ECSE 597: Circuit Simulation and Modelling
Assignment 2

Student Name: Dafne Culha
Student ID: 260785524

Department of Electrical and Computer Engineering
McGill University, Canada
November, 2021

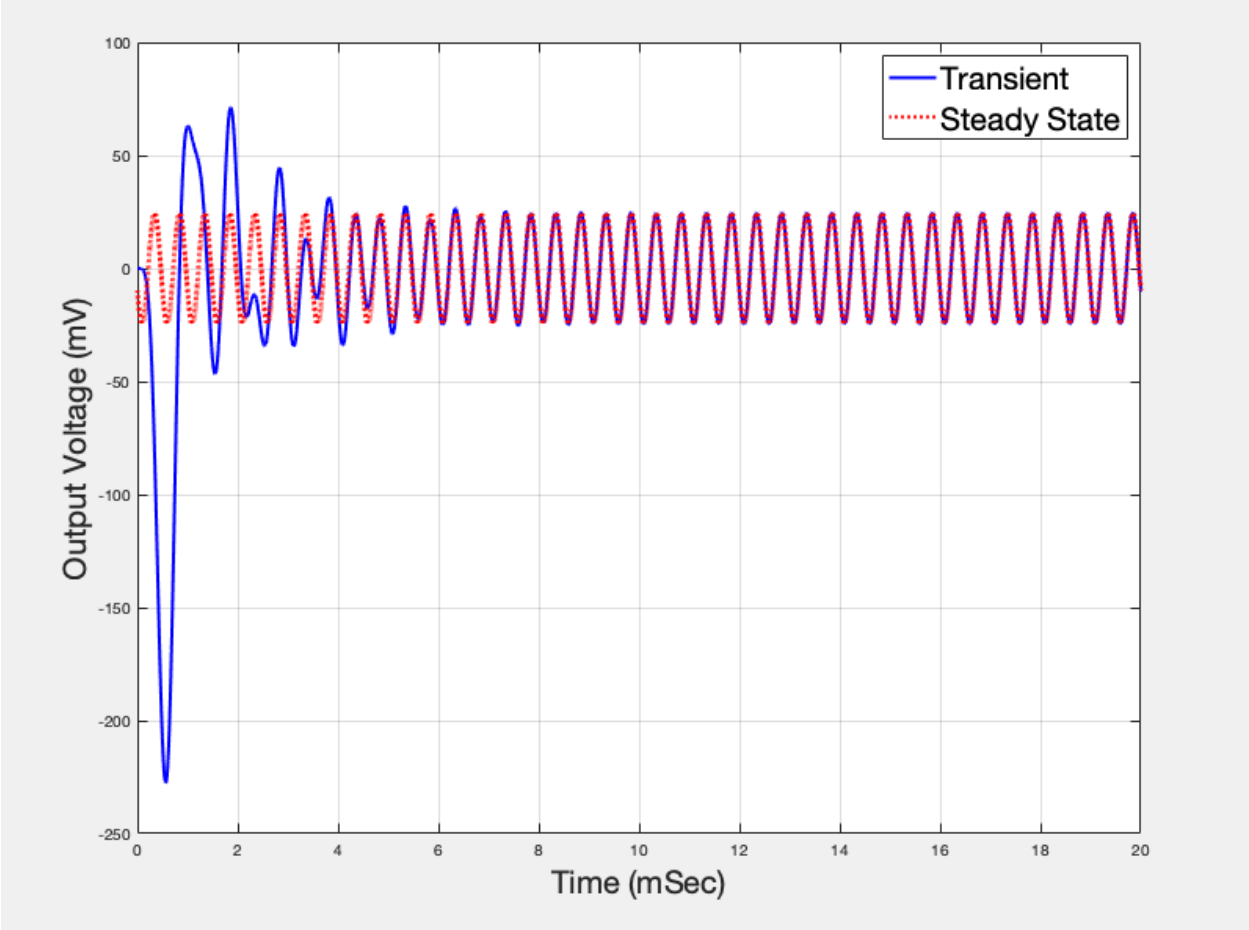
1. Backward Euler

1. *Test your function by running the provided Testbench_Question1.m file. This file simulates the netlists Circuit_chebychev_filter_TD.m (provided in the assignment) in the time domain (transient) and Circuit_chebychev_filter (also provided – this is the same circuit you used before to test your fsolve) in the frequency domain using your fsolve.m function which you developed in past assignments. In your submission, include the code for the new function transient_beuler.m as well as the output plot of the testbench function.*

