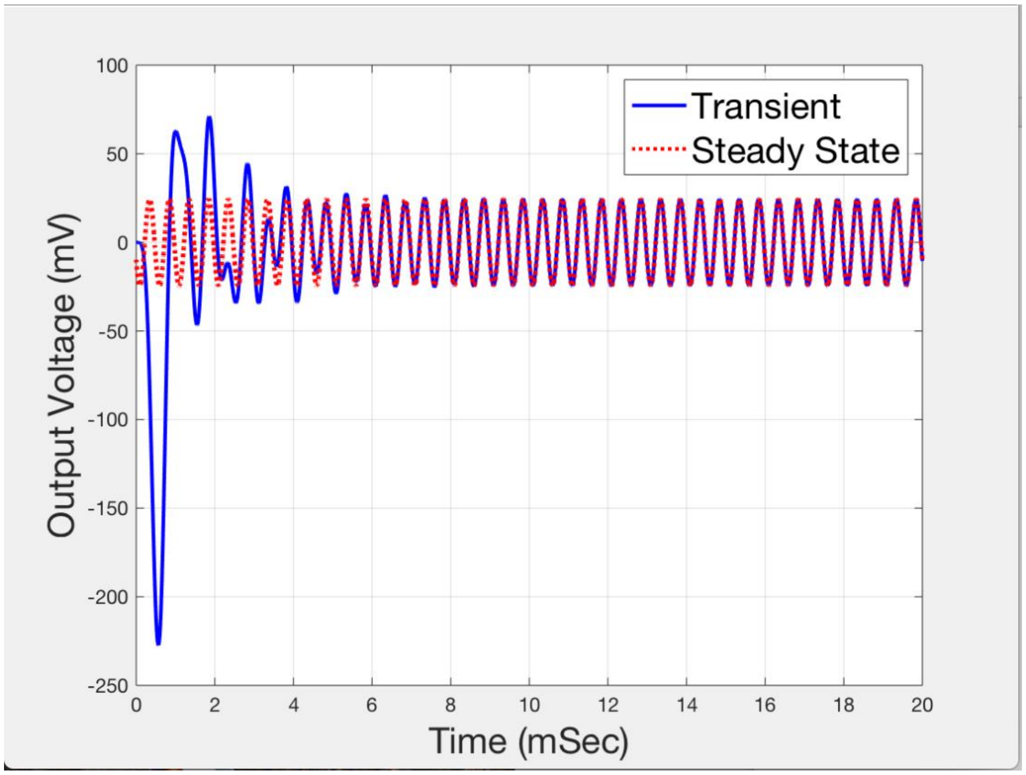
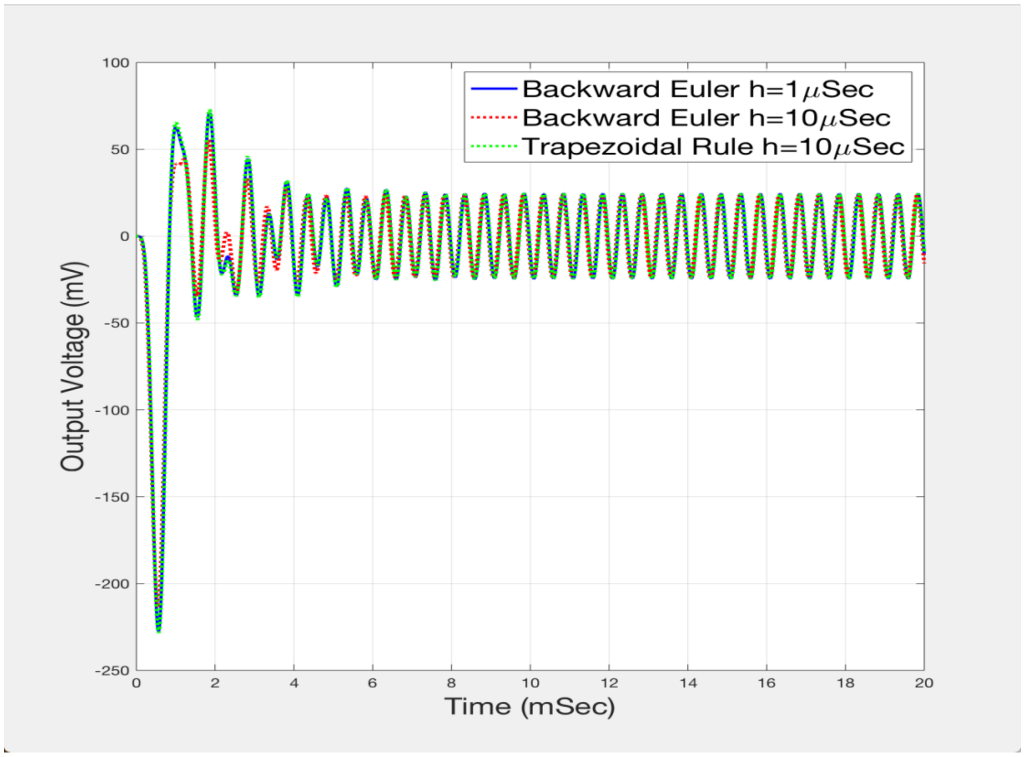
Question 1: Backward Euler



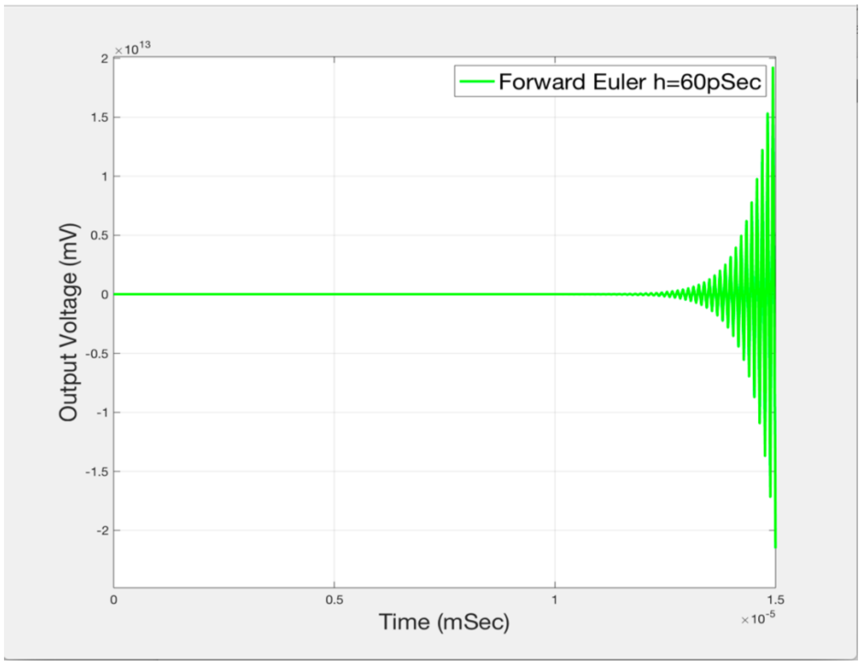
In the output figure, we can see that the transient plot is very unstable at the beginning. The transient plot begins with a scrambled state with a super low and high magnitude of output voltage around the steady state. While as time passes, the transient becomes more and more stable and approaches to steady state plot. After around 6ms, the transient plot is in steady state, and the output voltages are about from -25mV to +25mV. Finally, transient becomes a completely steady state.

