ELEC 4700 Assignment 4

Table of Contents

Circuit Modeling

Keegan Mauger 101042551

Section 1: Revisiting PA7 and Assignment 3

In section 1, the MNA modeling done in PA7 and the monte-carlo/finite difference method code devoloped in assignment 3 are combined. First, assignment 3 is repurposed to find the resistance of R3 from PA7.

Section 1.1: Finding the Resistance of R3

The width of the bottleneck from assignment 3 was modified to find a resistance suggested by the TAs. First, teh assignment code was re-run to find a plot of the average current vs Vin.

```
clear all
close all
clc
%set(0,'DefaultFigureWindowStyle','docked')
set(0,'defaultaxesfontsize',10)
set(0,'defaultaxesfontname','Times New Roman')
set(0,'DefaultLineLineWidth', 0.5);

%-----------------------------------------------------------------------
% Finite Difference Method Simulation
%-----------------------------------------------------------------------
```

```
Zfac = 15e-9;
Vin = linspace(0.1,10,30);
Complete = linspace(0,100,30);
for J=1:length(Vin)
%fprintf('\n%3.2f percent complete',Complete(J));
% Fixed bottleneck of 0.2x10-7m
% Solving V=V0 @ x=0 and V=0 @ x=L in region LxW
% Implement funtion 'pbaspect' to fix Z aspect ratio
clearvars -except Vin J Ix_total Iavg Zfac Complete

Conc = 1e19;
L = 200e-9;
W = 100e-9;
V0 = Vin(J);

fMesh = 1;
nx = fMesh*200;
ny = fMesh*100;
La = linspace(0,L,nx);
Wa = linspace(0,W,ny);
G = sparse(nx*ny);
%V = sparse(nx,ny);
F = sparse(1,nx*ny);


Acond = 1;              % background conductivity of region, low resistance
Bcond = 1e-2;           % Conductivity of boxes, highly resistive
cMap = zeros(nx,ny);
Lb = 40e-9;
Wb = 40e-9;
for u = 1:nx
    for v = 1:ny
        if (u >= 80 && u <= 120)
            if v >= 0 && v <= 40-1+J
                cMap(u,v) = Bcond;
            elseif v >= 60+1-J && v <= 100
                cMap(u,v) = Bcond;
            else
                cMap(u,v) = Acond;
            end
        else
            cMap(u,v) = Acond;
        end
    end
end




for i = 1:nx                    %Iteration through length
    for j = 1:ny                %Iteration through width
        n = j + (i-1)*ny;
```

```
        if i == 1           % x=0 BCs
            G(n,:) = 0;
            G(n,n) = 1;
            F(n) = V0;
        elseif i == nx      % x=1 BCs
            G(n,:) = 0;
            G(n,n) = 1;
            F(n) = 0;        %F(n)=0 sets z at final width to 0

% COMMENT BELOW FOR 1a
            %F(n) = 1;        %F(n)=1 sets z at final width to 1

        elseif j == 1                  % y=0 BCs
            nxm = j + (i-2)*ny;
            nxp = j + (i)*ny;
            nyp = j+1 + (i-1)*ny;

            rxm = (cMap(i,j) + cMap(i-1,j))/2;
            rxp = (cMap(i,j) + cMap(i+1,j))/2;
            ryp = (cMap(i,j) + cMap(i,j+1))/2;

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

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

            rxm = (cMap(i,j) + cMap(i-1,j))/2;
            rxp = (cMap(i,j) + cMap(i+1,j))/2;
            rym = (cMap(i,j) + cMap(i,j-1))/2;

            G(n,n) = -(rxm+rxp+rym);
            G(n,nxm) = rxm;
            G(n,nxp) = rxp;
            G(n,nym) = rym;


% COMMENT ABOVE FOR 1a

        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;
            rxp = (cMap(i,j) + cMap(i+1,j))/2;
            rym = (cMap(i,j) + cMap(i,j-1))/2;
            ryp = (cMap(i,j) + cMap(i,j+1))/2;
```

```
            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
% figure(1)
% spy(G)

V = G\F';

Vmap = zeros(nx,ny);
for i = 1:nx
    for j = 1:ny
        n = j + (i-1)*ny;
        Vmap(i,j) = V(n);
    end
end


for i = 1:nx
    for j = 1:ny
        if i == 1
            Ex(i, j) = (Vmap(i + 1, j) - Vmap(i, j));
        elseif i == nx
            Ex(i, j) = (Vmap(i, j) - Vmap(i - 1, j));
        else
            Ex(i, j) = (Vmap(i + 1, j) - Vmap(i - 1, j)) * 0.5;
        end
        if j == 1
            Ey(i, j) = (Vmap(i, j + 1) - Vmap(i, j));
        elseif j == ny
            Ey(i, j) = (Vmap(i, j) - Vmap(i, j - 1));
        else
            Ey(i, j) = (Vmap(i, j + 1) - Vmap(i, j - 1)) * 0.5;
        end
    end
end

Ex = -Ex;
Ey = -Ey;

eFlowx = cMap .* Ex;        %Jx
eFlowy = cMap .* Ey;        %Jy
C0 = sum(eFlowx(1, :));
Cnx = sum(eFlowx(nx, :));
Curr = (C0 + Cnx) * 0.5;

Ex = Ex';
Ey = Ey';
```

```
% figure(4)
% subplot(1, 2, 2), quiver(Ex, Ey);
% axis([0 nx 0 ny]);
% title('Electric Field Map')
% xlabel('Region Length')
% ylabel('Region Width')
% pbaspect([1 1 0.5])
% subplot(1, 2, 1), H = surf(La,Wa,Vmap');
% set(H, 'linestyle', 'none');
% %view(90, 270)
% title('Voltage Map')
% xlabel('Region Length')
% ylabel('Region Width')
% pbaspect([1 1 0.5])
% saveas(gcf,'Figure4')

% clearvars -except Ex Ey

%-------------------------------------------------------------------------
% Beginning Monte Carlo Simulation
%-------------------------------------------------------------------------


set(0,'DefaultFigureWindowStyle','docked')
set(0,'defaultaxesfontsize',10)
set(0,'defaultaxesfontname','Times New Roman')
set(0,'DefaultLineLineWidth', 0.5);



global C

C.q_0 = 1.60217653e-19;             % electron charge
C.hb = 1.054571596e-34;             % Dirac constant
C.h = C.hb * 2 * pi;                % Planck constant
C.m_0 = 9.10938215e-31;             % electron mass
C.kb = 1.3806504e-23;               % Boltzmann constant
C.eps_0 = 8.854187817e-12;          % vacuum permittivity
C.mu_0 = 1.2566370614e-6;           % vacuum permeability
C.c = 299792458;                    % speed of light
C.g = 9.80665;                      % metres (32.1740 ft) per s²
C.m_n = 0.26 * C.m_0;               % effective electron mass
C.am = 1.66053892e-27;              % atomic mass unit
C.T = 300;
C.vth = sqrt(2*C.kb * C.T / C.m_n);

temp = C.T;

SPECDIFF_BOUND = 0;

% figure(3)
% subplot(2,1,1);
% rectangle('Position',[0 0 200e-9 100e-9])
% hold on
```

```
% rectangle('Position',[0.8e-7 0 0.4e-7 0.4e-7])
% hold on
% rectangle('Position',[0.8e-7 0.6e-7 0.4e-7 0.4e-7])
% hold on

%-----------------------------------------------------------------------
% Initializing Positions
%-----------------------------------------------------------------------

    N = 30000;        % Number of electrons
    i = 0;
    j = 0;

    for i=1:N
        px(i) = 0 + (200e-9 - 0).*rand(1,1);
        py(i) = 0 + (100e-9 - 0).*rand(1,1);
        while (0.8e-7 <= px(i) && px(i) <= 1.2e-7) && (0 <= py(i) && py(i) <=
 (0.4e-7 - Zfac) ) ||...
                (0.8e-7 <= px(i) && px(i) <= 1.2e-7) && ((0.6e-7 + Zfac) <=
 py(i) && py(i) <= 1e-7)
            px(i) = 0 + (200e-9 - 0).*rand(1,1);
            py(i) = 0 + (100e-9 - 0).*rand(1,1);
        end
    end

%-----------------------------------------------------------------------
% Voltage Applied Across x-Dimension to Find Electric Field
%-----------------------------------------------------------------------

    V0x = Vin(J);
    V0y = 0;
    L = 200e-9;
    W = 100e-9;
    % E0x = V0x / L;
    % E0y = V0y / W;
    fMesh = 1;
    nx = fMesh*200;
    ny = fMesh*100;
    % G = sparse(nx,ny);
    % F = sparse(1,nx*ny);

    La = linspace(0,L,nx);
    Wa = linspace(0,W,ny);

    deltax = L/nx;
    deltay = W/ny;

    Ex = Ex./deltax;
    Ey = Ey./deltay;

    % Emapx = zeros(ny,nx);
    Emapx = Ex;
    % Emapy = zeros(ny,nx);
    Emapy = Ey;
```