The code for the new function transient_beuler.m is presented below:

```
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;

%r = [];
%r = zeros (size(sz)
sz = length(tpoints);
r = zeros (size(sz));

% assume initials are 0
x_n = zeros(length(G));


% from 1 till n-1
for i = 1:(sz-1)
    % slide E00 24/54
    % xn+1 = inv (G+C/h) * (bn+1 + (C/h) * xn)
    x_n = (G + C / h) \ (BTime(tpoints(i)) + (C / h) * x_n);

    r(i+1) = x_n(out);
end

end
```

The output plot of the testbench 1 is presented below:



2. *Explain the relation between the two plots in the output figure.*

In the plot, there are 2 plots displayed: one for transient analysis and one for the steady state response. Steady state plot displays the result the transient analysis should reach to should the system analysed is a stable system. As expected, the transient analysis plot starts with a higher amplitude than the steady state plot and it oscillates for around 6.5ms until it finally aligns completely with the steady state plot.

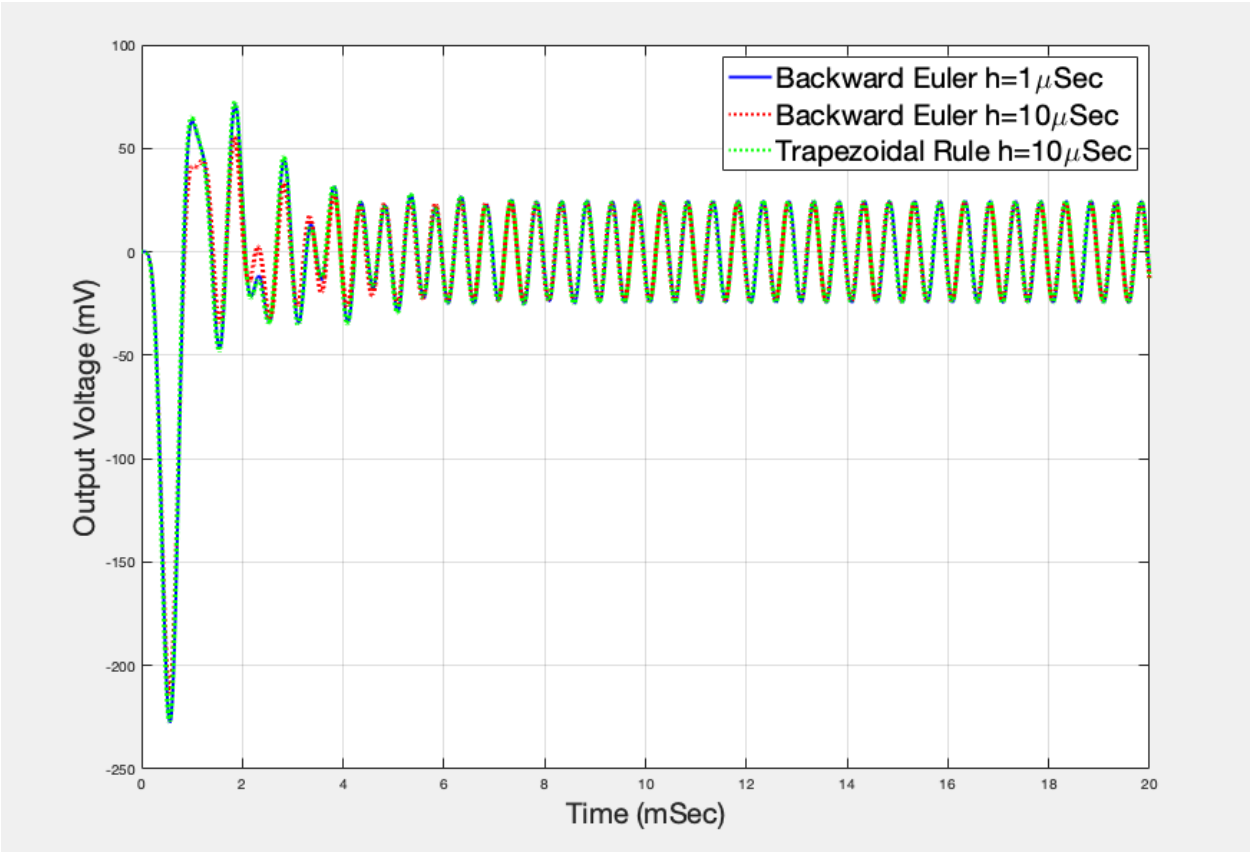
2. Trapezoidal Rule

1. *Test your function by running the provided Testbench_Question2.m file. This script also runs your code from Question 1 and compares BE to TR. In your submission, provide the code for the function you wrote as well as the plot from the testbench.*

