steps,dt,C,G,F,V,4);
xlabel('Time (s)'); ylabel('Voltage (V)'); legend('Vin','Vout');
title('Gaussian Input - Time Domain');
grid on;

subplot(2,1,2)
x = linspace(-length(Vin)/2,length(Vin)/2,length(Vin)); y =
 abs(fftshift(fft(Vin)));
plot(x,y);
hold on
x = linspace(-length(Vout)/2,length(Vout)/2,length(Vout)); y =
 abs(fftshift(fft(Vout)));
plot(x,y);
xlabel('Frequency (Hz)')
```

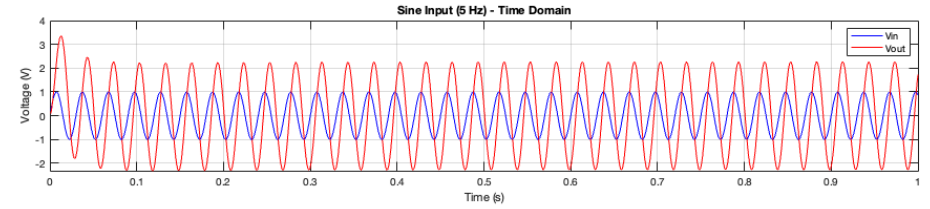
```
ylabel('Magnitude')
legend('Vin','Vout')
title('Gaussian Input - Frequency Domain');
grid on
xlim([-100 100])
```

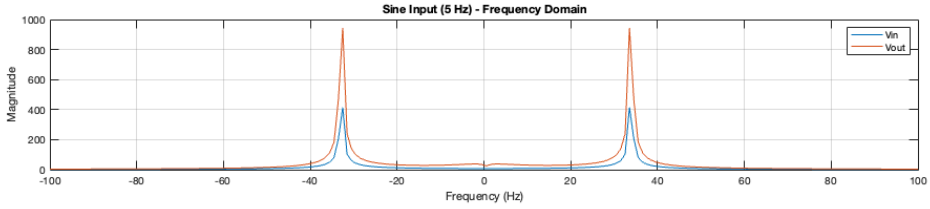


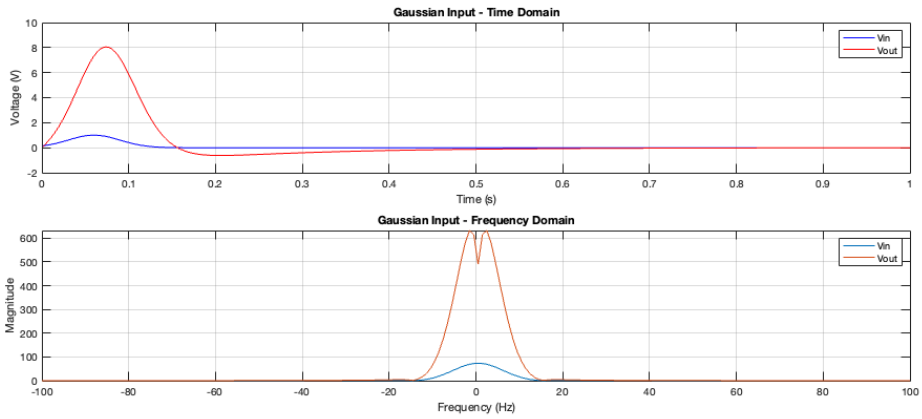












We can see from our FFTs for the sinusoidal inputs that the peaks at the FFTs occur at +/- the frequency of the input sinusoid, as expected. And so, for a sinusoid with larger frequency, those peaks also occur at larger frequencies. We also see that for the 5Hz wave, the amplitude of the output signal is much higher. This is because as we saw from the frequency response curve for our circuit, at 33.33 Hz the output signal has a lower gain value.