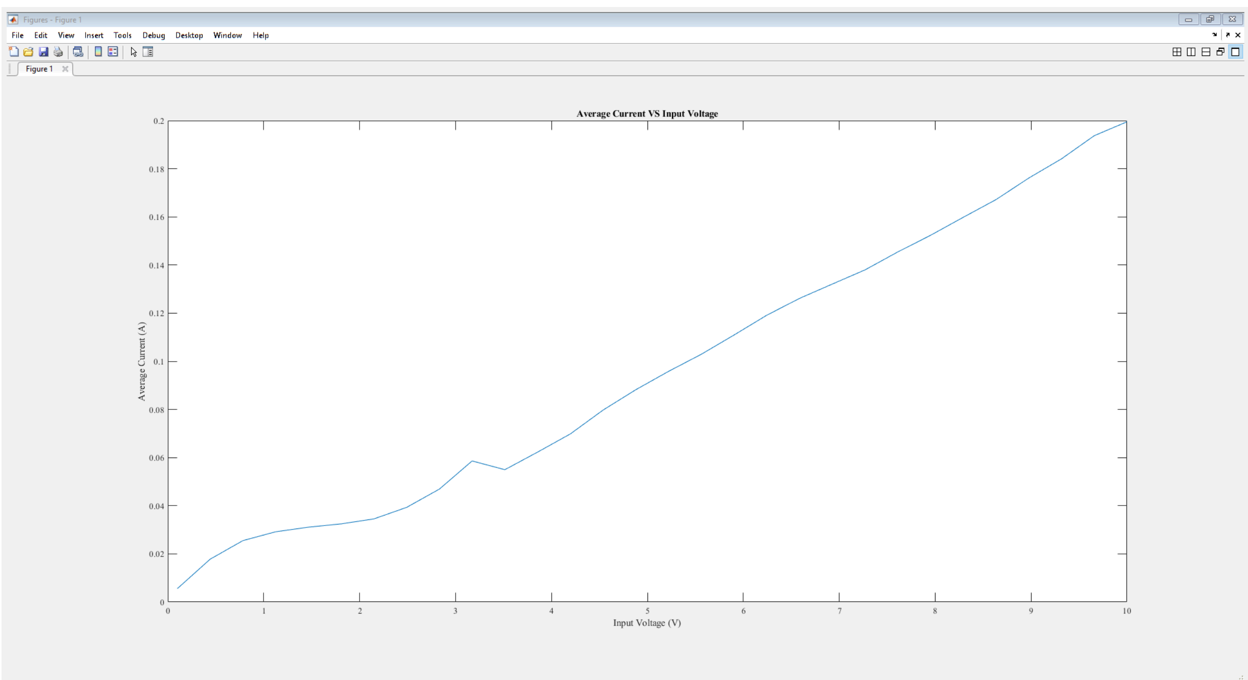
```
    % for i = 1:width(La)
    %     for j = 1:width(Wa)
    %         Emapx(j,i) = E0x;
    %         Emapy(j,i) = E0y;
    %     end
    % end
    % %surf(La,Wa,Emapx)
    %
    % Fex = abs(C.q_0*E0x);
    % aex = Fex/C.m_n;
    % Fey = abs(C.q_0*E0y);
    % aey = Fey/C.m_n;

    Fex = zeros(ny,nx);
    aex = zeros(ny,nx);
    Fey = zeros(ny,nx);
    aey = zeros(ny,nx);

    for i = 1:width(Wa)
        for j = 1:width(La)
            Fex(i,j) = abs(C.q_0*Emapx(i,j));
            aex(i,j) = Fex(i,j)/C.m_n;
            Fey(i,j) = abs(C.q_0*Emapy(i,j));
            aey(i,j) = Fey(i,j)/C.m_n;
        end
    end




%--------------------------------------------------------------------------
% Thermal Velocity and Direction
%--------------------------------------------------------------------------

    vth = C.vth;

    for j=1:N
        vx(j) = (vth/sqrt(2))*randn();
        vy(j) = (vth/sqrt(2))*randn();
        vth_calc(j) = sqrt(vx(j)^2 + vy(j)^2);
    end


    t = 0;
    T(1) = 0;
    dt = 1e-14;     % time step

    for l=1:N           %Scattering time step
        ndt(l) = dt;
    end
    P_scat = 0;
    Tmn = 0.2e-12;
```

```
px_prev = 0;
py_prev = 0;
T_prev = 0;
vx_total = 0;
vy_total = 0;

sampleidx = randi(N,10,1);
Ix = 0;
Jx = 0;
aex = aex';
aey = aey';



for t=2:1000
    vx_total = 0;
    vy_total = 0;
    for k=1:N

        rpx = 0;
        rpy = 0;
        if px(k) == 200e-9
            px(k) = 0;
            px_prev(k) = px(k);
        elseif px(k) == 0
            px(k) = 200e-9;
            px_prev(k) = px(k);
        else
            px(k) = px(k);
        end

        P_scat(k) = 1 - exp(-(dt/Tmn));
        if P_scat(k) > rand()
            vx(k) = (vth/sqrt(2))*randn();
            vy(k) = (vth/sqrt(2))*randn();
        else
            ndt(k) = ndt(k) + dt;
        end

        px_prev(k) = px(k);
        py_prev(k) = py(k);

        rpx = round(px(k)*1e9);
        if rpx == 0
            rpx = 1;
        end
        rpy = round(py(k)*1e9);
        if rpy == 0
            rpy = 1;
        end
```

```
            px(k) = px(k) + vx(k)*dt + aex(rpx,rpy)*dt^2;        % Adding
acceleration
            vx(k) = vx(k) + aex(rpx,rpy)*dt;
            py(k) = py(k) + vy(k)*dt + aey(rpx,rpy)*dt^2;
            vy(k) = vy(k) + aey(rpx,rpy)*dt;


            % Reflection on top and bottom borders
            if py(k) >= 100e-9 || py(k) <= 0
                vy(k) = -vy(k);
                if py(k) >= 100e-9
                    py(k) = 100e-9;
                end
                if py(k) <= 0
                    py(k) = 0;
                end
            end
            % Reflection on bottom of upper box
            if (py(k) >= (0.6e-7 + Zfac)) && (0.8e-7 <= px(k) && px(k) <=
1.2e-7)...
                    && ( 0.8e-7 <= px_prev(k) && px_prev(k) <= 1.2e-7)
                if SPECDIFF_BOUND == 1
                    vx(k) = (vth/sqrt(2))*randn();
                    vy(k) = (vth/sqrt(2))*randn();
                else
                    vy(k) = -vy(k);
                end
                py(k) = 0.601e-7 + Zfac;
                %end
                % Reflection on top of lower box
            elseif (py(k) <= 0.4e-7 - Zfac) && (0.8e-7 <= px(k) && px(k) <=
1.2e-7)...
                    && (0.8e-7 <= px_prev(k) && px_prev(k) <= 1.2e-7)
                if SPECDIFF_BOUND == 1
                    vx(k) = (vth/sqrt(2))*randn();
                    vy(k) = (vth/sqrt(2))*randn();
                else
                    vy(k) = -vy(k);
                end
                py(k) = 0.399e-7 - Zfac;
                %end
                % Reflection on left of lower box
            elseif (0 <= py(k) && py(k) <= 0.4e-7 - Zfac) && (0.8e-7 <= px(k)
&& px(k) <= 1e-7)
                if SPECDIFF_BOUND == 1
                    vx(k) = (vth/sqrt(2))*randn();
                    vy(k) = (vth/sqrt(2))*randn();
                else
                    vx(k) = -vx(k);
                end
                px(k) = 0.799e-7;
                %end
                % Reflection on right of lower box
```

```
            elseif (0 <= py(k) && py(k) <= 0.4e-7 - Zfac) && (1e-7 <= px(k) &&
px(k) <= 1.2e-7)
                if SPECDIFF_BOUND == 1
                    vx(k) = (vth/sqrt(2))*randn();
                    vy(k) = (vth/sqrt(2))*randn();
                else
                    vx(k) = -vx(k);
                end
                px(k) = 1.201e-7;
                %end
                % Reflection on left of upper box
            elseif (0.6e-7 + Zfac <= py(k) && py(k) <= 1e-7) && (0.8e-7 <=
px(k) && px(k) <= 1e-7)
                if SPECDIFF_BOUND == 1
                    vx(k) = (vth/sqrt(2))*randn();
                    vy(k) = (vth/sqrt(2))*randn();
                else
                    vx(k) = -vx(k);
                end
                px(k) = 0.799e-7;
                %end
                % Reflection on right of upper box
            elseif (0.6e-7 + Zfac <= py(k) && py(k) <= 1e-7) && (1e-7 <= px(k)
&& px(k) <= 1.2e-7)
                if SPECDIFF_BOUND == 1
                    vx(k) = (vth/sqrt(2))*randn();
                    vy(k) = (vth/sqrt(2))*randn();
                else
                    vx(k) = -vx(k);
                end
                px(k) = 1.201e-7;
            end

            % x-axis transition
            if px(k) > 200e-9
                px(k) = 200e-9;
                %              px_prev(k) = px(k);
            elseif px(k) < 0
                px(k) = 0;
                %              px_prev(k) = px(k);
            else
                px(k) = px(k);
            end

            v(k) = sqrt(vx(k)^2 + vy(k)^2);
            v2(k) = v(k).*v(k);

            vx_total = vx_total + vx(k);        % Drift velocity x
            vy_total = vy_total + vy(k);        % Drift velocity y
        end
        vx_total_alt = sum(vx);
```

```
        vx_drift = 1/N * vx_total;
        vy_drift = 1/N * vy_total;
        Jx = C.q_0 * Conc * vx_drift;   %Concentration = 1e19 per m^-2
        Ix_prev = Ix;
        Ix = Jx * W;
        Ix_total(t) = Ix;



    end


    Iavg(J) = mean(Ix_total);
    %bottleneck(J) = 20e-9 - (2*g) + 2e-9;


end

figure(1)
plot(Vin,Iavg)
xlabel('Input Voltage (V)')
ylabel('Average Current (A)')
title('Average Current VS Input Voltage')
saveas(gcf,'Figure1')
```