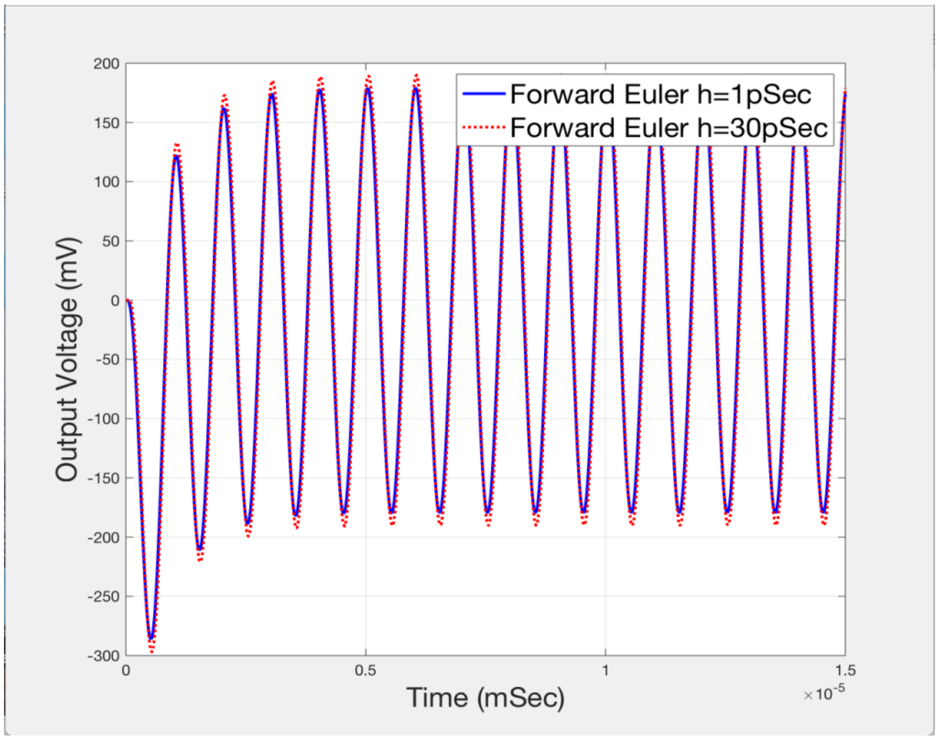
Question 2: Trapezoidal Rule

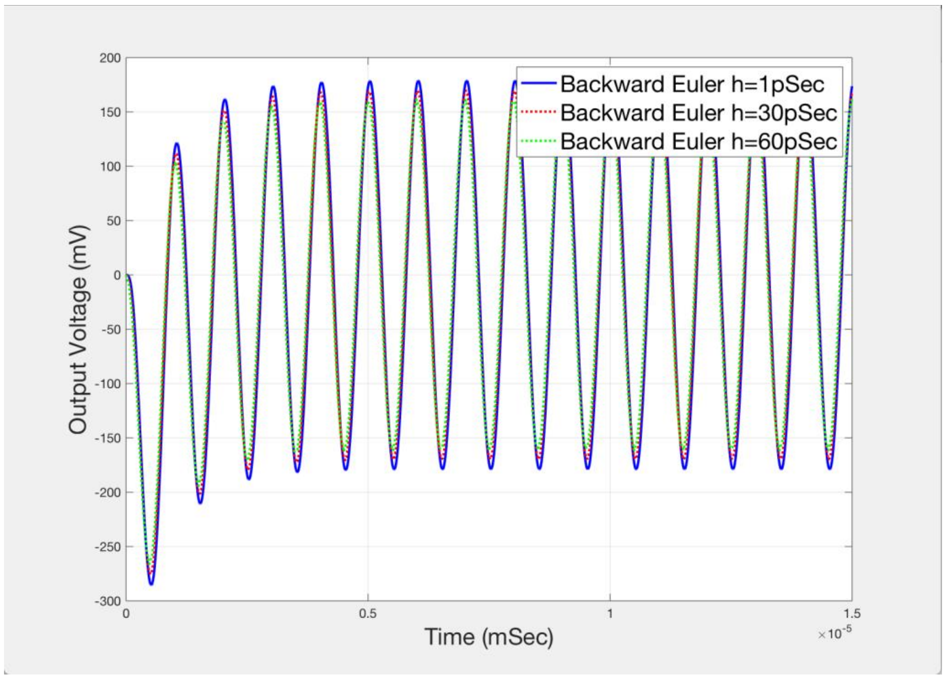


From the diagram, we can see that there are two kinds of method: Backward Euler and Trapezoidal Rule, and there are three kinds of plots: Backward Euler at h=1us and at h=10 us, and trapezoidal rule at h=10us. From the diagram, we can observe that the error of Backward Euler method depends on the step size. Similar to Question1, the three plots are all unstable at the beginning, while they become steady state after about 4ms. We can see that the plot of trapezoidal rule at h=10us is similar to Backward Euler at h=1us. Since the trapezoidal rule with a larger step size can perform the same plot as the Backward Euler with a smaller step size, the trapezoidal rule performs better. If we observe the plot of Backward Euler at h=10us, it has a larger scrambled state than the plot of trapezoidal rule with the same step size h=10us, though it also approves that trapezoidal rule performs better. Therefore, trapezoidal rule has a higher stability condition than the Back Euler method.