Increasing our number of points/decreasing our timestep increases the accuracy of our simulation and produced plots.

Part 05 -- Transient Circuit Simulation w/ Added Noise

In this part of the assignment, we will include thermal noise in R3 in our transient circuit simulations. We will model this behaviour by introducing a capacitor and current source in parallel with R3 -- the current source will be random, and so will serve to model the noise, while the capacitor will serve to alter the bandwidth of this noise component. Within our source code, these changes will manifest themselves through changes in our C-matrix and F-matrix. The C-matrix used for our simulations will now be:

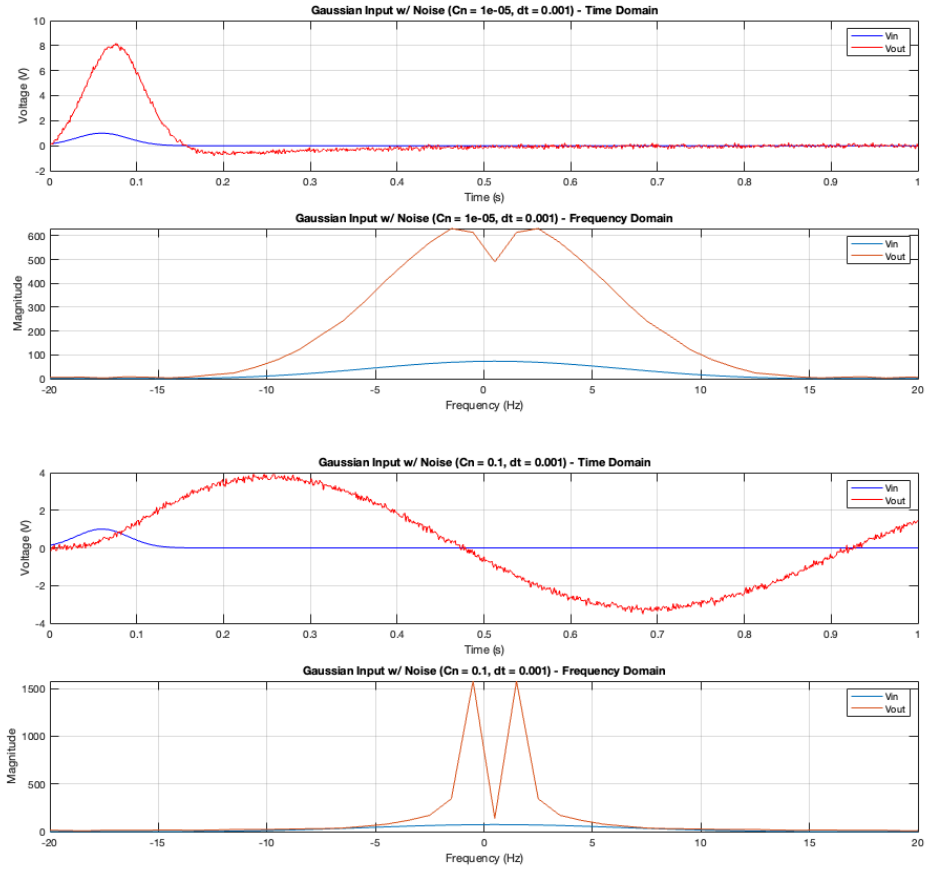
```
C = ["0" "0" "0" "0" "0" "0";
"-C1" "C1" "0" "0" "0" "0";
"0" "0" "Cn" "0" "0" "0";
"0" "0" "0" "0" "0" "0";
"0" "0" "0" "0" "0" "0";
"0" "0" "0" "0" "0" "L1"]
```

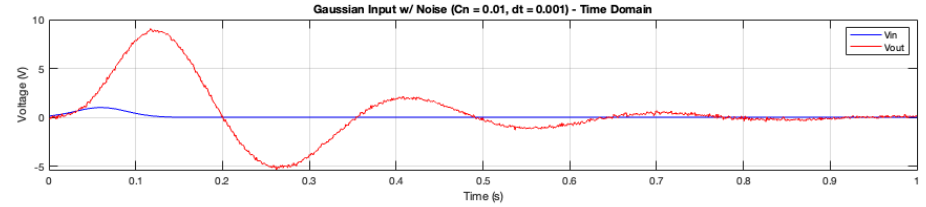
```
C =