Section 1.2: Calculating the resistance of R3

The solved resistance of R3 was then found to be approximately 51.32 ohms, found through a linear fit of the plot.

```
linFit = polyfit(Vin,Iavg,1);
G3 = linFit(1);
```

```
R3 = 1/G3;

fprintf('\nThe resistance of R3 is %f ohms.',R3);
clearvars -except R3
```

```
The resistance of R3 is 51.312778 ohms.
```

Section 1.3: The C and G matricies

Using code from PA7, the C and G matricies of the circuit were formed are are displayed below.

```
%-----------------------------------------------------------------------
% Reusing Modified Nodal Analysis Code for Circuit - DC Case
%-----------------------------------------------------------------------

global G b C
nodes = 6;
G = sparse(nodes,nodes);
C = sparse(nodes,nodes);
b = sparse(nodes,1);

% MNA setup
Vin = 1;
Vprobe = 0;
R1 = 1;
R2 = 2;
%R3 = 10; %R3 found in part 1
R4 = 0.1;
R5 = 1000;
C1 = 0.25;
L1 = 0.2;
alpha = 100;

cap(1,2,C1);
res(1,2,R1);
res(2,0,R2);
res(3,6,R3);
res(4,5,R4);
res(5,0,R5);
xr = vol(6,0,Vprobe);
ind(2,3,L1);
vol(1,0,Vin);
vcvs(4,0,xr,0,alpha);

w = 0;
s = j*w;

A = G + s*C;
A0 = full(A);
G0 = full(G);
C0 = full(C);

fprintf('\nThe G matrix is given as:\n');
```

```
disp(G0);

fprintf('\nThe C matrix is given as:\n');
disp(C0);
```

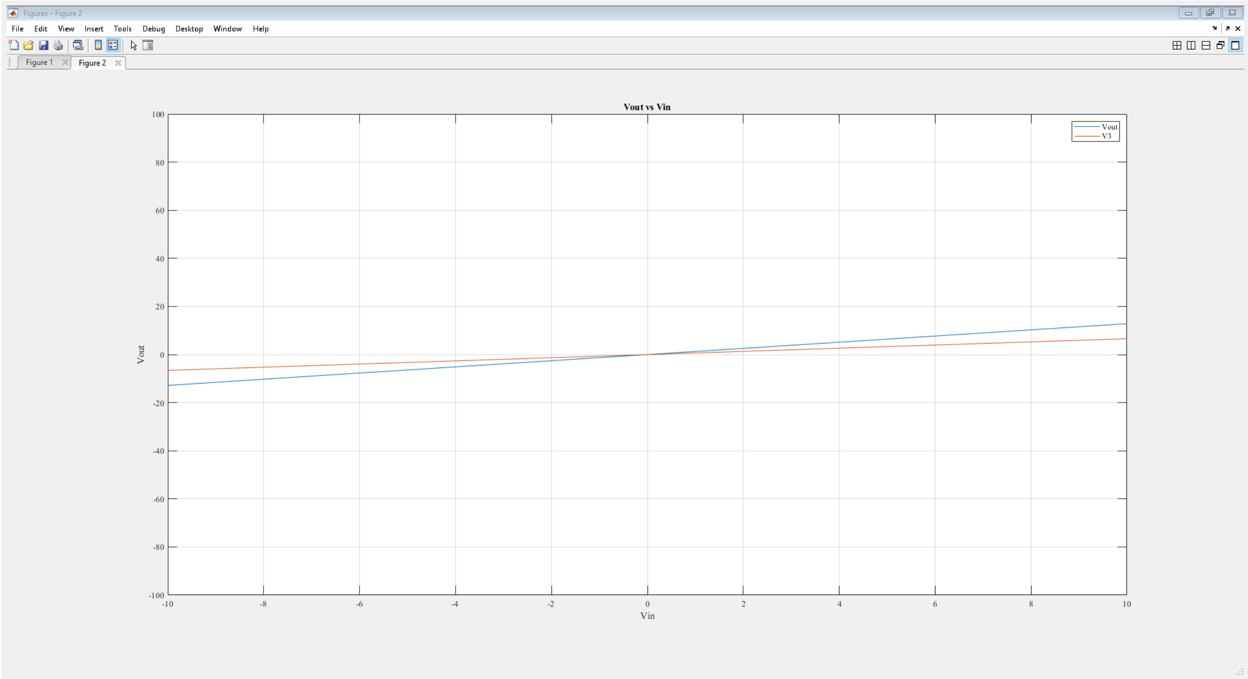
```
The G matrix is given as:
  Columns 1 through 7

    1.0000   -1.0000         0         0         0         0         0
   -1.0000    1.5000         0         0         0         0         0
         0         0    0.0195         0         0   -0.0195         0
         0         0         0   10.0000  -10.0000         0         0
         0         0         0  -10.0000   10.0010         0         0
         0         0   -0.0195         0         0    0.0195    1.0000
         0         0         0         0         0    1.0000         0
         0    1.0000   -1.0000         0         0         0         0
    1.0000         0         0         0         0         0         0
         0         0         0    1.0000         0         0 -100.0000

  Columns 8 through 10

         0    1.0000         0
    1.0000         0         0
   -1.0000         0         0
         0         0    1.0000
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0


The C matrix is given as:
  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         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         0         0
         0         0         0         0         0         0         0

  Columns 8 through 10

         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
```

Section 1.4: DC Sweep of Vin from -10V to 10V

Next, the value of Vin (at node 1) was swept from -10V to 10V and the resulting output at node 3 and the output were plotted.

```
V0 = linspace(-10,10,21);
b0 = sparse((width(G)),width(V0));
for i = 1:width(V0)
    b0(9,i) = V0(i);
end

x = sparse((width(G)),width(V0));

for j = 1:width(V0)
    x(:,j) = (G + s*C) \ b0(:,j);
end

figure(2)
plot(V0,x(5,:))
hold on
plot(V0,x(3,:))
grid on
axis([-10 10 -100 100])
title('Vout vs Vin')
xlabel('Vin')
ylabel('Vout')
legend('Vout','V3')
saveas(gcf,'Figure2')
hold off
```

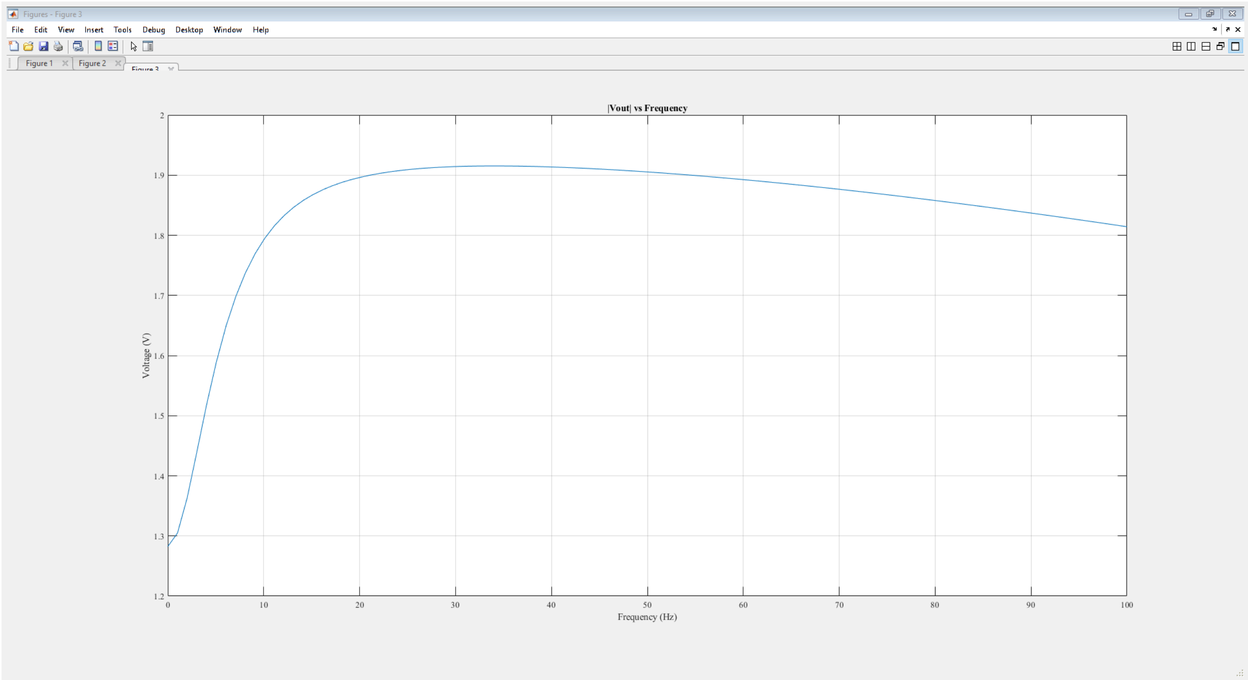


Section 1.5: The AC Case; Vout as a function of w

Vout was then plotted as a function of w (omega), where omega was swept from 0Hz to 100Hz over 100 points.

```
%-------------------------------------------------------------------------
% AC Case
%-------------------------------------------------------------------------

clearvars -except R3
global G b C
nodes = 6;
G = sparse(nodes,nodes);
C = sparse(nodes,nodes);
b = sparse(nodes,1);

% MNA setup
Vin = 1;
Vprobe = 0;
R1 = 1;
R2 = 2;
%R3 = 10;
R4 = 0.1;
R5 = 1000;
C1 = 0.25;
L1 = 0.2;
alpha = 100;

cap(1,2,C1);
res(1,2,R1);
```

