Contents

Assignment 4

```
Circuit Modeling

Author: Ragini Bakshi, April 2021
```

```
set(0,'DefaultFigureWindowStyle','docked')
set(0, 'defaultaxesfontsize', 12)
set(0, 'defaultaxesfontname', 'Times New Roman')
set(0, 'DefaultLineLineWidth',2);

clear all
close all
```

1. Modelling the Current-Voltage Characteristics & Linear fit to determine the resistance value of the device and use it for R3

using fixed bottleneck and Voltage sweep from 0.1V to 10V

```
steps_for_voltage = 50;
stepped_voltage = linspace(0.1, 10, steps_for_voltage);
for i=1:steps_for_voltage
    [Curr, Vmap, Ex, Ey, eFlowx, eFlowy, Cnx] = current_solver(stepped_voltage(i)); % calls current_solver function from assignment 2
    plotting_current(i) = Cnx;
end

% figure(1)
% plot(plotting_current, stepped_voltage)
r_inv = polyfit(stepped_voltage, plotting_current,1);
r_fitted = 10/r_inv(1); % note: im afraid this is not complete: I ran out of time
%so I've simply obtained the current from the electric field, not the number of particles passing the
%right boundary of the active region and subtracting the ones passing backwards through the left, waiting
%for the simulation to stabilize and then summing the average to get the
%current (involved implementing assignment 3 but its giving an error message so I've left it out).
```

Defining Circuit Parameters

The capacitor and inductor in the circuit give it a Bandpass-like filter. There are no non-linear capacitors

```
R1 = 1;
C_1 = 0.25;
R2 = 2;
L = 0.2;
R3 = r_fitted;                          % changed this from 10ohms to fitted value using assignment 2
alpha = 100;
R4 = 0.1;
Ro = 1000;

% DC sweep for 100 Vin values
Vin = -10:0.1:10;
%Vin = Vin';
```

3. Breaking the circuit down into Differential Equations

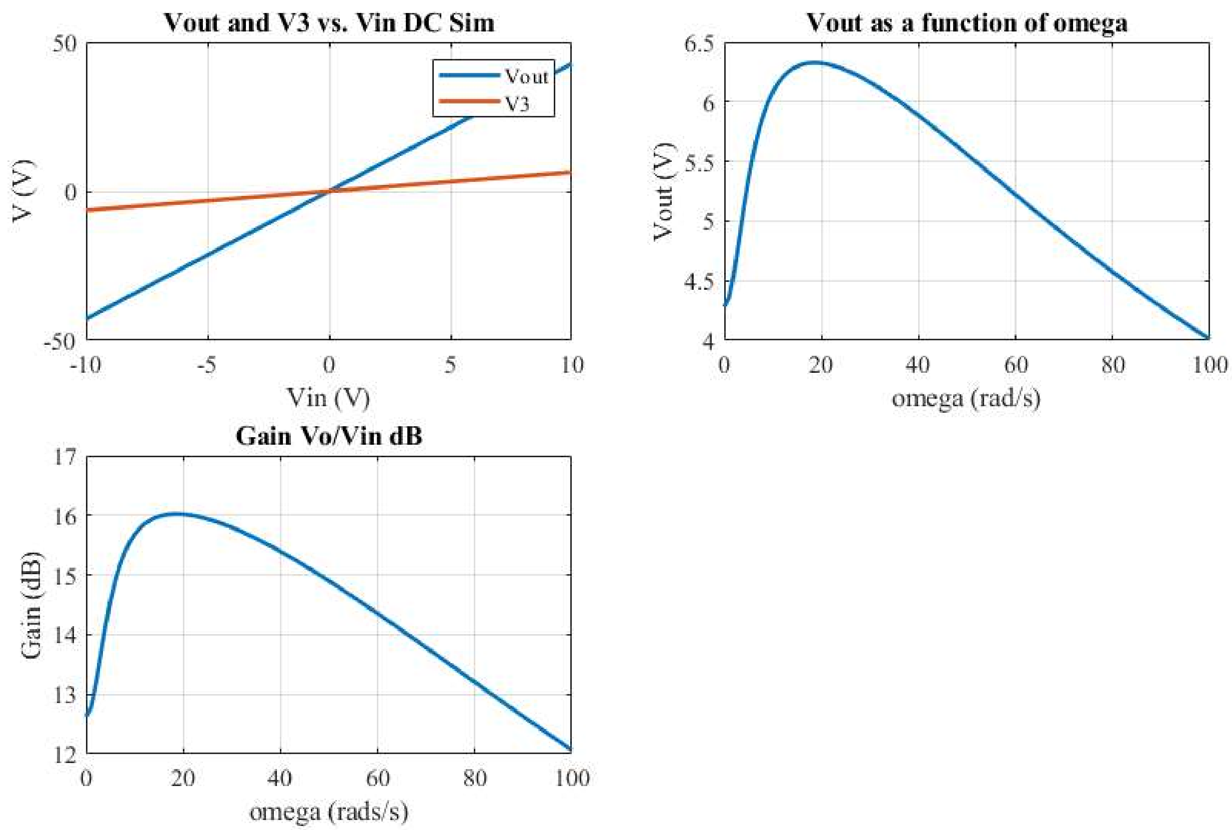
3.a.i. Assuming V1 is the current that flows thru Node 1, V2 is the current that flows thru Node 2, Iin is the current going into the circuit, Il is the current thru inductor, etc, the differential equations become:

