EEET1467 CIRCUIT&SYSTEMS SIMULATION

Min-Project

NUMERICAL SOLUTION OF *ODEs*

1. Introduction:

In this project, a differential equations will be solved with PSPICE and Matlab program. The differential equations is a RLC circuit description equations as following:

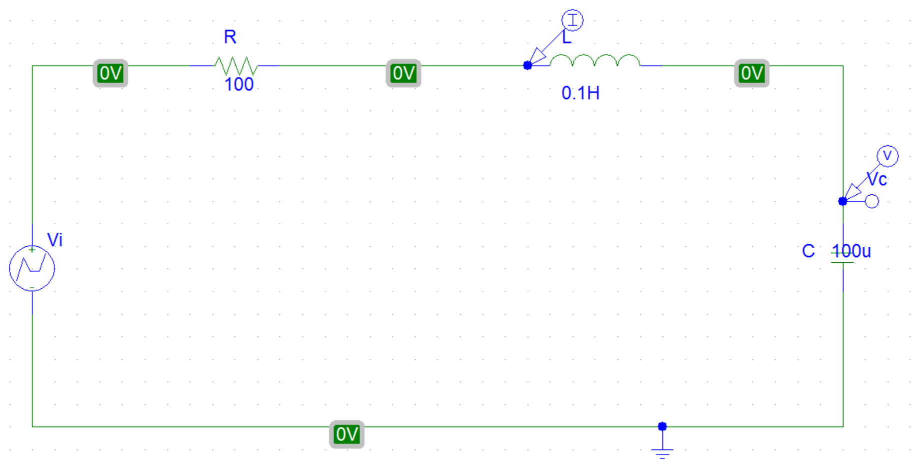
$$\frac{di_L}{dt} = \frac{1}{L}(V_i - i_L R - v_C)$$
$$\frac{dv_C}{dt} = \frac{1}{C}(i_L)$$

where i_L and v_C are the unknowns to be determined and V_i is a step voltage.

Two numerical methods will be implemented with Matlab program; they are the explicit Euler's Method and the trapezoidal rule (central finite-difference). At the first simulation, we will use a small time step to compare Matlab simulation the PSPICE outputs. Then we change to a large time step making the second simulation to compare the stability of two numerical methods.

2. PSPICE Simulation:

Considering the following circuit:

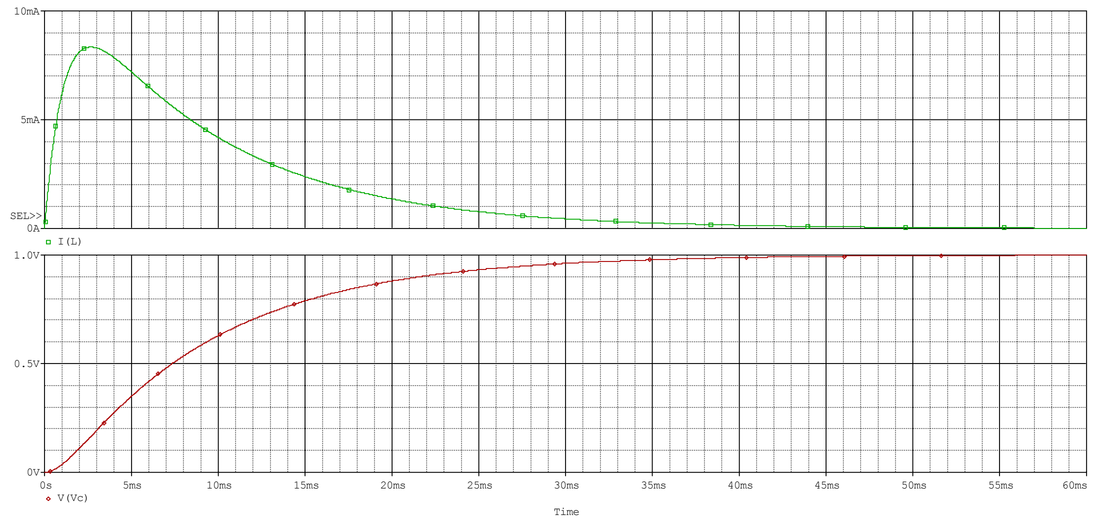


In this circuit, have:

$i_C = i_L = C\frac{dV_C}{dt}$, $V_L = L\frac{di_L}{dt}$, apply KVL, have: $V_i = i_L R + L\frac{di_L}{dt} + V_C$, $i_L = C\frac{dV_C}{dt}$, which is equivalent to the target differential equations. So this circuit could be used to simulate target equations.