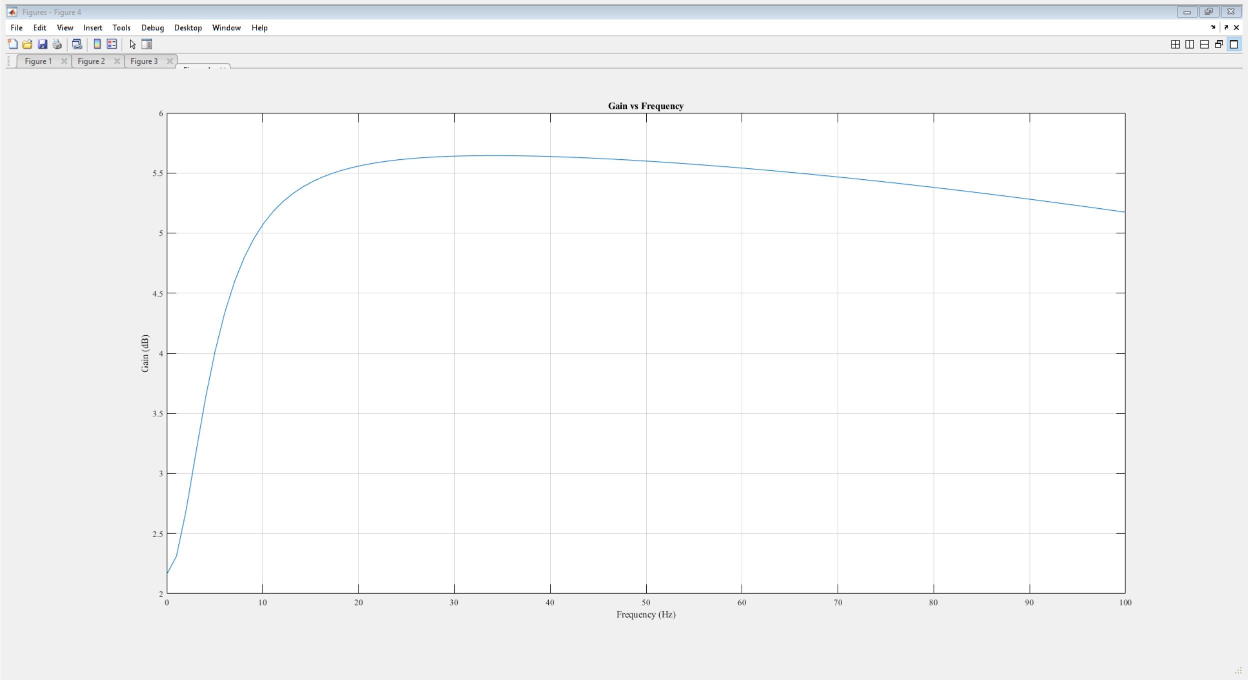
```
res(2,0,R2);
res(3,6,R3);
res(4,5,R4);
res(5,0,R5);
xr = vol(6,0,Vprobe);
ind(2,3,L1);
vol(1,0,Vin);
vcvs(4,0,xr,0,alpha);

w = linspace(0,100,100);
s = j*w;

% A = G + s.*C;
% A0 = full(A);

%V0 = linspace(-10,10,21);
% b0 = sparse((width(G)),width(w));
% for i = 1:width(w)
%     b0(6,i) = V0(i);
% end

% x = (G + s*C) \ b;
x = sparse((width(G)),width(w));

for i = 1:width(w)
    x(:,i) = (G + s(i)*C) \ b;
end

figure(3)
plot(w,abs(x(5,:)))
%hold on
%plot(w,abs(x(3,:)))
grid on
title('|Vout| vs Frequency')
xlabel('Frequency (Hz)')
ylabel('Voltage (V)')
%legend('Vout','V3')
saveas(gcf,'Figure3')
hold off
```



Section 1.6: The AC Case; Plotting Vo/Vi in dB

The gain of Vout over Vin was also plotted, in dB.

```
mag = abs(x(5,:)./x(1,:));
gain = mag2db(mag);
figure(4)
plot(w,gain)
grid on
title('Gain vs Frequency')
ylabel('Gain (dB)')
xlabel('Frequency (Hz)')
saveas(gcf,'Figure4')
```

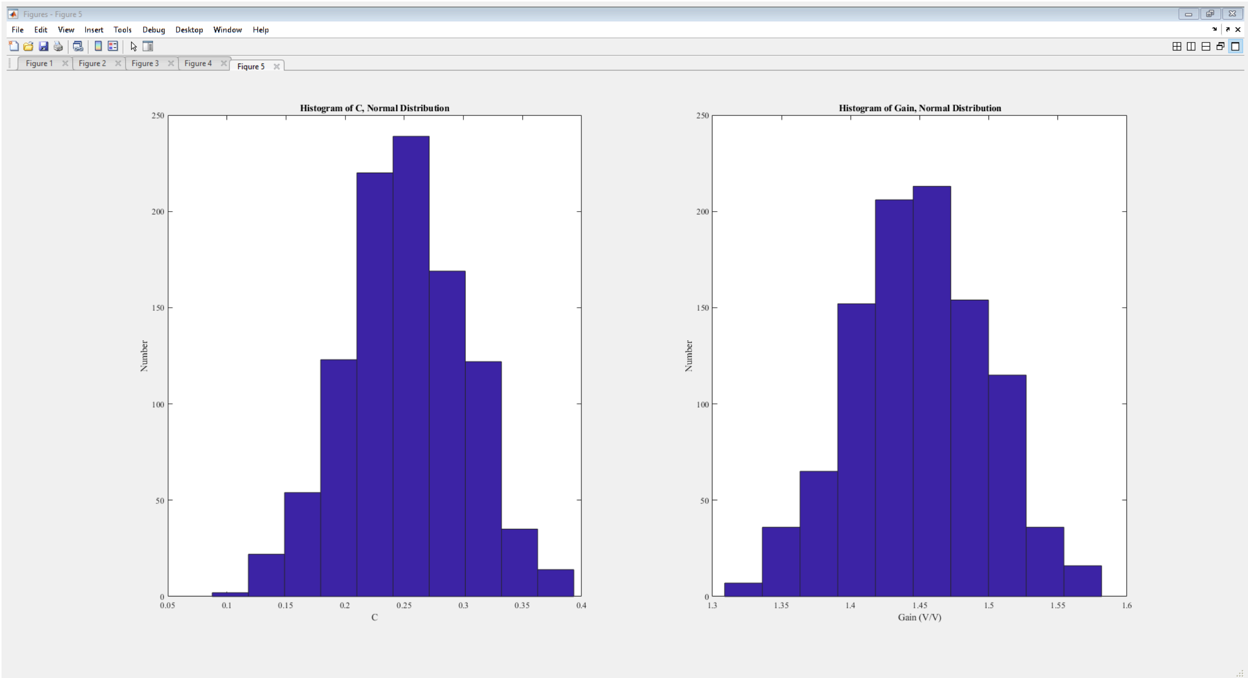


Section 1.7: The AC Case; Gain with Random Perturbations on C

Setting the capacitor C to have a normal distribution of random perturbations with a standard deviation of 0.05 and w = pi, the gain and value of C was plotted as a histogram.

```
%-------------------------------------------------------------------------
% AC Case - Normal Perturbations on C
%-------------------------------------------------------------------------

clearvars -except R3
global G b C
nodes = 6;
G = sparse(nodes,nodes);
C = sparse(nodes,nodes);
b = sparse(nodes,1);

% MNA setup
Vin = 1;
Vprobe = 0;
R1 = 1;
R2 = 2;
%R3 = 10;
R4 = 0.1;
R5 = 1000;
C1 = 0.25;
L1 = 0.2;
alpha = 100;

cap(1,2,C1);
```

```
res(1,2,R1);
res(2,0,R2);
res(3,6,R3);
res(4,5,R4);
res(5,0,R5);
xr = vol(6,0,Vprobe);
ind(2,3,L1);
vol(1,0,Vin);
vcvs(4,0,xr,0,alpha);

w = pi;
s = j*w;
std = 0.05;
mu = C1;
numR = 1000;
r = normrnd(mu,std,1,numR);

C0 = zeros(width(G),length(G),numR);
for i = 1:width(r)
    C(1:2,1:2) = 0;
    cap(1,2,r(i));
    C0(:,:,i) = C;
end

x = sparse((width(G)),numR);
for i = 1:numR
    x(:,i) = (G + s*C0(:,:,i)) \ b;
end

mag = abs(x(5,:)./x(1,:));
gain = mag2db(mag);

figure(5)
subplot(1,2,1);
hist(r)
title('Histogram of C, Normal Distribution')
xlabel('C')
ylabel('Number')
subplot(1,2,2);
hist(mag,10)
title('Histogram of Gain, Normal Distribution')
xlabel('Gain (V/V)')
ylabel('Number')
saveas(gcf,'Figure5')
```



Section 2: Transient Circuit Simulation

In Section 2, the time frequency domain solution method of the circuit would be replaced by a time domain solution, using the Backwards Euler method. Additionally, the circuit takes the form of a series bandpass RLC circuit, due to the series inductance and capacitance. The frequency response would be expected to have a high peak magnitude where the frequency passes through the filter.