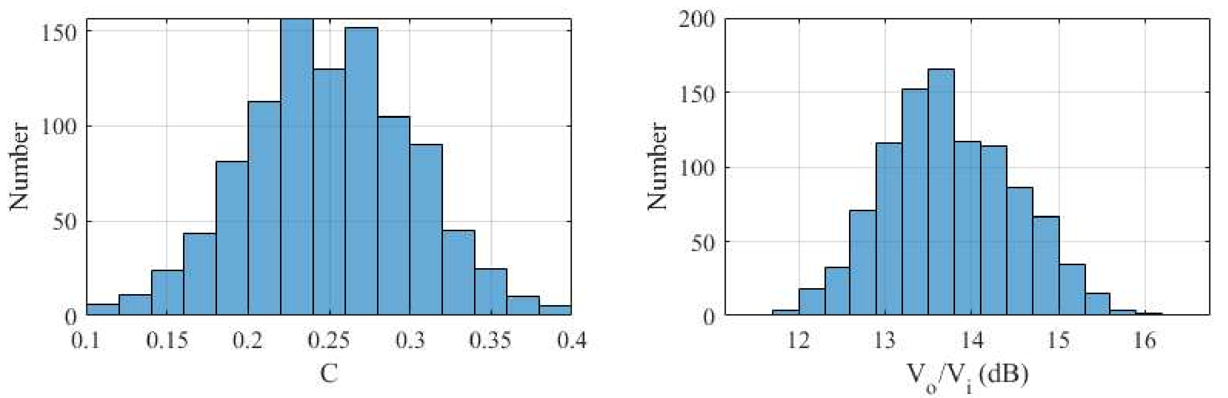
```
G1 = 1/R1;
G2 = 1/R2;
G3 = 1/R3;
G4 = 1/R4;
Go = 1/Ro;

% V1 = Vin;

% KCL equations for this network:
% G1(V1 - V2) + sC_1(V1 - V2) + Iin = 0;
% G1(V2 - V1) + sC_1(V2 - V1) + V2*G2 + Il = 0;
% V2 - V3 -sL = 0
% G3*V3 - Il = 0
% V4 - G3*alpha*V3 = 0
% G4*(V4 - V5) + I_alpha = 0
% G4*(V5 - V4) + V5*Go = 0

% These equations were used to construct the G conductance matrix and the C
% capacitance matrix. The F vector is for the source.
G = zeros(6,6);
C = zeros(6,6);
F = zeros(6,1);

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) = -G3*alpha;
G(4,4) = 1;
G(5,5) = -(G4+Go);
G(5,4) = G4;
G(6,2) = -1;
G(6,3) = 1;

C(2,1) = C_1;
C(2,2) = -C_1;
C(6,6) = L;

index = 0;
for Vin = linspace(-10,10,100)
   index = index + 1;
   F(1) = Vin;
   V = G\F;
   Vout(index) = V(5);
   V_3(index) = V(3);
   Vin_vector(index) = Vin;
end

% 3.a.ii. Converting to frequency domain requires the taking the time derivative
% which can then be solved for any omega
index = 0;
F(1) = 1;
for w = linspace(0,100,100)
    index = index + 1;
    omega(index) = w;
    V_ac = (G + 1j*omega(index).*C)\F;
    V_out_ac(index) = V_ac(5);
    gain(index) = 20*log10(abs(V_ac(5))/F(1));
end

% 3.a.iii. Writing down the matrices used to describe the network
display('C matrix'); C
display('G matrix'); G
display('F vector'); F

% 3.b.i. Plot 1: Vout vs Vin DC simulation (circuit has some visible gain)
figure
subplot(3,2,1)
plot(Vin_vector, Vout);
title('Vout and V3 vs. Vin DC Sim');
xlabel('Vin (V)');
ylabel('V (V)');
hold on;
plot(Vin_vector, V_3);
legend('Vout', 'V3')
grid on;
```

```
% 3.b.ii. Plot 2: Voltage as a function of omega (going from 0 to 100) with a peak
% at approx 18 showing Bandpass filter-like behaviour
subplot(3,2,2)
plot(omega, abs(V_out_ac));
title('Vout as a function of omega');
xlabel('omega (rad/s)');
ylabel('Vout (V)');
grid on;

% 3.b.ii. Plot 3: Gain in dB
subplot(3,2,3)
plot(omega, gain);
title('Gain Vo/Vin dB');
xlabel('omega (rads/s)');
ylabel('Gain (dB)');
grid on;

% 3.b.iii. Plot 4 and 5: Monte-Carlo simulation of the circuit multiple times with a
% given standard deviation = 0.05 and normal distribution centered around the
% given values of C = 0.25 and Gain figures, with omega = pi
std = 0.05;

index = 0;
for i = linspace(0,100,1000)
    index = index + 1;
    C(2,1) = C_1 + std*randn();
    C(2,2) = -(C_1 + std*randn());
    C(6,6) = L + std*randn();
    C_o(index) = C_1 + std*randn();
    V_vector = (G + 1j*pi.*C_o(index))\F;
    V_vec = (G + 1j*pi.*C)\F;
    gain_d(index) = 20*log10(abs(V_vec(5))/F(1));
end

subplot(3,2,5)
histogram(C_o)
xlim([0.10,0.40])
xlabel('C');
ylabel('Number');
grid on;

subplot(3,2,6)
histogram(gain_d)
xlabel('V_o/V_i (dB)');
ylabel('Number');
grid on;
```

```
C matrix

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 matrix

G =

    1.0000         0         0         0         0         0
    1.0000   -1.5000         0         0         0   -1.0000
         0         0   -0.0671         0         0    1.0000
         0         0   -6.7060    1.0000         0         0
         0         0         0   10.0000  -10.0010         0
         0   -1.0000    1.0000         0         0         0

F vector

F =

     1
     0
     0
     0
     0
```