The code for the new function transient_trapez.m is presented below:

```
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

tpoints = t1:h:t2;

r = [];

sz = length (tpoints);
%for speed
r = zeros (size(sz));

x_n = zeros(length(G), sz);


% from 1 till n-1
for i = 1:(sz - 1)
    % slide E00 28/54
    % xn+1 = inv ((G + 2 * C / h)) * (bn + bn+1 + (2C/h -G) * xn)
    x_n = (G + 2 * C / h) \ (BTime(tpoints(i)) + BTime(tpoints(i+1)) + (2 * (C / h) - G) * x_n);


    r(i+1) = x_n(out);
end

end
```

The output plot of the testbench 2 is presented below:



2. By examining the plot, what can you deduce about the BE and TR methods?

The plot provides the backward euler at 1us (blue), backward euler at 10us (red) and trapezoidal rule at 10us (green) solutions. All results reach steady state and as expected, backward euler at 10us reaches the steady state faster than backward euler at 1us. Less expectedly, it is observed that the plots of backward euler at 1us and trapezoidal rule at 10us align very closely. This means the trapezoidal rule technique is more efficient with the same step size h than backward euler (when compared to red). Since it reaches the stability at the same time as backward euler with a larger step size (when compared to blue), it should have a higher stability condition than backward euler.