Section 2.1: Vin and Vout from Numerical Time Domain Solution

Using the Backwards Euler method, the numerical solution of the circuit was found for three inputs: a step function, a a sin function, and a gaussian function. The transient analysis of the output was then plotted.

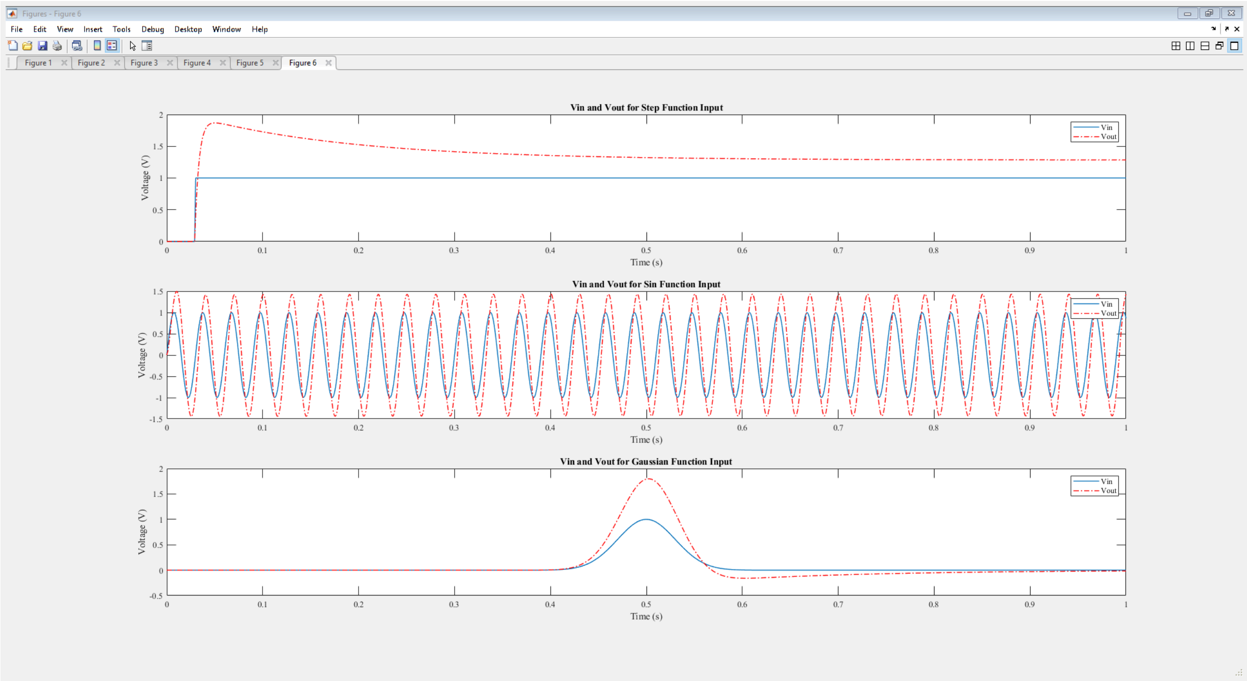
```
clearvars -except R3
% close all
% clc
%set(0,'DefaultFigureWindowStyle','docked')
set(0,'defaultaxesfontsize',10)
set(0,'defaultaxesfontname','Times New Roman')
set(0,'DefaultLineLineWidth', 0.5);

%-----------------------------------------------------------------------------
% Modified Nodal Analysis - Transient Analysis
%-----------------------------------------------------------------------------

global G b C
nodes = 6;
outNode = 5;
G = sparse(nodes,nodes);
```

```
C = sparse(nodes,nodes);
b = sparse(nodes,1);

N = 1000; %num points
t_vec_1 = linspace(0,1,N);
t_vec_2 = linspace(0+0.06,1-0.06,1000-120);
t = 0;
u_t = 0;
v_t = 0;
w_t = 0;

f = 1/0.03;
A1 = 1;
w = 2*pi*f;

X = t_vec_2;
A2 = 1;
mu = 0.5;
sigma = 0.03;
delay = 0.06;


gaussian = A2*exp(-((X-mu).^2/(2*sigma.^2)));

z = 0;

for k = 1:numel(t_vec_1)
    t = t_vec_1(k);
    %input 1
    if t < 0.03
        u_t(k) = 0;
    else
        u_t(k) = 1;
    end
    %input 2
    v_t(k) = A1*sin(w*t);
    %input 3
    if t < 0.06
        w_t(k) = 0;
    elseif t >= 0.06 && t < 0.94
        w_t(k) = gaussian(k-60);
    elseif t >= 0.94
        w_t(k) = 0;
    else
        w_t(k) = 0;
    end


end

% plot(t_vec_1,u_t);
% hold on
% plot(t_vec_1,v_t);
% plot(t_vec_1,w_t);
```

```
% hold off




%------------------------------------------------------------------------
% MNA setup
%------------------------------------------------------------------------
Vin = 1;
Vprobe = 0;
R1 = 1;
R2 = 2;
% R3 = 50;
%R3 = 123.346641;
R4 = 0.1;
R5 = 1000;
C1 = 0.25;
L1 = 0.2;
alpha = 100;

cap(1,2,C1);
res(1,2,R1);
res(2,0,R2);
res(3,6,R3);
res(4,5,R4);
res(5,0,R5);
xr = vol(6,0,Vprobe);
ind(2,3,L1);
vol(1,0,Vin);
vcvs(4,0,xr,0,alpha);

b1 = b*u_t;
b2 = b*v_t;
b3 = b*w_t;

%------------------------------------------------------------------------
% FDM Solution: Backwards Euler with Timestep 0.001
%------------------------------------------------------------------------
h = 0.001;
x = sparse(width(G),numel(t_vec_1));
C_h = C*(1/h);

for n=1:1000
    if n == 1000
        break;
    end
    BE_LHS = C_h + G;
    BE_RHS = C_h*x(:,n) + b1(:,n+1);
    [L, U, P, Q]= lu( sparse(BE_LHS) , 0.1 );
    Z = L \(P* sparse(BE_RHS));
    Y = U \ Z;
    x(:,n+1) = Q*Y;
```

```
end
BE1 = x;
for n=1:1000
    if n == 1000
        break;
    end
    BE_LHS = C_h + G;
    BE_RHS = C_h*x(:,n) + b2(:,n+1);
    [L, U, P, Q]= lu( sparse(BE_LHS) , 0.1 );
    Z = L \(P* sparse(BE_RHS));
    Y = U \ Z;
    x(:,n+1) = Q*Y;
end
BE2 = x;
for n=1:1000
    if n == 1000
        break;
    end
    BE_LHS = C_h + G;
    BE_RHS = C_h*x(:,n) + b3(:,n+1);
    [L, U, P, Q]= lu( sparse(BE_LHS) , 0.1 );
    Z = L \(P* sparse(BE_RHS));
    Y = U \ Z;
    x(:,n+1) = Q*Y;
end
BE3 = x;



figure(6)
subplot(3,1,1);
plot(t_vec_1, u_t,'LineWidth',1);
hold on;
plot(t_vec_1, BE1(outNode,:),'LineStyle','-.', 'color','r','LineWidth',1);
title('Vin and Vout for Step Function Input')
ylabel('Voltage (V)')
xlabel('Time (s)')
legend('Vin','Vout')
hold off

subplot(3,1,2);
plot(t_vec_1, v_t,'LineWidth',1);
hold on;
plot(t_vec_1, BE2(outNode,:),'LineStyle','-.', 'color','r','LineWidth',1);
title('Vin and Vout for Sin Function Input')
ylabel('Voltage (V)')
xlabel('Time (s)')
legend('Vin','Vout')
hold off

subplot(3,1,3);
plot(t_vec_1, w_t,'LineWidth',1);
hold on;
plot(t_vec_1, BE3(outNode,:),'LineStyle','-.', 'color','r','LineWidth',1);
```

```
title('Vin and Vout for Gaussian Function Input')
ylabel('Voltage (V)')
xlabel('Time (s)')
legend('Vin','Vout')
hold off
saveas(gcf,'Figure6')
```