0



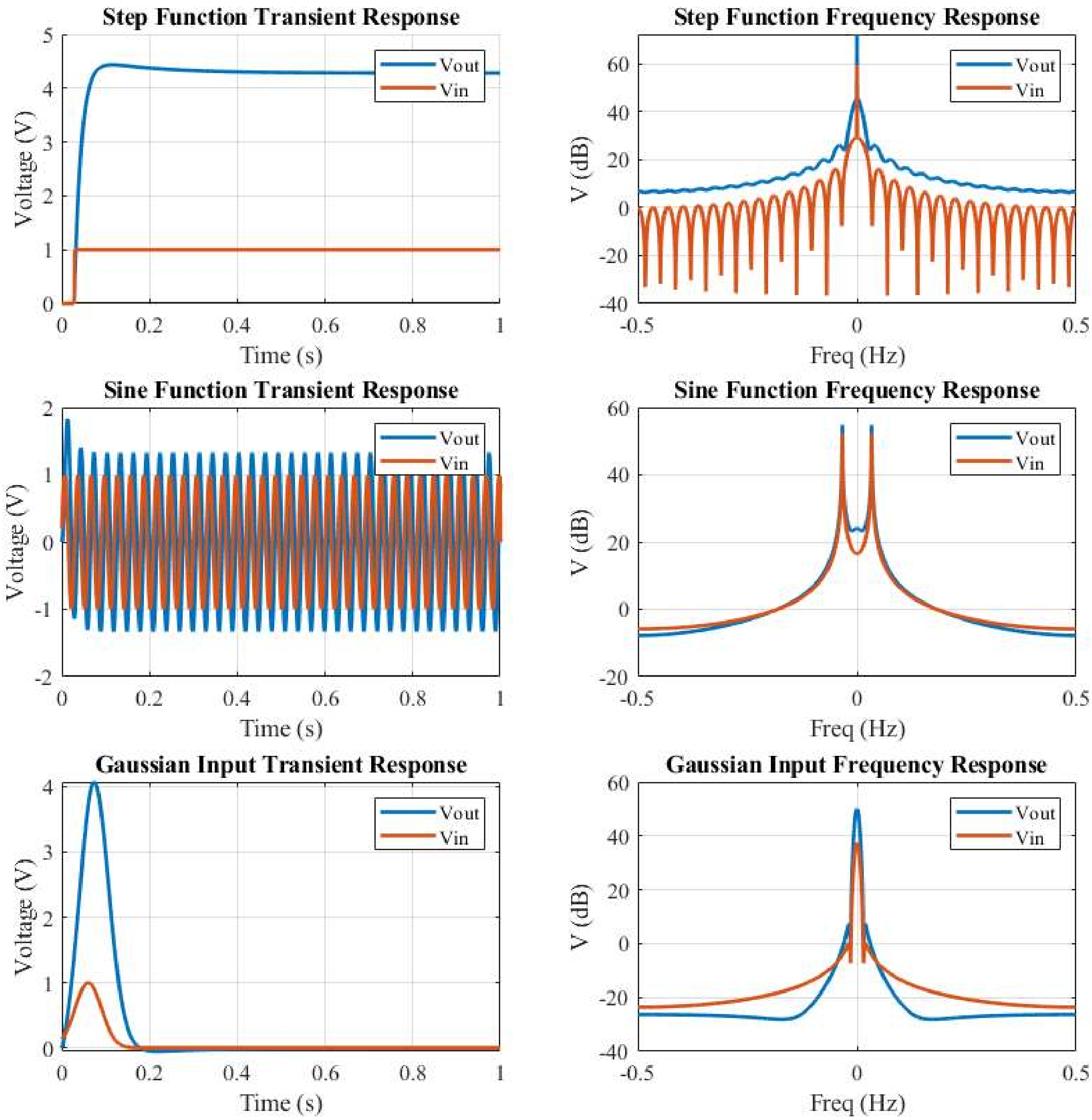


4. Transient Circuit Simulation

```
sim_time = 1;           % simulate for 1s
steps = 1000;           % simulate for 1000 steps
dt = sim_time/steps;    % dt for given time and step size

% Define the 3 types of input signals
Vin_a = zeros(1, steps);    % pulse
Vin_b = Vin_a;              % sine
Vin_c = Vin_a;              % gaussian

% Create F, Vout fields
F_a = zeros(6,1);
F_b = zeros(6,1);
F_c = zeros(6,1);

Vout_a = zeros(6, steps);
```

```
Vout_b = zeros(6, steps);
Vout_c = zeros(6, steps);

for i = 1:1:steps-1 % i*dt = time thus col number * dt = current time
    Vout_a(:,1) = 0;
    if ((i+1)*dt) > 0.029
        Vin_a(i+1) = 1;
        F_a(1) = Vin_a(i+1);
        Vout_a(:,i+1) = (C/dt + G) \ (C*Vout_a(:,i)/dt + F_a);
    else % its less than 0.03, should be off
        Vout_a(:,i+1) = (C/dt + G) \ (C*Vout_a(:,i)/dt + F_a);
    end
end

vector = linspace(0,1,steps);

figure(2)
subplot(3,2,1)
hold on;
plot(vector, Vout_a(5,:));
hold on;
grid on;
plot(vector, Vin_a);
hold on;
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Step Function Transient Response');
legend('Vout', 'Vin');

% do fft of pulse here
subplot(3,2,2)
fourier_a_out = 20*log10(abs(fftshift(fft(Vout_a(5,:)))));
fourier_a_in = 20*log10(abs(fftshift(fft(Vin_a))));
fourier_vector = -0.5: dt: 0.5-dt;
plot(fourier_vector, fourier_a_out);
hold on;
grid on;
plot(fourier_vector, fourier_a_in);
xlabel('Freq (Hz)');
ylabel('V (dB)');
legend('Vout','Vin');
title('Step Function Frequency Response');

for j = 1:1:steps-1
    Vin_b(j) = sin(2*pi*(j/0.03)*dt);
    if j > 1
        F_b(1,1) = Vin_b(1,j);
        Vout_b(:, j+1) = (C/dt + G) \ (C*Vout_b(:,j)/dt + F_b);
    else % j = 1
        F_b(1,1) = Vin_b(1,1);
        Vout_b(:, j) = (C/dt + G) \ (C*Vout_b(:,1)/dt + F_b);
    end
end


% plot transient of sine here
subplot(3,2,3)
plot(vector, Vout_b(5,:));
hold on;
grid on;
plot(vector, Vin_b)
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Sine Function Transient Response');
legend('Vout', 'Vin');

% do fft of sine here
subplot(3,2,4)
fourier_b_out = 20*log10(abs(fftshift(fft(Vout_b(5,:)))));
fourier_b_in = 20*log10(abs(fftshift(fft(Vin_b))));
plot(fourier_vector, fourier_b_out);
hold on;
grid on;
plot(fourier_vector, fourier_b_in);
xlabel('Freq (Hz)');
ylabel('V (dB)');
legend('Vout','Vin');
title('Sine Function Frequency Response');
```

```
Vin_c = gaussmf(vector, [0.03 0.06]);

for k = 1:1:steps-1
    if k > 1
        F_c(1,1) = Vin_c(1,k);
        Vout_c(:, k+1) = (C/dt + G) \ (C*Vout_c(:,k)/dt + F_c);
    else % k = 1
        F_c(1,1) = Vin_c(1,1);
        Vout_c(:, k) = (C/dt + G) \ (C*Vout_c(:,1)/dt + F_c);
    end
end

% plot gaussian here
subplot(3,2,5)
plot(vector, Vout_c(5,:));
hold on;
grid on;
plot(vector, Vin_c)
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Gaussian Input Transient Response');
legend('Vout', 'Vin');

% do fft of gaussian here
subplot(3,2,6)
fourier_c_out = 20*log10(abs(fftshift(fft(Vout_c(5,:)))));
fourier_c_in = 20*log10(abs(fftshift(fft(Vin_c))));
plot(fourier_vector, fourier_c_out);
hold on;
grid on;
plot(fourier_vector, fourier_c_in);
xlabel('Freq (Hz)');
ylabel('V (dB)');
legend('Vout','Vin');
title('Gaussian Input Frequency Response');
```