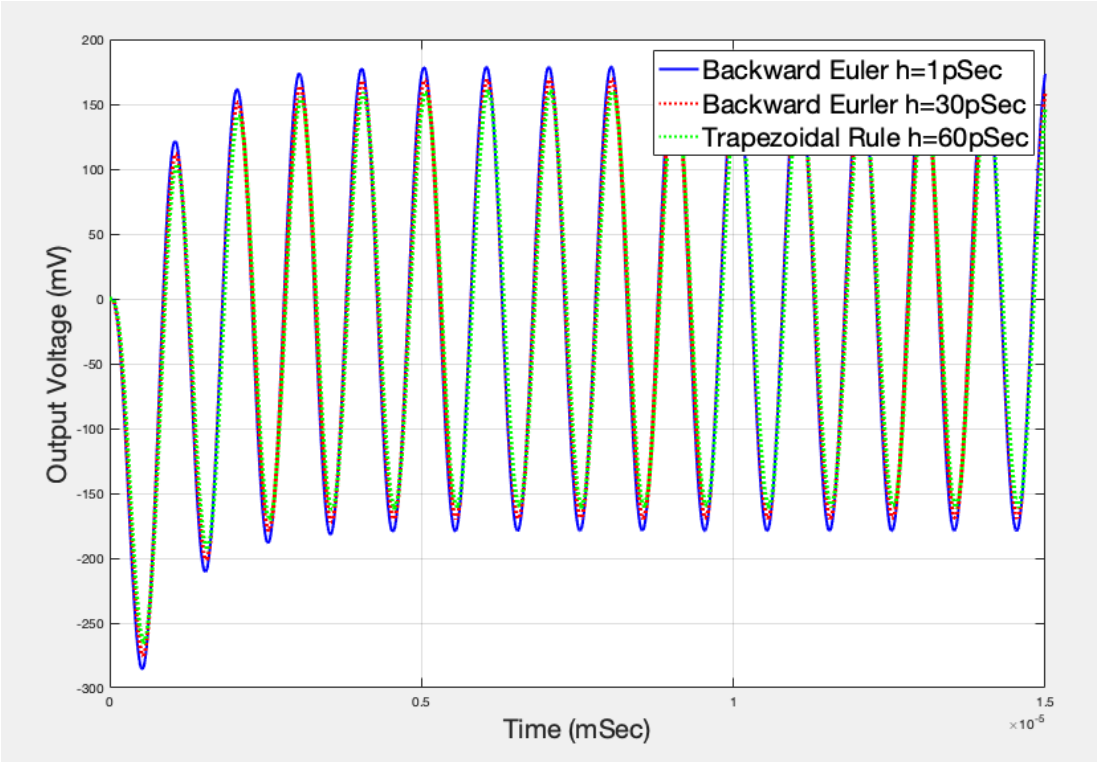
3. Forward Euler

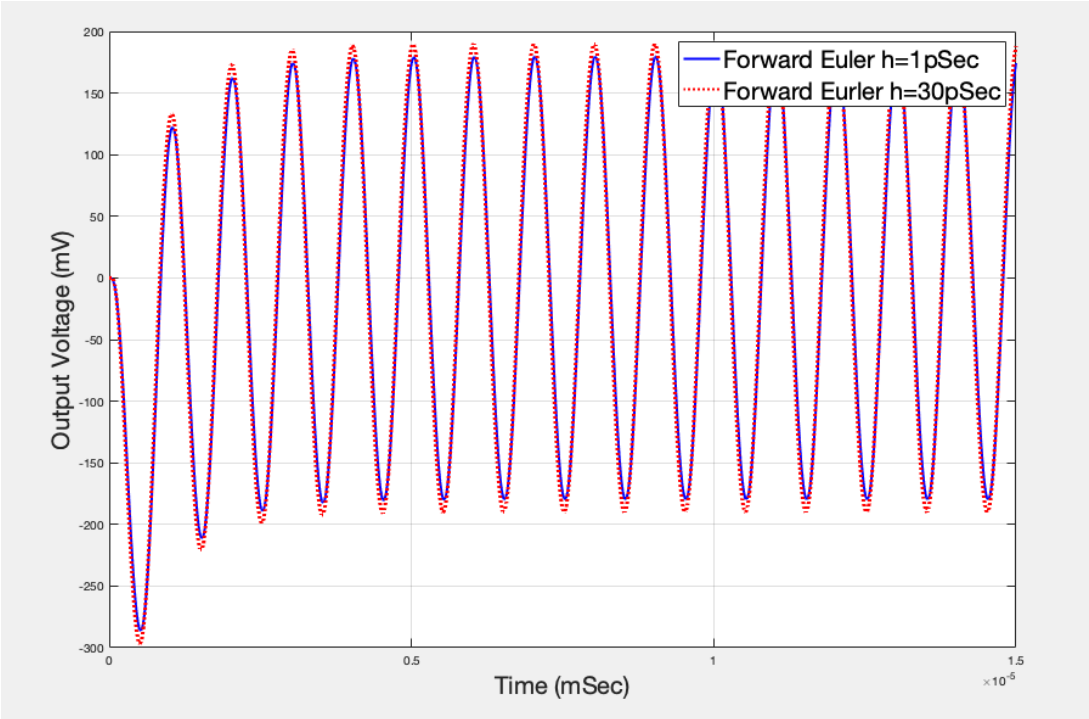
1. *In order to test your code, run the provided test bench script Testbench_Question3.m which simulates the circuit in the provided netlist Q3BEcircuit.m.*

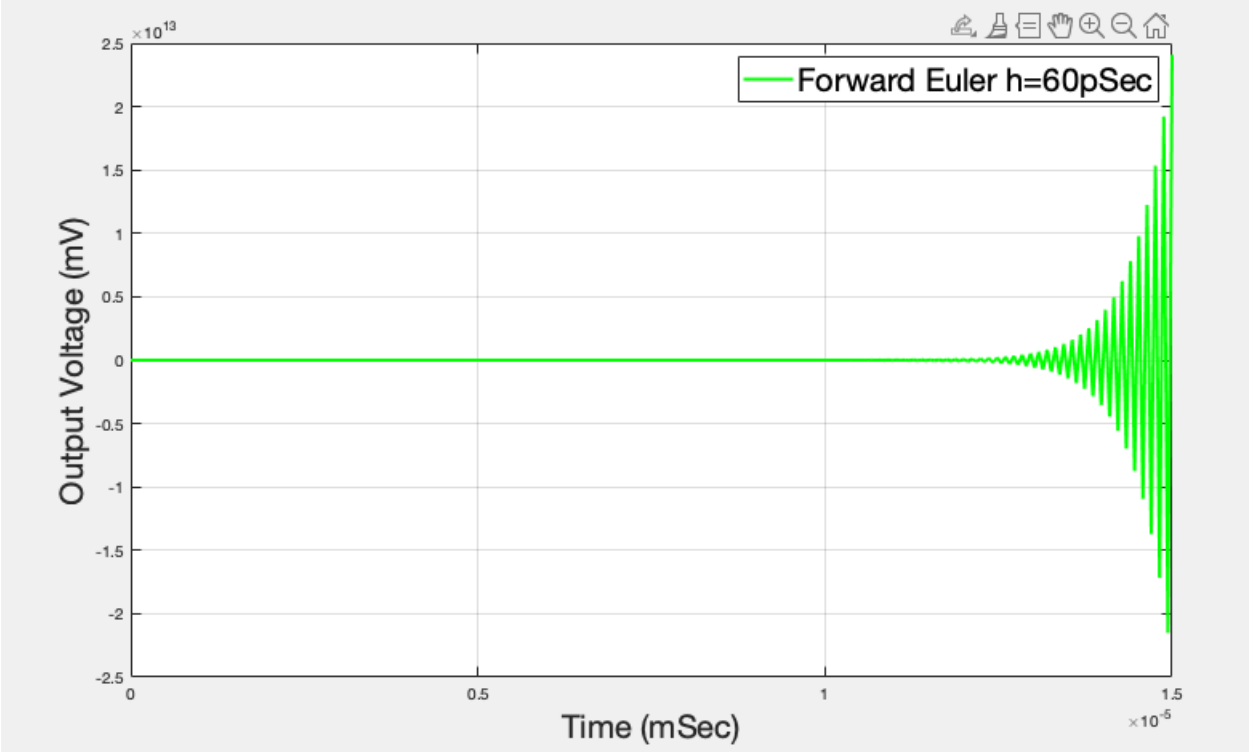
The code for the new function transient_feuler.m is presented below:

```
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

r = [];

eigenvalue = eig (-C \ G)

tpoints = t1:h:t2;

sz = length(tpoints);
%for speed
r = zeros (size(sz));
% assume initials are 0
x_n = zeros(length(G));

% from 1 till n-1
for i = 1:(sz - 1)
    % slide E00 24/54
    % xn+1 = inv (C/h) * (bn - (G-C/h) * xn)
    x_n = (C / h) \ (BTime(tpoints(i)) - (G - C / h) * x_n);

    %xn+1
    r(i+1) = x_n(out);
end

end
```