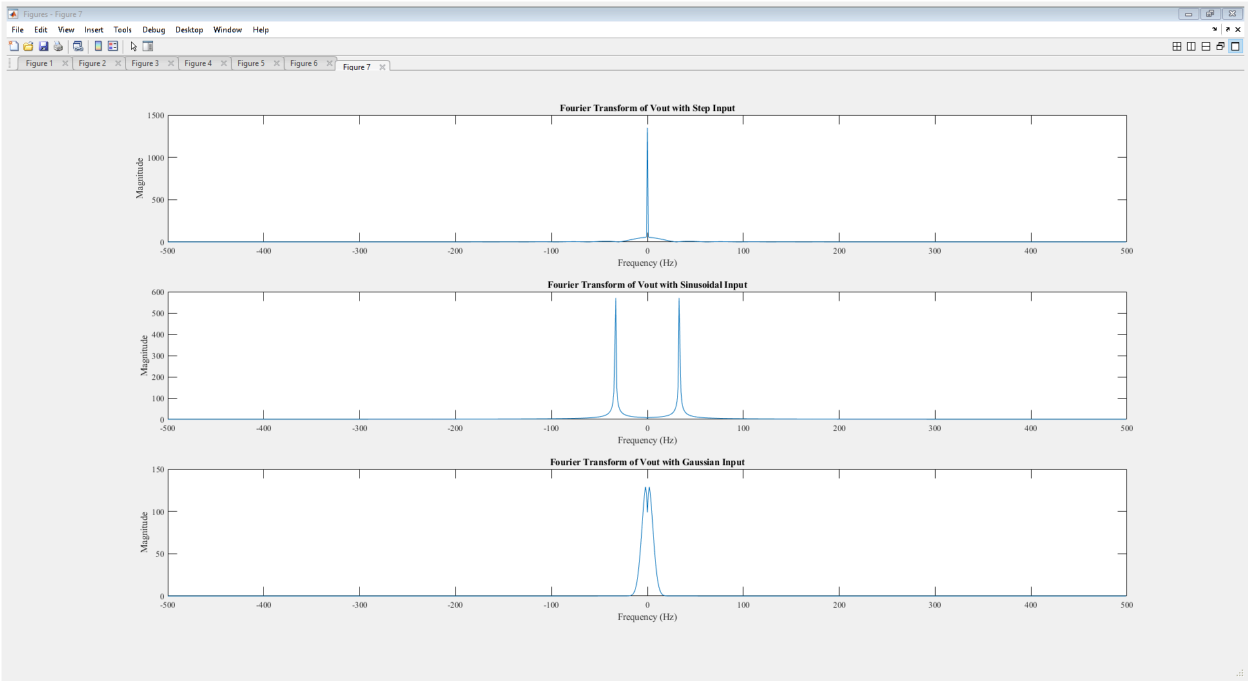


Section 2.2: Fourier Transform of the Output Solution

The Fourier transform of the output signal was then plotted for each of the three inputs. It was also noted that an increase in timestep for the Backwards Euler solution led to a decrease in accuracy of the solution.

```
Fs = 1/h;
dF = Fs/N;
f = -Fs/2:dF:Fs/2 - dF;

%F_vec_1 = 1./t_vec_1;
figure(7)
subplot(3,1,1);
fullBE1 = full(BE1(outNode,:));
FT1 = fft(fullBE1);
plot(f,fftshift(abs(FT1)))
title('Fourier Transform of Vout with Step Input')
ylabel('Magnitude')
xlabel('Frequency (Hz)')

subplot(3,1,2);
fullBE2 = full(BE2(outNode,:));
FT2 = fft(fullBE2);
```

```
plot(f,fftshift(abs(FT2)))
title('Fourier Transform of Vout with Sinusoidal Input')
ylabel('Magnitude')
xlabel('Frequency (Hz)')

subplot(3,1,3);
fullBE3 = full(BE3(outNode,:));
FT3 = fft(fullBE3);
plot(f,fftshift(abs(FT3)))
title('Fourier Transform of Vout with Gaussian Input')
ylabel('Magnitude')
xlabel('Frequency (Hz)')
saveas(gcf,'Figure7')
```



Section 3: Circuit Analysis Including Noise

Following the circuit analysis, a noise source and capacitance was then added in parallel with the resistor R3.

Section 3.1: The Updated C Matrix

The addition of the noise capacitor Cn resulted in an update to the C matrix. The updated matrix is given below.

```
clearvars -except R3
% close all
% clc
%set(0,'DefaultFigureWindowStyle','docked')
set(0,'defaultaxesfontsize',10)
set(0,'defaultaxesfontname','Times New Roman')
set(0,'DefaultLineLineWidth', 0.5);

%--------------------------------------------------------------------------
```

```
% Modified Nodal Analysis - Transient Analysis
%-------------------------------------------------------------------------

global G b C
nodes = 6;
outNode = 5;
inNode = 9;
N = 1000; %num points
G = sparse(nodes,nodes);
C = sparse(nodes,nodes);
b = sparse(nodes,1);


t_vec_1 = linspace(0,1,1000);
t_vec_2 = linspace(0+0.06,1-0.06,1000-120);
t = 0;
u_t = 0;
v_t = 0;
w_t = 0;

f = 1/0.03;
A1 = 1;
w = 2*pi*f;

X = t_vec_2;
A2 = 1;
mu = 0.5;
sigma = 0.03;
delay = 0.06;


gaussian = A2*exp(-((X-mu).^2/(2*sigma.^2)));

z = 0;

for k = 1:numel(t_vec_1)
    t = t_vec_1(k);
    %input 1
    if t < 0.03
        u_t(k) = 0;
    else
        u_t(k) = 1;
    end
    %input 2
    v_t(k) = A1*sin(w*t);
    %input 3
    if t < 0.06
        w_t(k) = 0;
    elseif t >= 0.06 && t < 0.94
        w_t(k) = gaussian(k-60);
    elseif t >= 0.94
        w_t(k) = 0;
    else
        w_t(k) = 0;
```

```
    end


end

% plot(t_vec_1,u_t);
% hold on
% plot(t_vec_1,v_t);
% plot(t_vec_1,w_t);
% hold off






%-------------------------------------------------------------------------
% MNA setup
%-------------------------------------------------------------------------
Vin = 1;
Vprobe = 0;
R1 = 1;
R2 = 2;
%R3 = 123.346641;
% R3 = 50;
R4 = 0.1;
R5 = 1000;
C1 = 0.25;
L1 = 0.2;
alpha = 100;
In = 0.001;
Cn = 0.00001;

cap(1,2,C1);
res(1,2,R1);
res(2,0,R2);
res(3,6,R3);
res(4,5,R4);
res(5,0,R5);
xr = vol(6,0,Vprobe);
ind(2,3,L1);
vol(1,0,Vin);
vcvs(4,0,xr,0,alpha);
cur(4,0,6);
cap(4,0,Cn);

fullC = full(C);
fprintf('\nThe updated C matrix is given by:\n');
disp(fullC);
```

The updated C matrix is given by:
Columns 1 through 7

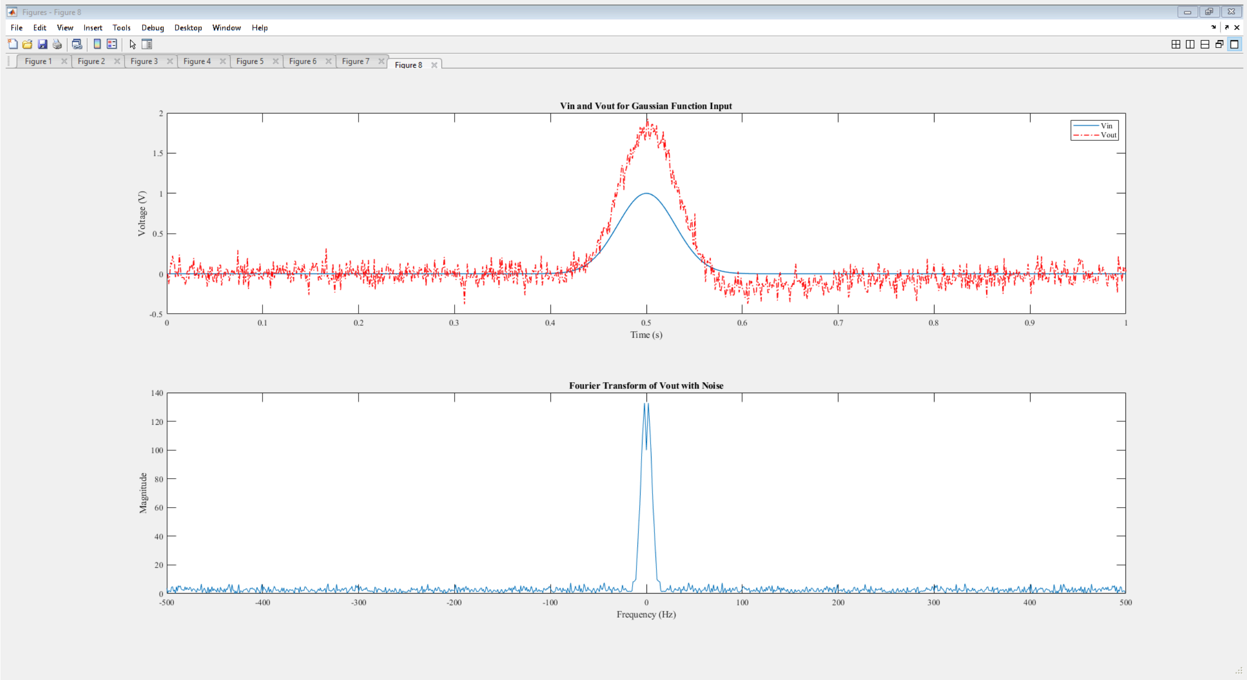
```
   0.2500   -0.2500         0         0         0         0         0
  -0.2500    0.2500         0         0         0         0         0
        0         0         0         0         0         0         0
        0         0         0    0.0000         0         0         0
        0       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

  Columns 8 through 10

        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
```

Section 3.2: Plotting Vout with Included Noise

The time domain and frequency domain plots of Vout with included noise were then formed.

```
% b1 = b*u_t;
% b2 = b*v_t;
b3 = zeros(10,1000);
b3(inNode,:) = w_t;

% r = normrnd(mu,std,1000);
for i = 1:1000
    r = In*randn();
     b3(4,i) = r;
     b3(6,i) = -r;
    %b3(4,i) = 1;
end
% make b*wt and In into 2 different things

%--------------------------------------------------------------------------
% FDM Solution: Backwards Euler with Timestep 0.001
%--------------------------------------------------------------------------
h = 0.001;
x = sparse(width(G),numel(t_vec_1));
C_h = C*(1/h);

for n=1:1000
    if n == 1000
        break;
```

```
    end
    BE_LHS = C_h + G;
    BE_RHS = C_h*x(:,n) + b3(:,n+1);
    [L, U, P, Q]= lu( sparse(BE_LHS) , 0.1 );
    Z = L \(P* sparse(BE_RHS));
    Y = U \ Z;
    x(:,n+1) = Q*Y;
end
BE3 = x;


Fs = 1/h;
dF = Fs/N;
f = -Fs/2:dF:Fs/2 - dF;

figure(8)
subplot(2,1,1);
plot(t_vec_1, w_t,'LineWidth',1);
hold on;
plot(t_vec_1, BE3(outNode,:),'LineStyle','-.', 'color','r','LineWidth',1);
title('Vin and Vout for Gaussian Function Input')
ylabel('Voltage (V)')
xlabel('Time (s)')
legend('Vin','Vout')
hold off

subplot(2,1,2);
fullBE3 = full(BE3(outNode,:));
FT3 = fft(fullBE3);
plot(f,fftshift(abs(FT3)))
title('Fourier Transform of Vout with Noise')
ylabel('Magnitude')
xlabel('Frequency (Hz)')
saveas(gcf,'Figure8')
```



Section 3.3: Varying Cn

Next, Cn was changed between 3 different values of 0.0001, 0.00001, and 0.000001 Farads to view the effects on the response. It was found that the capacitance had a large effect on the non-noise voltage, but minimal effect on the noise itself.