5. Circuit with Noise

A capacitor Cn and current source I_n have been added such that I_n helps model the thermal noise generated in R3. This I_n is added in parallel to R3 in the circuit. Cn acts to limit the noise and impacts the C matrix by adding an extra KCL equation to the circuit representation. C_noise, F_noise and G_noise are defined here again since I did this question seperate from part 4 but compiled the 2 scripts for final submission

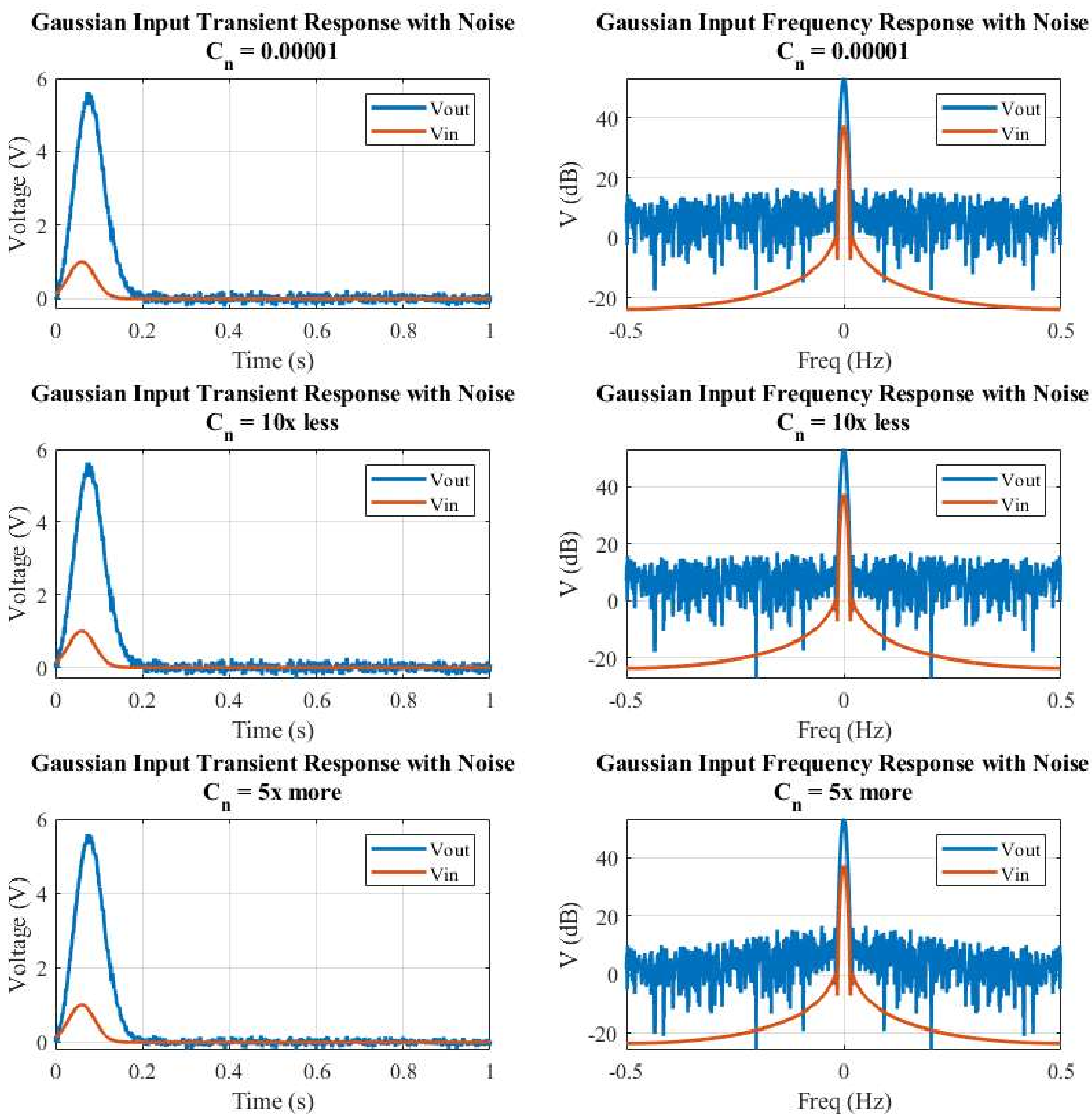
```
Cn = 0.00001;
I_n_trans = 0.001*randn(steps,1);
I_n = I_n_trans';

C_noise = zeros(8,8);
F_noise = zeros(8,1);
G_noise = zeros(8,8);

C_noise(1,1) = C_1;
C_noise(1,2) = -C_1;
C_noise(2,1) = -C_1;
C_noise(2,2) = C_1;
C_noise(6,6) = -L;
C_noise(3,3) = Cn;
```

```
G_noise(8,1) = 1;
G_noise(8,2) = -1;
G_noise(8,8) = 1;
G_noise(2,1) = -1;
G_noise(2,2) = 1.5;
G_noise(2,6) = 1;
G_noise(3,3) = 0.1;
G_noise(3,6) = -1;
G_noise(4,4) = 10;
G_noise(4,5) = -10;
G_noise(4,7) = 1;
G_noise(5,4) = -10;
G_noise(5,5) = 10;
G_noise(6,2) = 1;
G_noise(6,3) = -1;
G_noise(7,3) = -10;
G_noise(7,4) = 1;
G_noise(1,1) = 1;

% 5.a. Updated C matrix
display('C matrix with noise'); C_noise
%display('G matrix with noise'); G_noise

Vout_noise = zeros(8,steps);

for m = 1:1:steps-1
    if m > 1
        F_noise(1,1) = Vin_c(1,m);
        F_noise(3,1) = -I_n(1,m);
        Vout_noise(:, m+1) = (C_noise/dt + G_noise) \ (C_noise*Vout_noise(:,m)/dt + F_noise);
    else % m = 1
        F_noise(1,1) = Vin_c(1,1);
        Vout_noise(:, m) = (C_noise/dt + G_noise) \ (C_noise*Vout_noise(:,1)/dt + F_noise);
    end
end

% plot of gaussian
figure(3)
subplot(3,2,1)
plot(vector, Vout_noise(5,:));
hold on;
grid on;
plot(vector, Vin_c)
xlabel('Time (s)');
ylabel('Voltage (V)');
title({
    ['Gaussian Input Transient Response with Noise']
    ['C_n = 0.00001']
    });
legend('Vout', 'Vin');

% fft of gaussian
subplot(3,2,2)
fourier_noise_out = 20*log10(abs(fftshift(fft(Vout_noise(5,:)))));
fourier_noise_in = 20*log10(abs(fftshift(fft(Vin_c))));
plot(fourier_vector, fourier_noise_out);
hold on;
grid on;
plot(fourier_vector, fourier_noise_in);
xlabel('Freq (Hz)');
ylabel('V (dB)');
legend('Vout','Vin');
title({
    ['Gaussian Input Frequency Response with Noise']
    ['C_n = 0.00001']
    });

% 5.c.vi. Changing Cn to 10x less the given value
Cn = 0.000001;
C_noise(3,3) = Cn;

for m = 1:1:steps-1
    if m > 1
        F_noise(1,1) = Vin_c(1,m);
        F_noise(3,1) = -I_n(1,m);
        Vout_noise(:, m+1) = (C_noise/dt + G_noise) \ (C_noise*Vout_noise(:,m)/dt + F_noise);
    else % m = 1