The output plots of the testbench 3 are presented below:


Backward Euler h=1pSec
Backward Eurler h=30pSec
Trapezoidal Rule h=60pSec
Output Voltage (mV)
Time (mSec)
200
150
100
50
0
-50
-100
-150
-200
-250
-300
0
0.5
1
1.5
×10^-5


Forward Euler h=1pSec
Forward Eurler h=30pSec
Output Voltage (mV)
Time (mSec)
200
150
100
50
0
-50
-100
-150
-200
-250
-300
0
0.5
1
1.5
×10^-5



2. *Examine and comment on what you learn from the output files.*

From examining the plots, one can see that at h=60ps, trapezoidal rule reaches stability in figure 1 while the forward euler never stabilizes. This means that the stability condition for forward euler method is less than h=60ps and the trapezoidal rule should have a higher stability condition than forward euler. Furthermore, when figure 1 and 2 is examined side by side, it can be seen that forward euler and backward euler methods both stabilize around the same times when their step sizes are the same.

3. *It can be shown that, when the C matrix is invertible, the poles of the circuit are the eigenvalues of the matrix –inv(C)G (note the negative sign). Determine the stability condition of the forward euler method for this circuit and experimentally verify your results by running simulations (note the eig function in matlab computes the eigenvalues of a matrix).*

The eigenvalues of the matrix -inv(C)*G were calculated using the eig function in MATLAB as eig (-C\G). The returned eigenvalues were

-0.1206e10
-1.0000e10
-2.3473e10
-3.5321e10

For each of these poles, the stability condition of abs(h*lambda+1)<0 was solved.