  6×6 string array

    "0"     "0"     "0"     "0"     "0"     "0"
    "-C1"   "C1"    "0"     "0"     "0"     "0"
    "0"     "0"     "Cn"    "0"     "0"     "0"
    "0"     "0"     "0"     "0"     "0"     "0"
    "0"     "0"     "0"     "0"     "0"     "0"
    "0"     "0"     "0"     "0"     "0"     "L1"
```

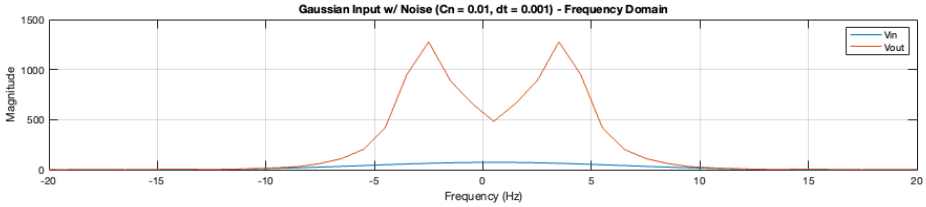
The current source will be derived by randomly sampling from a Gaussian distribution centered at 0 and multiplying this distribution by a scale factor of 0.001. This value will be fed as input into our F-matrix. The input for our electrical circuit will be a Gaussian pulse.

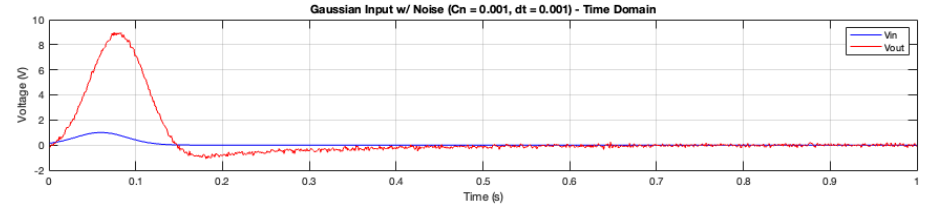
```
% define parameters

Cn = [0.00001 0.1 0.01 0.001 0.0001 0.000001];
number_of_steps = [1000 500 2000];
dt = [1/number_of_steps(1) 1/number_of_steps(2) 1/number_of_steps(3)];

for i = 1:(length(Cn)+length(number_of_steps))-1

    if i >= length(Cn)+1
        Cn_temp = Cn(1);

        C = [0 0 0 0 0 0;
            -C1 C1 0 0 0 0;
            0 0 Cn(1) 0 0 0;
            0 0 0 0 0 0;
            0 0 0 0 0 0;
            0 0 0 0 0 L1];

        number_of_steps_temp = number_of_steps(i-length(Cn)+1);
        dt_temp = dt(i-length(Cn)+1);
    else
        Cn_temp = Cn(i);

        C = [0 0 0 0 0 0;
            -C1 C1 0 0 0 0;
            0 0 Cn(i) 0 0 0;
            0 0 0 0 0 0;
            0 0 0 0 0 0;
            0 0 0 0 0 L1];

        number_of_steps_temp = number_of_steps(1);
        dt_temp = dt(1);
    end

    F = [1;0;0;0;0;0];

    V = [0;0;0;0;0;0];

    figure;
    subplot(2,1,1)
    [Vin,Vout] =
 bwd_euler("gaussian",number_of_steps_temp,dt_temp,C,G,F,V,5);
    xlabel('Time (s)'); ylabel('Voltage (V)'); legend('Vin','Vout');
    title(strcat("Gaussian Input w/ Noise ","(Cn =
 ",string(Cn_temp),", dt = ",string(dt_temp),") - Time Domain"));
    grid on;

    subplot(2,1,2)
    x = linspace(-length(Vin)/2,length(Vin)/2,length(Vin)); y =
 abs(fftshift(fft(Vin)));
```

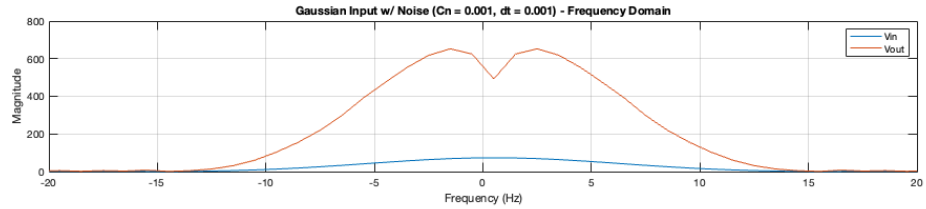
```
    plot(x,y);
    hold on
    x = linspace(-length(Vout)/2,length(Vout)/2,length(Vout)); y =
 abs(fftshift(fft(Vout)));
    plot(x,y);
    xlabel('Frequency (Hz)')
    ylabel('Magnitude')
    legend('Vin','Vout')
    title(strcat("Gaussian Input w/ Noise ","(Cn =
 ",string(Cn_temp),", dt = ",string(dt_temp),") - Frequency Domain"));
    grid on;
    xlim([-20 20]);