        F_noise(1,1) = Vin_c(1,1);
        Vout_noise(:, m) = (C_noise/dt + G_noise) \ (C_noise*Vout_noise(:,1)/dt + F_noise);
```

```
    end
end

% plot gaussian here
figure(3)
subplot(3,2,3)
plot(vector, Vout_noise(5,:));
hold on;
grid on;
plot(vector, Vin_c)
xlabel('Time (s)');
ylabel('Voltage (V)');
title({
    ['Gaussian Input Transient Response with Noise']
    ['C_n = 10x less']
    });
%subtitle('C_n = 10x less');
legend('Vout', 'Vin');

% fft of gaussian here
subplot(3,2,4)
fourier_noise_out = 20*log10(abs(fftshift(fft(Vout_noise(5,:)))));
fourier_noise_in = 20*log10(abs(fftshift(fft(Vin_c))));
plot(fourier_vector, fourier_noise_out);
hold on;
grid on;
plot(fourier_vector, fourier_noise_in);
xlabel('Freq (Hz)');
ylabel('V (dB)');
legend('Vout','Vin');
title({
    ['Gaussian Input Frequency Response with Noise']
    ['C_n = 10x less']
    });

% 5.c.vi. Changing Cn to 5x greater than the given value
Cn = 0.00005;
C_noise(3,3) = Cn;

for m = 1:1:steps-1
    if m > 1
        F_noise(1,1) = Vin_c(1,m);
        F_noise(3,1) = -I_n(1,m);
        Vout_noise(:, m+1) = (C_noise/dt + G_noise) \ (C_noise*Vout_noise(:,m)/dt + F_noise);
    else % m = 1
        F_noise(1,1) = Vin_c(1,1);
        Vout_noise(:, m) = (C_noise/dt + G_noise) \ (C_noise*Vout_noise(:,1)/dt + F_noise);
    end
end

% plot gaussian here
subplot(3,2,5)
plot(vector, Vout_noise(5,:));
hold on;
grid on;
plot(vector, Vin_c)
xlabel('Time (s)');
ylabel('Voltage (V)');
title({
    ['Gaussian Input Transient Response with Noise']
    ['C_n = 5x more']
    });
legend('Vout', 'Vin');

% fft of gaussian here
subplot(3,2,6)
fourier_noise_out = 20*log10(abs(fftshift(fft(Vout_noise(5,:)))));
fourier_noise_in = 20*log10(abs(fftshift(fft(Vin_c))));
plot(fourier_vector, fourier_noise_out);
hold on;
grid on;
plot(fourier_vector, fourier_noise_in);
xlabel('Freq (Hz)');
ylabel('V (dB)');
legend('Vout','Vin');
title({
    ['Gaussian Input Frequency Response with Noise']
    ['C_n = 5x more']
    });
```

```
% Observations: for a larger Cn value larger noise values are removed from
% the system and thus the bandwidth is a bit smaller compared to normal
% (note figure 3, subplot #5 has the least amount of noise (clear in the
% time domain) compared to the other 2).
% For smaller Cn the opposite effect occurs and the bandwidth is larger.
```

```
C matrix with noise

C_noise =

  Columns 1 through 7

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

  Column 8

         0
         0
         0
         0
         0
         0
         0
         0
```