- **-1 < 1-h*0.1206e10 <1**

  0 > h*0.1206e10 > 2

  0 > h > 1.6584e-9

- **-1 < 1-h*1.0000e10 <1**

  0 > h*1.0000e10 > 2

  0 > h > 2.0000e-9

- **-1 < 1-h*2.3473e10 < 1**

  0 > h*2.3473e10 > 2

  0 > h > 8.5204e-11

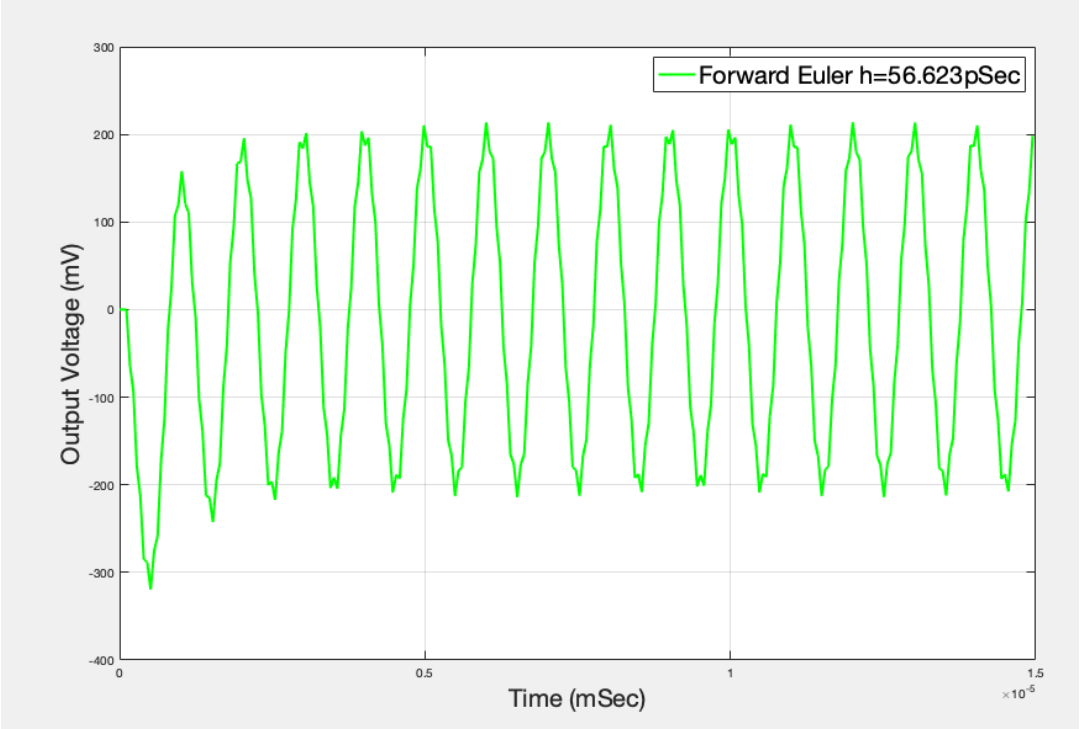
- **-1 < 1-h*3.5321e10 < 1**

  0 > h*3.5321e10 > 2

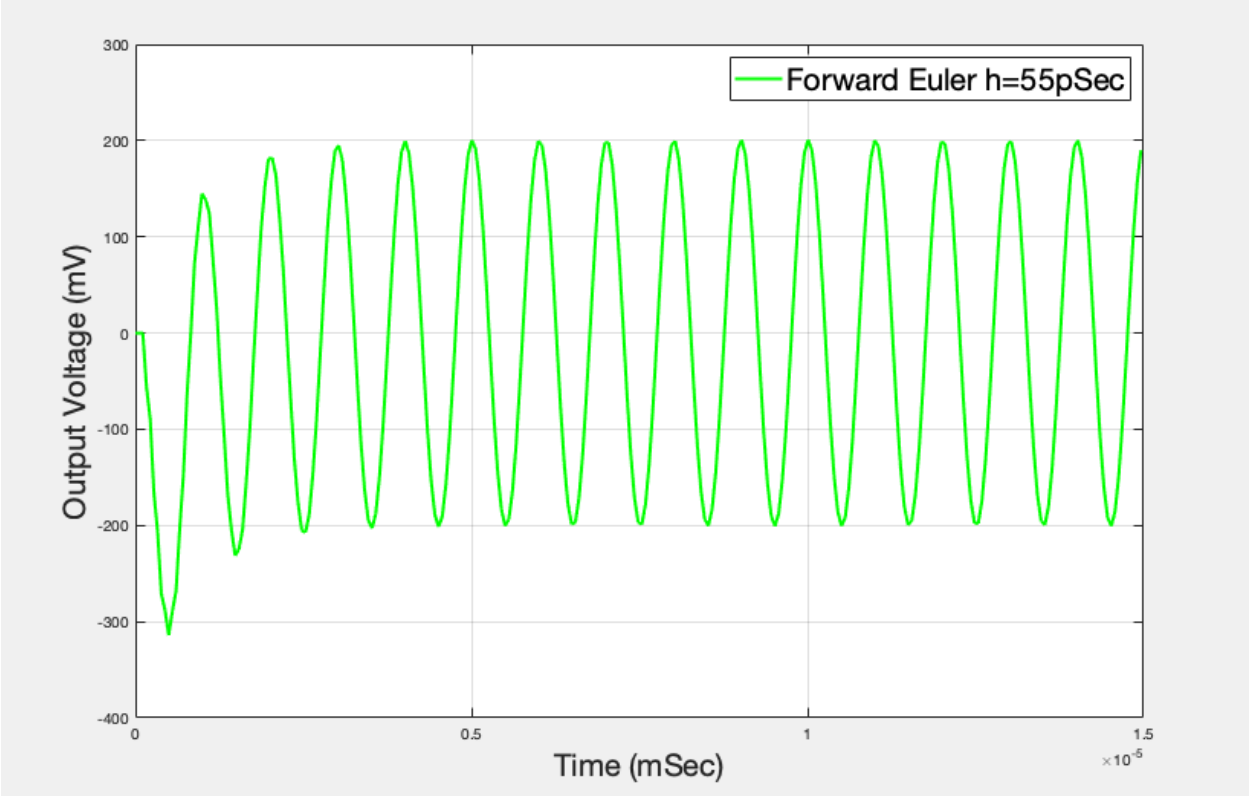
  0 > h > 5.6623e-11

Therefore, the stability condition is 0 > h > 5.6623e-11 for the system. (h is already expected to be a positive value.) To test this, h was set at, above and below 5.6623e-11 to observe the behaviour of the plot.