Question 3: Forward Euler

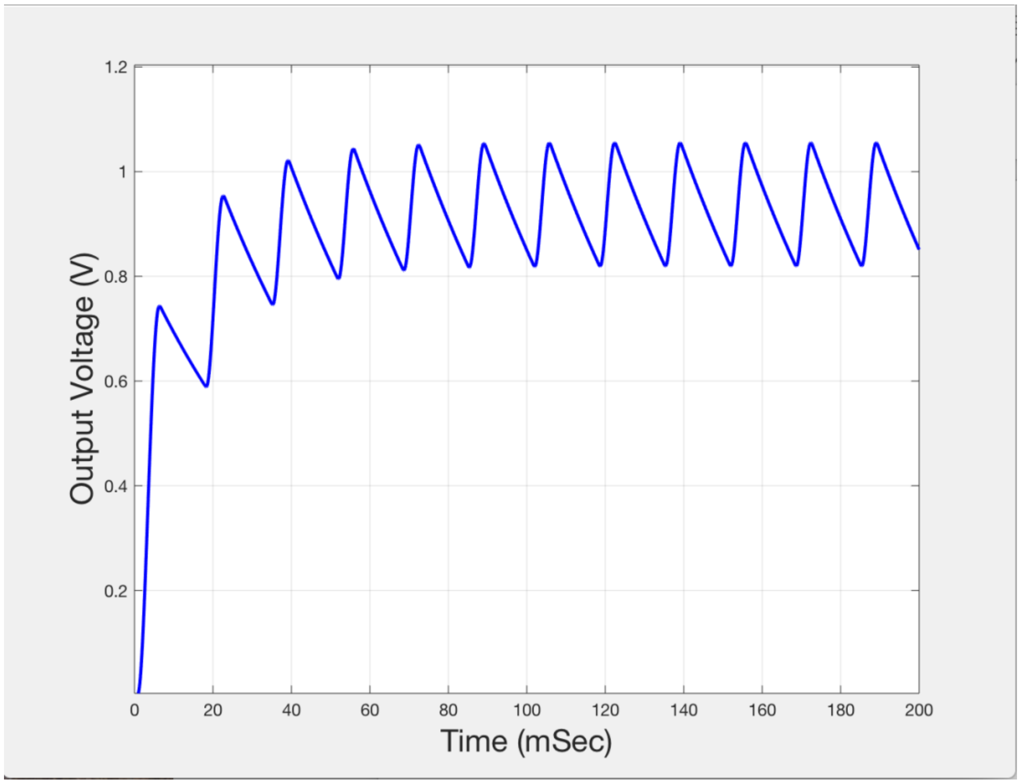






The diagram shows the stability of Backward Euler and Forward Euler at h =1ps, h = 30 ps, and h = 60ps, respectively. For Backward Euler, it will reach stable state at h=1ps, h = 30ps and h = 60ps. While for Forward Euler, it will reach stable state only at h=1ps and h=30ps. When h = 60ps, the Forward Euler will be diverse and unstable in the end. Connecting what we have learned from class, the Backward Euler method will always be stable while Forward Euler is not always stable. The Forward Euler will be unstable in some conditions.

Question 4: Nonlinear Circuits.