5.c.vii. Changing the Time-Step

Two new time steps were used here: 100 and 5000. These are plotted in addition to 1000 in the time and frequency domain for a gaussian input. I tried to make a function that would do the looping based on step size and that is being called here: step_solver. Note that a larger step size = smaller dt = better plotting resolution.

```
% Reset Cn and C matrix to Cn = 0.00001
Cn = 0.00001;
C_noise(3,3) = Cn;

% call the step solving function with 5000, 1000, 100 step inputs
[vector_1, Vout_1, I_new_1] = step_solver(C_noise, G_noise, 5000);
[vector_2, Vout_2, I_new_2] = step_solver(C_noise, G_noise, 1000);
[vector_3, Vout_3, I_new_3] = step_solver(C_noise, G_noise, 100);

% 5.c.vii. plot of Vout with varying time steps
figure(4)
plot(vector_1, Vout_1(5,:))
hold on;
plot(vector_2, Vout_2(5,:))
hold on;
plot(vector_3, Vout_3(5,:))
```

```
grid on;
legend('5000 Steps', '1000 Steps', '100 Steps')
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Gaussian Input Transient Response with Noise for varying Time Steps');

% 5.c.vii. frequency plot of Vout for varying time steps
figure(5)
fourier_noise_5k = 20*log10(abs(fftshift(fft(Vout_1(5,:)))));
fourier_noise_1k = 20*log10(abs(fftshift(fft(Vout_2(5,:)))));
fourier_noise_100 = 20*log10(abs(fftshift(fft(Vout_3(5,:)))));
plot(vector_1, fourier_noise_5k);
hold on;
grid on;
plot(vector_2, fourier_noise_1k);
hold on;
plot(vector_3, fourier_noise_100);
xlabel('Freq (Hz)');
ylabel('V (dB)');
legend('5000 Steps', '1000 Steps', '100 Steps');
title('Gaussian Input Frequency Response with Noise for varying Time Steps');
```

Extra Checks for Section 5

Not a part of the assignment, used for debugging purposes only

```
% Uncomment to output check for: are the noise values for different steps
% bandlimited by capacitor
% figure(6)
% plot(vector_1, I_new_1)
% hold on;
% plot(vector_2, I_new_2)
% hold on;
% plot(vector_3, I_new_3)
% grid on;
% legend('5000 Steps', '1000 Steps', '100 Steps');
% title({
%     ['I_n values plotted for each step size confirming that']
%     ['noise is bandlimited by the capacitor in parallel with the noisy resistor']
%     });

% Uncomment to output check for: is randn() is returning pseudorandom normally distributed values
% figure(7)
% subplot(1,3,1)
% histogram(I_new_1)
% title('5000 Steps');
% subplot(1,3,2)
% histogram(I_new_2)
% title('1000 Steps');
% subplot(1,3,3)
% histogram(I_new_3)
% title('100 Steps');
```

6. Non-linearity

If the voltage source on the output stage described by the transconductance equation: $V = \alpha I_3$ is changed to the new equation: $V = \alpha I_3 + \beta(I_3)^2 + \gamma(I_3)^3$ then in addition to the G (conductance matrix), C(capacitance matrix), F(time-varying vector with independent sources), we will need to create a "B" non-linear vector to hold the new equation. Since the circuit is now non-linear, frequency analysis will have to be performed to get the soln for an operating point (ex. t=0) and a Jacobean can then find the non-linear solution. An iterative solution at each time step would need to be implemented using Newton-Raphson method. For it to properly converge, we can add an additional G_min conductance. The new MNA matrix equation to be solved is of the form: $CdV/dt + GV + B = F$ and to get the new Voltage, the B vector must be included in the main loops used in this assignment that solve for V. For example, in line 196, new Vout_a(:,i+1) = (C/dt + G) \ (C*Vout_a(:,i)/dt + F_a - B);

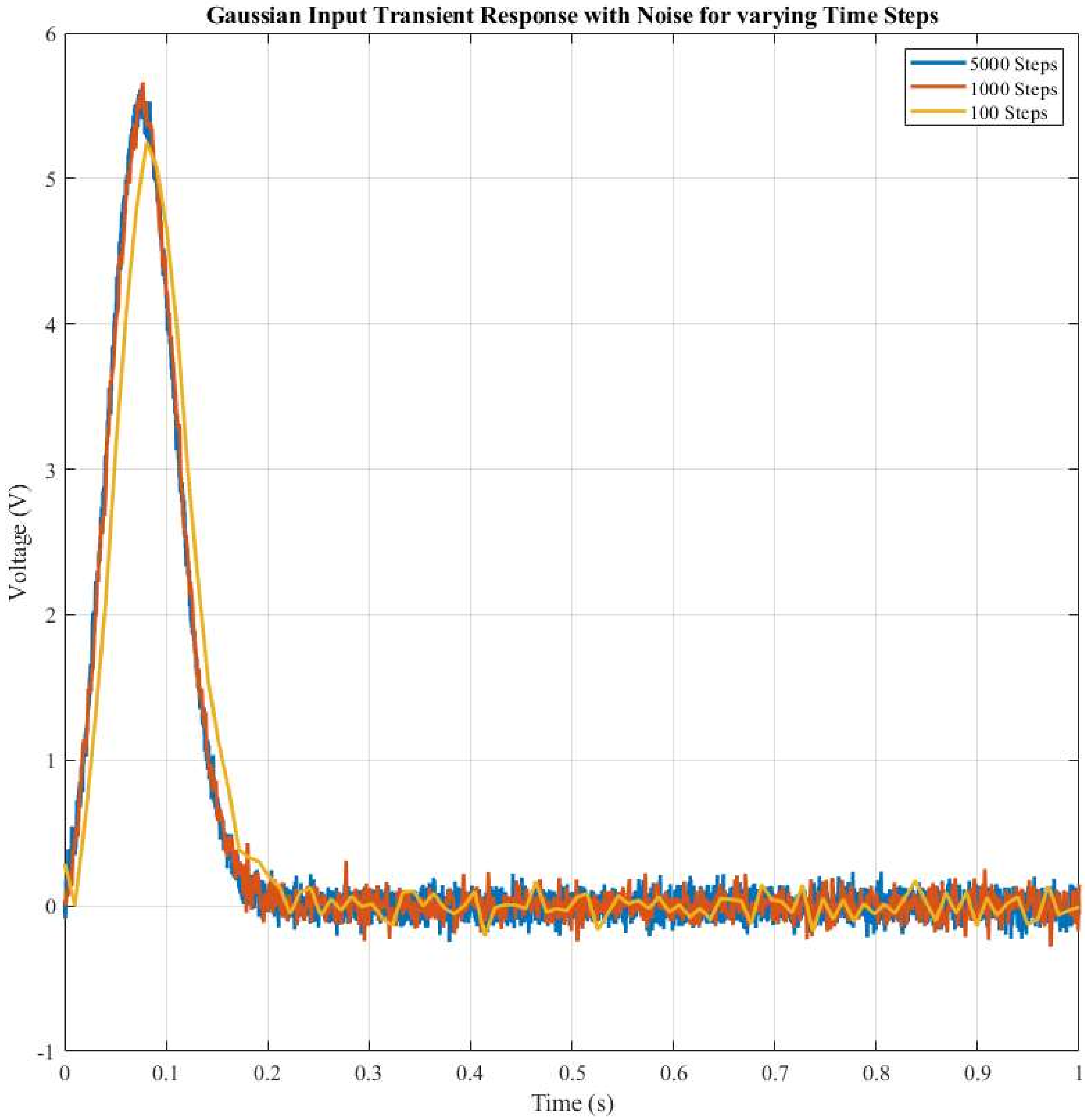
From Assignment 2

```
function [Curr, Vmap, Ex, Ey, eFlowx, eFlowy, Cnx] = current_solver(stepped_voltage)
%current_solver
% Inputs: stepped_voltage = the present voltage value that goes from 0.1 to
% 10V (assignment 4 specs)
% Outputs: Cnx = average Current for that Voltage value
%          Curr, Vmap, Ex, Ey, eFlowx, eFlowy

Len = 200e-9;
Wid = 100e-9;
ny = 200;
nx = 100;
```