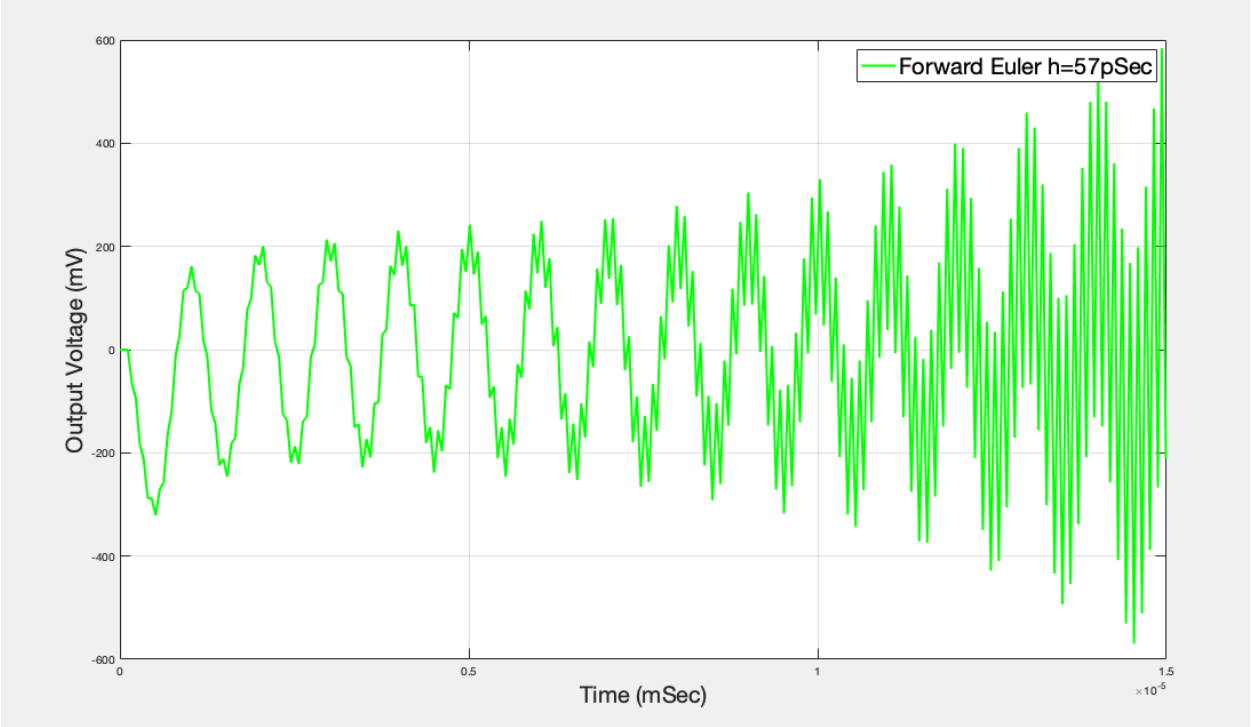
At h = 56.623e-12 s:



Slightly below calculated stability condition at h = 55.000e-12s:



Slightly above calculated stability condition at h= 57.000e-12s:



As it can be observed from the plots, when at h=57ps, the system doesn't stabilize to reach the steady state and it is stable when h is set at 56.62ps or below. This confirms the calculated value of h.