end
```

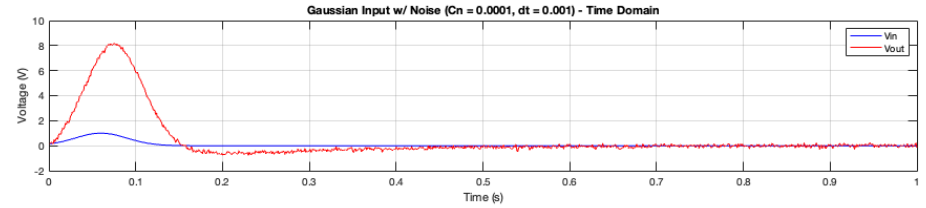


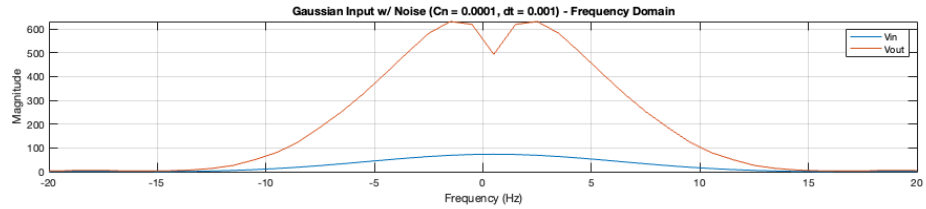

Gaussian Input w/ Noise (Cn = 0.01, dt = 0.001) - Time Domain
Voltage (V)
Time (s)
Vin
Vout

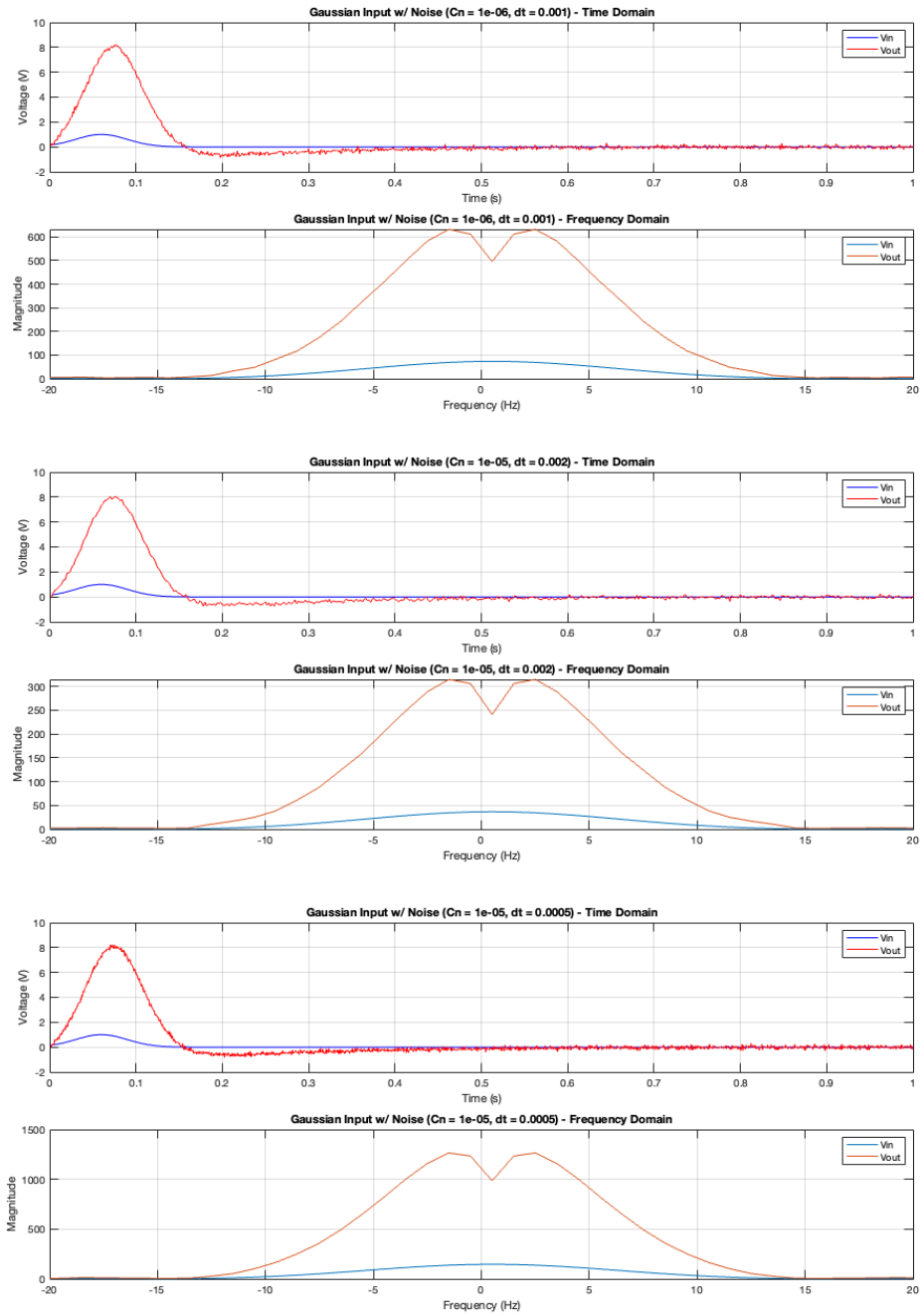

Gaussian Input w/ Noise (Cn = 0.01, dt = 0.001) - Frequency Domain
Magnitude
Frequency (Hz)
Vin
Vout


Gaussian Input w/ Noise (Cn = 0.001, dt = 0.001) - Time Domain
Voltage (V)
Time (s)
Vin
Vout


Gaussian Input w/ Noise (Cn = 0.001, dt = 0.001) - Frequency Domain
Magnitude
Frequency (Hz)
Vin
Vout


Gaussian Input w/ Noise (Cn = 0.0001, dt = 0.001) - Time Domain
Voltage (V)
Time (s)
Vin
Vout


Gaussian Input w/ Noise (Cn = 0.0001, dt = 0.001) - Frequency Domain
Magnitude
Frequency (Hz)
Vin
Vout



As can be seen from the plots above, we have ran transient simulations of our circuit for various values of Cn and timesteps used. We can see from these results that for larger Cn, the bandwidth of our signal increases. The consequence of this is that higher frequencies introduces instablity in our signal and prevents it from stabilizing towards a steady-state value. When we decrease our value for Cn, our bandwidth and oscillations in our signal are reduced and we are able to re-produce our Gaussian pulse.

We can also see from our plots that for smaller timesteps, the accuracy of our simulated results increase (notice the higher resolution of our noise component as our timestep dt gets smaller).

Part 06 -- Accounting for Non-Linearities

In this part of the assignment, we describe how we may account for non-linearities in our circuit by replacing our nodal equation $V = \alpha I_3$ to $V = \alpha I_3 + \beta I_3^2 + \gamma I_3^3$. We could capture all the information pertaining to these non-linearities in a separate B-matrix:

$$C\frac{dV}{dt} + GV + B = F$$

Applying Backwards-Euler finite difference method to this expression yields the following:

$$V_i = (\frac{C}{dt} + G)\backslash(F + V_{i-1}(\frac{C}{dt}) - B)$$

We can write our B-matrix as:

$$B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -\beta^{1/2} - \gamma^{1/3} \\ 0 \\ 0 \end{pmatrix}$$

From here, we can choose to derive values for our B-matrix from various sources of non-linearities (ex: electronic circuit elements that are described by non-linear expressions and relationships).

Published with MATLAB® R2020b