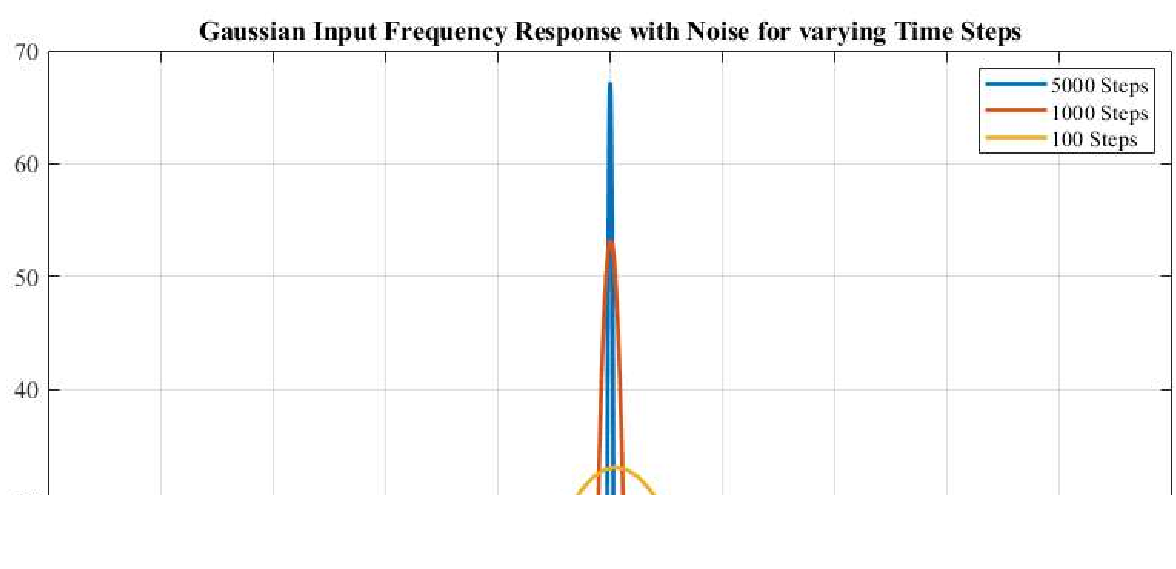
```
G = sparse(ny*nx, ny*nx);
V = zeros(nx*ny,1);
boundary_condition = [1 1 0 0];
part = 3;

cMap = zeros(nx, ny);
boxL = round(0.4*nx);
boxR = round(0.6*nx);
boxT = round(0.6*ny);
boxB = round(0.4*ny);

for i = 1:nx
    for j = 1:ny
        cMap(i,j) = 1; % outside the box
            if ((i>=boxL && i<=boxR && j>=boxT) || (i>=boxL && i<=boxR && j<=boxB ))
                cMap(i,j) = 0.01; % inside the box
            end
        end
end

% figure(1)
% plot(cMap)
%
% figure(2)
% surf(cMap)

% Using Info from Finite Diff LaPlace Lectures Slides 37,39
% G Matrix
for i = 1:nx
    for j = 1:ny
        n = j + (i - 1) * ny;

        if i == 1
            G(n, :) = 0;
            G(n, n) = 1;
            V(n) = stepped_voltage;
        elseif i == nx
            G(n, :) = 0;
            G(n, n) = 1;
        elseif j == 1
            nxm = j + (i - 2) * ny;
            nxp = j + (i) * ny;
            nyp = j + 1 + (i - 1) * ny;

            rxm = (cMap(i, j) + cMap(i - 1, j)) / 2.0;
            rxp = (cMap(i, j) + cMap(i + 1, j)) / 2.0;
            ryp = (cMap(i, j) + cMap(i, j + 1)) / 2.0;

            G(n, n) = -(rxm+rxp+ryp);
            G(n, nxm) = rxm;
            G(n, nxp) = rxp;
            G(n, nyp) = ryp;

        elseif j ==  ny
            nxm = j + (i - 2) * ny;
            nxp = j + (i) * ny;
            nym = j - 1 + (i - 1) * ny;

            rxm = (cMap(i, j) + cMap(i - 1, j)) / 2.0;
            rxp = (cMap(i, j) + cMap(i + 1, j)) / 2.0;
            rym = (cMap(i, j) + cMap(i, j - 1)) / 2.0;

            G(n, n) = -(rxm + rxp + rym);
            G(n, nxm) = rxm;
            G(n, nxp) = rxp;
            G(n, nym) = rym;
        else
            nxm = j + (i-2)*ny;
            nxp = j + (i)*ny;
            nym = j-1 + (i-1)*ny;
            nyp = j+1 + (i-1)*ny;

            rxm = (cMap(i,j) + cMap(i-1,j))/2.0;
            rxp = (cMap(i,j) + cMap(i+1,j))/2.0;
            rym = (cMap(i,j) + cMap(i,j-1))/2.0;
            ryp = (cMap(i,j) + cMap(i,j+1))/2.0;

            G(n,n) = -(rxm+rxp+rym+ryp);
```

```
            G(n,nxm) = rxm;
            G(n,nxp) = rxp;
            G(n,nym) = rym;
            G(n,nyp) = ryp;
        end

    end
end

Voltage = G\V;
Vmap = zeros(nx,ny);

% zz = reshape(Voltage,[10,20]); % doesnt work: try loop to reshape vmap
for x = 1:nx
    for y = 1:ny
        n = y + (x-1)*ny;
        Vmap(x,y) = Voltage(n);
    end
end

% figure(3)
% surf(Vmap)

% Perform gradient for Ex and Ey
[Ey, Ex] = gradient(Vmap);

Ex = -Ex;
Ey = -Ey;

% figure(4)
% quiver(Ex', Ey');
% axis([0 nx 0 ny]);

% normalize to get current from current density
eFlowx = cMap.*Ex;
eFlowy = cMap.*Ey;


Curr = [eFlowx(:),eFlowy(:)];

% uncomment here to check that total current, avg of end currents is the same
% C0 = sum(eFlowx(1,:));
 Cnx = sum(eFlowx(nx,:));
% Currentt = (C0 + Cnx) * 0.5;

end
```