4. Nonlinear Circuits Transient

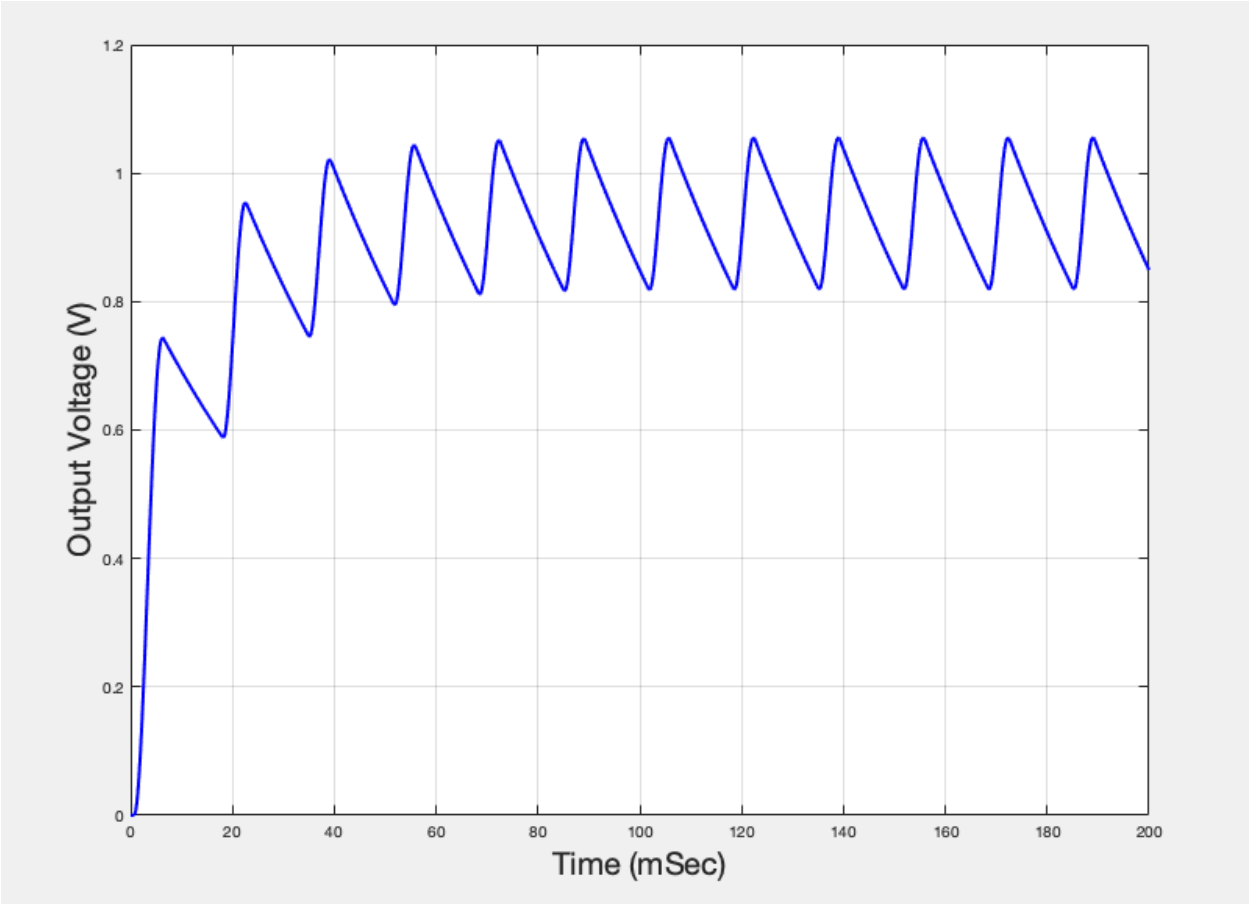
1. *Test your circuit by running the provided script Testbench_Question4 which simulates the rectifier circuit in netlist Circuit_Rectifier.m also provided.*

The code for the new function nl_transient_beuler.m is presented below:

```
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;
```

```
    maxerr = 1e-6;
    r = [];

    % assume initials are 0
    sz = length (b);
    x_0 = zeros(sz);
    x_n = zeros(sz);



    iteration = 0;
for i = tpoints
    error_tolerable = false;

    iteration = iteration + 1;
    while(~error_tolerable)
        % f vector
        f = f_vector(x_n);
        % Jacobian
        J = nlJacobian(x_n);

        % Slides G00 33/45
        % 0 = fn+1 + (G + C / h) * xn+1 - bn+1 - (C / h) * xn
        % xn = x_0
        % xn+1 = x_n
        phi = f + (G + C / h) * x_n - BTime(i + h) - (C / h) * x_0 ;
        % delta phi
        d_p = G + (C / h) + J;

        %delta x = - inv(delta phi) * phi
        d_x = - d_p \ phi;

        % new point
        x_n = x_n + d_x;

        % error
        err = norm(d_x);

        if (abs(err) <= maxerr)
            error_tolerable = true;
        end

    end

    %r = [r;x_n(out)];
    r(iteration) = x_n(out);
    x_0 = x_n;
end
```

The output plot of the testbench 4 is presented below:


Output Voltage (V)
Time (mSec)
0
0.2
0.4
0.6
0.8
1
1.2
0
20
40
60
80
100
120
140
160
180
200