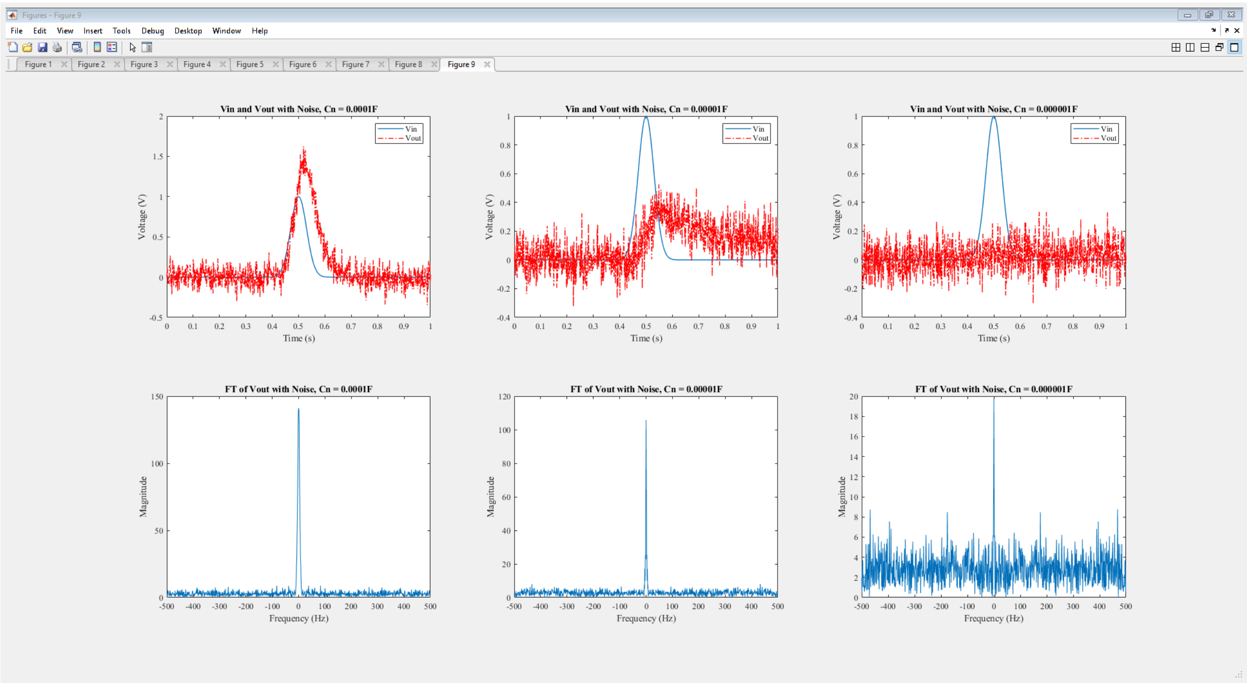
```
clearvars -except R3
% close all
% clc
%set(0,'DefaultFigureWindowStyle','docked')
set(0,'defaultaxesfontsize',10)
set(0,'defaultaxesfontname','Times New Roman')
set(0,'DefaultLineLineWidth', 0.5);

H = [0.0001,0.00001,0.000001];
for J = 1:3
clearvars -except J H R3 BE3 N
%-------------------------------------------------------------------------
% Modified Nodal Analysis - Transient Analysis
%-------------------------------------------------------------------------

global G b C
nodes = 6;
outNode = 5;
inNode = 9;
N = 1000; %num points
G = sparse(nodes,nodes);
C = sparse(nodes,nodes);
b = sparse(nodes,1);
```

```
t_vec_1 = linspace(0,1,1000);
t_vec_2 = linspace(0+0.06,1-0.06,1000-120);
t = 0;
u_t = 0;
v_t = 0;
w_t = 0;

f = 1/0.03;
A1 = 1;
w = 2*pi*f;

X = t_vec_2;
A2 = 1;
mu = 0.5;
sigma = 0.03;
delay = 0.06;


gaussian = A2*exp(-((X-mu).^2/(2*sigma.^2)));

z = 0;

for k = 1:numel(t_vec_1)
    t = t_vec_1(k);
    %input 1
    if t < 0.03
        u_t(k) = 0;
    else
        u_t(k) = 1;
    end
    %input 2
    v_t(k) = A1*sin(w*t);
    %input 3
    if t < 0.06
        w_t(k) = 0;
    elseif t >= 0.06 && t < 0.94
        w_t(k) = gaussian(k-60);
    elseif t >= 0.94
        w_t(k) = 0;
    else
        w_t(k) = 0;
    end


end

% plot(t_vec_1,u_t);
% hold on
% plot(t_vec_1,v_t);
% plot(t_vec_1,w_t);
% hold off
```

```
%-------------------------------------------------------------------------
% MNA setup
%-------------------------------------------------------------------------
Vin = 1;
Vprobe = 0;
R1 = 1;
R2 = 2;
%R3 = 123.346641;
% R3 = 50;
R4 = 0.1;
R5 = 1000;
C1 = 0.25;
L1 = 0.2;
alpha = 100;
In = 0.001;
Cn = H(J);

cap(1,2,C1);
res(1,2,R1);
res(2,0,R2);
res(3,6,R3);
res(4,5,R4);
res(5,0,R5);
xr = vol(6,0,Vprobe);
ind(2,3,L1);
vol(1,0,Vin);
vcvs(4,0,xr,0,alpha);
cur(4,0,6);
cap(4,0,Cn);



% b1 = b*u_t;
% b2 = b*v_t;
b3 = zeros(10,1000);
b3(inNode,:) = w_t;

% r = normrnd(mu,std,1000);
for i = 1:1000
    r = In*randn();
     b3(4,i) = r;
     b3(6,i) = -r;
    %b3(4,i) = 1;
end
% make b*wt and In into 2 different things

%-------------------------------------------------------------------------
% FDM Solution: Backwards Euler with Timestep 0.001
%-------------------------------------------------------------------------
h = H(J);
x = sparse(width(G),numel(t_vec_1));
```

```
C_h = C*(1/h);

for n=1:1000
    if n == 1000
        break;
    end
    BE_LHS = C_h + G;
    BE_RHS = C_h*x(:,n) + b3(:,n+1);
    [L, U, P, Q]= lu( sparse(BE_LHS) , 0.1 );
    Z = L \(P* sparse(BE_RHS));
    Y = U \ Z;
    x(:,n+1) = Q*Y;
end
BE3{J} = x;

end

Fs = 1/0.001;
dF = Fs/N;
f = -Fs/2:dF:Fs/2 - dF;

S1 = BE3{1};
S2 = BE3{2};
S3 = BE3{3};

figure(9)
subplot(2,3,1);
plot(t_vec_1, w_t,'LineWidth',1)
hold on;
plot(t_vec_1, S1(outNode,:),'LineStyle','-.', 'color','r','LineWidth',1);
title('Vin and Vout with Noise, Cn = 0.0001F')
ylabel('Voltage (V)')
xlabel('Time (s)')
legend('Vin','Vout')
hold off

subplot(2,3,2);
plot(t_vec_1, w_t,'LineWidth',1)
hold on;
plot(t_vec_1, S2(outNode,:),'LineStyle','-.', 'color','r','LineWidth',1);
title('Vin and Vout with Noise, Cn = 0.00001F')
ylabel('Voltage (V)')
xlabel('Time (s)')
legend('Vin','Vout')
hold off

subplot(2,3,3);
plot(t_vec_1, w_t,'LineWidth',1)
hold on;
plot(t_vec_1, S3(outNode,:),'LineStyle','-.', 'color','r','LineWidth',1);
title('Vin and Vout with Noise, Cn = 0.000001F')
ylabel('Voltage (V)')
xlabel('Time (s)')
legend('Vin','Vout')
```

```
hold off

subplot(2,3,4);
fullS1 = full(S1(outNode,:));
FTS1 = fft(fullS1);
plot(f,fftshift(abs(FTS1)))
title('FT of Vout with Noise, Cn = 0.0001F')
ylabel('Magnitude')
xlabel('Frequency (Hz)')
hold off

subplot(2,3,5);
fullS2 = full(S2(outNode,:));
FTS2 = fft(fullS2);
plot(f,fftshift(abs(FTS2)))
title('FT of Vout with Noise, Cn = 0.00001F')
ylabel('Magnitude')
xlabel('Frequency (Hz)')
hold off

subplot(2,3,6);
fullS3 = full(S3(outNode,:));
FTS3 = fft(fullS3);
plot(f,fftshift(abs(FTS3)))
title('FT of Vout with Noise, Cn = 0.000001F')
ylabel('Magnitude')
xlabel('Frequency (Hz)')
saveas(gcf,'Figure9')
hold off
```