```
function [tpoints,r] = transient_beuler(t1,t2,h,out)
% [tpoints,r] = beuler(t1,t2,h,out)
% Perform transient analysis for LINEAR Circuits using Backward Euler
% assume zero initial condition.
% Inputs:  t1 = starting time point (typically 0)
%          t2 = ending time point
%          h  = step size
%          out = output node
% Outputs  tpoints = are the time points at which the output
%                    was evaluated
%          r       = value of the response at above time points
% plot(tpoints,r) should produce a plot of the transient response

global G C b

% tpoints is a vector containing all the timepoints from t1 to t2 with step
% size h
tpoints = t1:h:t2;
zs = zeros(size(b));
r = zeros(1,length(tpoints));

for x = 1:length(tpoints)-1
    zs = inv(G+C/h)*(BTime(tpoints(x+1))+(C/h)*zs);

    r(x+1)=zs(out);
end
```

```
function [tpoints,r] = transient_trapez(t1,t2,h,out)
% [tpoints,r] = Transient_trapez(t1,t2,h,out)
% Perform Transient Analysis using the Trapezoidal Rule for LINEAR
% circuits.
% assume zero initial condition.
% Inputs:  t1 = starting time point (typically 0)
%          t2 = ending time point
%          h  = step size
%          out = output node
% Outputs  tpoints = are the time points at which the output
%                    was evaluated
%          r       = value of the response at above time points
% plot(tpoints,r) should produce a plot of the transient response
global G C b

zs1 = zeros(size(G,2),1);
zs = zeros(size(b));

tpoints = t1:h:t2;
%same as beuler
r = zeros(1,length(tpoints));

for x = 1:length(tpoints)-1
    zs = inv(G+2*C/h)*(BTime(tpoints(x+1))+BTime(tpoints(x))+(2*C/h-G)*zs);

    r(x+1)=zs(out);
end
```

```
function [tpoints,r] = transient_feuler(t1,t2,h,out)
% [tpoints,r] = Transient_feuler(t1,t2,h,out)
% Perform Transient analysis for LINEAR circuit using Forward Euler
% This function assumes the C matrix is invertible and will not work
% for circuits where C is not invertible.
% assume zero initial condition.
% Inputs:  t1 = starting time point (typically 0)
%          t2 = ending time point
%          h  = step size
%          out = output node
% Outputs  tpoints = are the time points at which the output
%                    was evaluated
%          r       = value of the response at above time points
% plot(tpoints,r) should produce a plot of the transient response
global G C b

zs1 = zeros(size(G,2),1);
zs = zs1;

tpoints = t1:h:t2;
%same as beuler
r = zeros(1,length(tpoints));

for x = 1:length(tpoints)-1
    zs = inv(C/h)*(BTime(tpoints(x))+(C/h-G)*zs);

    r(x+1)=zs(out);
end
```

```
function [tpoints,r] = nl_transient_beuler(t1,t2,h,out)
% [tpoints,r] = beuler(t1,t2,h,out)
% Perform transient analysis for NONLINEAR Circuits using Backward Euler
% Assume zero initial condition.
% Inputs:  t1 = starting time point (typically 0)
%          t2 = ending time point
%          h  = step size
%          out = output node
% Outputs  tpoints = are the time points at which the output
%                    was evaluated
%          r       = value of the response at above time points
% plot(tpoints,r) should produce a plot of the transient response

global G C b


tpoints = t1:h:t2;
r = zeros(1,length(tpoints));
maxer = 10^(-6);

zs = zeros(size(b));
zs2 = zeros(size(b));

for x=1:length(tpoints)-1

    x_d = intmax();

    while x_d >= maxer
        fun = f_vector(zs2);
        pi = fun+(G+C/h)*zs2-BTime(tpoints(x+1))-(C/h)*zs;

        nlJ = nlJacobian(zs2);
        pi2 = G+C/h+nlJ;

        x_d2 = -1*inv(pi2)*pi;

        zs2 = zs2+x_d2;
        x_d = norm(x_d2);
    end

    r(x+1) = zs2(out);
    zs = zs2;
end
```