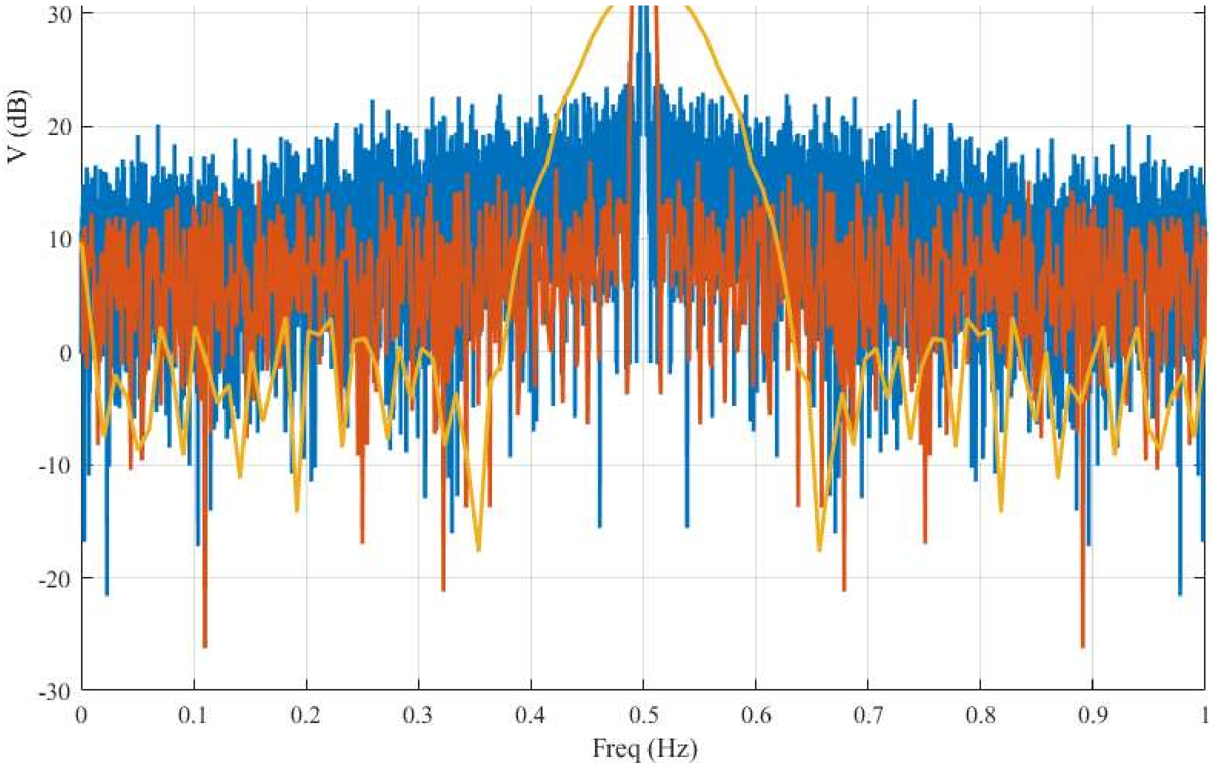
```
function [x_axis, output_V, I_new] = step_solver(C_noise, G_noise, step_count)
%Step_Solver takes the C, G, #iterations as input
%   Outputs a vector to be plotted for x-axis, the output voltage, and
%   current noise levels


dt = 1/step_count;                      % change dt based on current step size
x_axis = linspace(0,1,step_count);      % x values to be plotted from 0s to 1s
input_V = gaussmf(x_axis, [0.03 0.06]); % input function = gaussian for these x-values
output_V = zeros(8,step_count);         % create output vector
F_value = zeros(8,1);                   % F vector

I_new = 0.001*randn(1, step_count);     % define thermal noise

for k = 1:1:step_count-1
    if k > 1
        F_value(1,1) = input_V(1,k);
        F_value(3,1) = I_new(1,k);
        output_V(:, k+1) = (C_noise/dt + G_noise) \ (C_noise*output_V(:,k)/dt + F_value);
    else % k = 1
        F_value(1,1) = input_V(1,1);
        output_V(:, k) = (C_noise/dt + G_noise) \ (C_noise*output_V(:,1)/dt + F_value);
    end
end
end
```



Gaussian Input Transient Response with Noise for varying Time Steps
5000 Steps
1000 Steps
100 Steps
Voltage (V)
Time (s)
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 Response with Noise for varying Time Steps
5000 Steps
1000 Steps
100 Steps
70
60
50
40


V (dB)
Freq (Hz)
30
20
10
0
-10
-20
-30
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1