Section 3.4: Varying Time Step

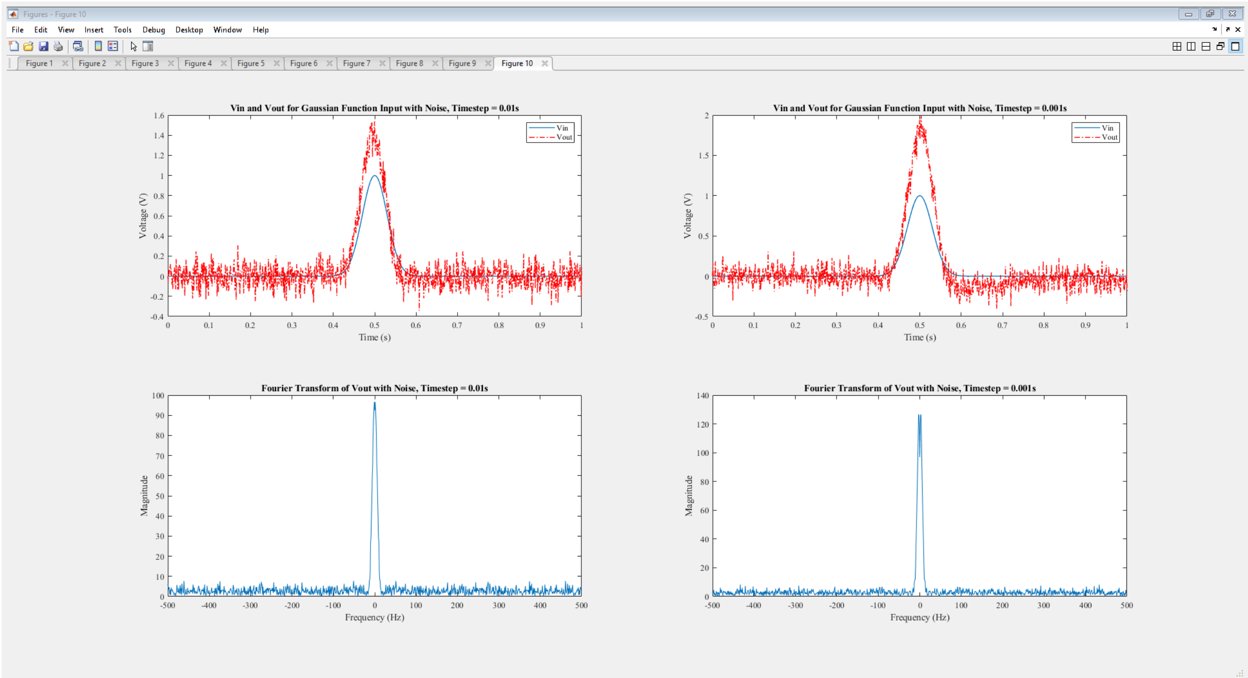
Finally, the time step was varied between 0.01s and 0.001s, and the resulting plots were found. It was seen that a larger time step resulted in a less accurate solution, as expected.

```
clearvars -except R3
% close all
% clc
%set(0,'DefaultFigureWindowStyle','docked')
set(0,'defaultaxesfontsize',10)
set(0,'defaultaxesfontname','Times New Roman')
set(0,'DefaultLineLineWidth', 0.5);

H = [0.01, 0.001];
for J = 1:2
clearvars -except J H R3 BE3 N
%---------------------------------------------------------------------------
% Modified Nodal Analysis - Transient Analysis
%---------------------------------------------------------------------------

global G b C
nodes = 6;
outNode = 5;
inNode = 9;
N = 1000; %num points
G = sparse(nodes,nodes);
C = sparse(nodes,nodes);
b = sparse(nodes,1);


t_vec_1 = linspace(0,1,1000);
t_vec_2 = linspace(0+0.06,1-0.06,1000-120);
t = 0;
u_t = 0;
v_t = 0;
w_t = 0;

f = 1/0.03;
A1 = 1;
w = 2*pi*f;

X = t_vec_2;
A2 = 1;
mu = 0.5;
sigma = 0.03;
delay = 0.06;


gaussian = A2*exp(-((X-mu).^2/(2*sigma.^2)));

z = 0;

for k = 1:numel(t_vec_1)
```

```
    t = t_vec_1(k);
    %input 1
    if t < 0.03
        u_t(k) = 0;
    else
        u_t(k) = 1;
    end
    %input 2
    v_t(k) = A1*sin(w*t);
    %input 3
    if t < 0.06
        w_t(k) = 0;
    elseif t >= 0.06 && t < 0.94
        w_t(k) = gaussian(k-60);
    elseif t >= 0.94
        w_t(k) = 0;
    else
        w_t(k) = 0;
    end


end

% plot(t_vec_1,u_t);
% hold on
% plot(t_vec_1,v_t);
% plot(t_vec_1,w_t);
% hold off





%--------------------------------------------------------------------------
% MNA setup
%--------------------------------------------------------------------------
Vin = 1;
Vprobe = 0;
R1 = 1;
R2 = 2;
%R3 = 123.346641;
% R3 = 50;
R4 = 0.1;
R5 = 1000;
C1 = 0.25;
L1 = 0.2;
alpha = 100;
In = 0.001;
Cn = 0.00001;

cap(1,2,C1);
res(1,2,R1);
res(2,0,R2);
```

```
res(3,6,R3);
res(4,5,R4);
res(5,0,R5);
xr = vol(6,0,Vprobe);
ind(2,3,L1);
vol(1,0,Vin);
vcvs(4,0,xr,0,alpha);
cur(4,0,6);
cap(4,0,Cn);



% b1 = b*u_t;
% b2 = b*v_t;
b3 = zeros(10,1000);
b3(inNode,:) = w_t;

% r = normrnd(mu,std,1000);
for i = 1:1000
    r = In*randn();
     b3(4,i) = r;
     b3(6,i) = -r;
    %b3(4,i) = 1;
end
% make b*wt and In into 2 different things

%--------------------------------------------------------------------------
% FDM Solution: Backwards Euler with Timestep 0.001
%--------------------------------------------------------------------------
h = H(J);
x = sparse(width(G),numel(t_vec_1));
C_h = C*(1/h);

for n=1:1000
    if n == 1000
        break;
    end
    BE_LHS = C_h + G;
    BE_RHS = C_h*x(:,n) + b3(:,n+1);
    [L, U, P, Q]= lu( sparse(BE_LHS) , 0.1 );
    Z = L \(P* sparse(BE_RHS));
    Y = U \ Z;
    x(:,n+1) = Q*Y;
end
BE3{J} = x;

end

Fs = 1/0.001;
dF = Fs/N;
f = -Fs/2:dF:Fs/2 - dF;

S1 = BE3{1};
S2 = BE3{2};
```

```
figure(10)
subplot(2,2,1);
plot(t_vec_1, w_t,'LineWidth',1)
hold on;
plot(t_vec_1, S1(outNode,:),'LineStyle','-.', 'color','r','LineWidth',1);
title('Vin and Vout for Gaussian Function Input with Noise, Timestep = 0.01s')
ylabel('Voltage (V)')
xlabel('Time (s)')
legend('Vin','Vout')
hold off

subplot(2,2,2);
plot(t_vec_1, w_t,'LineWidth',1)
hold on;
plot(t_vec_1, S2(outNode,:),'LineStyle','-.', 'color','r','LineWidth',1);
title('Vin and Vout for Gaussian Function Input with Noise, Timestep =
 0.001s')
ylabel('Voltage (V)')
xlabel('Time (s)')
legend('Vin','Vout')
hold off

subplot(2,2,3);
fullS1 = full(S1(outNode,:));
FTS1 = fft(fullS1);
plot(f,fftshift(abs(FTS1)))
title('Fourier Transform of Vout with Noise, Timestep = 0.01s')
ylabel('Magnitude')
xlabel('Frequency (Hz)')
hold off

subplot(2,2,4);
fullS2 = full(S2(outNode,:));
FTS2 = fft(fullS2);
plot(f,fftshift(abs(FTS2)))
title('Fourier Transform of Vout with Noise, Timestep = 0.001s')
ylabel('Magnitude')
xlabel('Frequency (Hz)')
saveas(gcf,'Figure10')
hold off
```



Section 4: Non-Linearity

For a non-linear circuit solution as presented, a new matrix would be required in the time-domain equation to represent the non-linear elements. The modifications to the equation may be represented as $C * (dx/dt) + Gx + H(x) - F(t) = 0$, where $H(x)$ is the new matrix. The purpose of this is to enable the solution via Newton-Ralphson iteration, where the non-linear equation and it's derivative (Jacobian) can be used to find the solution at the next time step, and the end of the solution can be determined by the reduction of errors in the solution compared to each previous step. To implement this in the circuit, first the DC solution of the circuit would be found using an initial assumption for the DC solution of x, and iterate until a stable solution was found. This solution would then serve as the stating point of the AC analysis, when t=0. Then, the problem can be solved using modified Backwards Euler equations or other integration methods.

Published with MATLAB® R2021b