Before we do the simulation with PSPICE, firstly, we determine the components value as: R=100Ω,L=0.1H,C=100μF. The voltage source Vi is a step voltage source whose peak value is 1V. So the initial value of VC and IL could be set as 0V and 0A.

From above circuit and component values, using transient analysis, we could get the following PSPICE simulation results:



Above plot is the output of IL, below is the output of VC. It shows that the maximum value of IL is about 8.4mA which occurs at about 2.4ms. VC 70percent rising edge is about 12ms. This result will be compared with Matlab program output.

3. The Simulation with Euler's Method:

In this section, we will solve the target equations with Euler's Method. In Euler's Method, for differential equation, $\frac{dx}{da} = f(x,t)$, if we know the x value in t_{n-1} as x_{n-1}, then the x's approximation value in t_n could be calculated with:

$$x_n = f(x,t) \times \Delta T + x_{n-1}.$$

If the time step ΔT is small enough, the series of $\{x_0, x_1, \cdots x_n\}$ could be a good approximation solution for the target differential equation.

In this case, the target differential equations' Euler's Method computation equation could be:

$$IL_n = \frac{\Delta T}{L}(V_i - IL_{n-1}R - VC_{n-1}) + IL_{n-1}$$

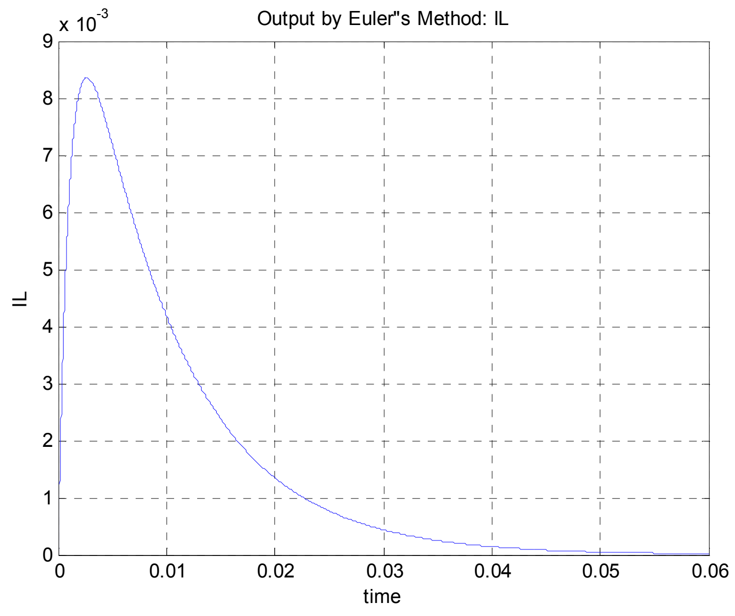
$$VC_n = \frac{\Delta T}{C} IL_{n-1} + VC_{n-1}$$

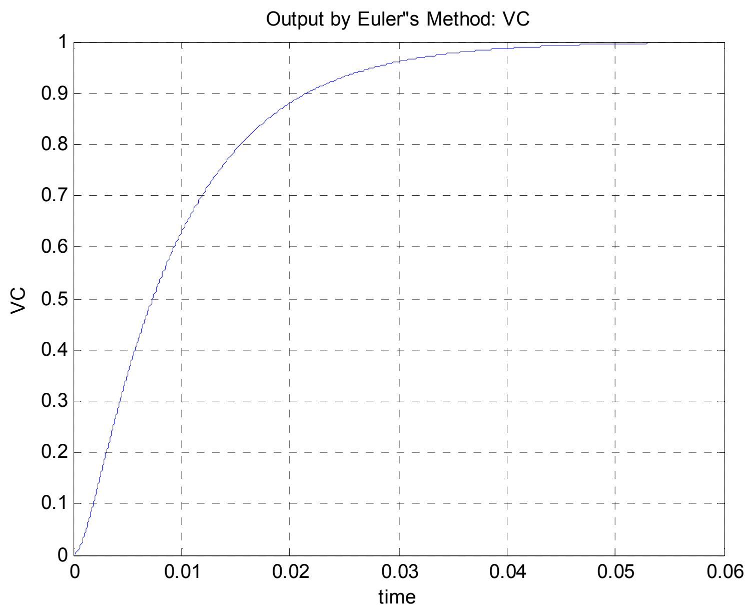
Based on above equations and considering our component initial values, which are: R=100Ω, L=0.1H, C=100μF, Vi=1V. Setting time step set as 0.01 ms and simulation time length from 0ms to 60ms, we could implement Euler's Method with following Matlab program:

```
% Initialization
Tinit=0; Tend=6*10^(-2); Tstep=10^(-5);    % simulating 0-60ms, time step is 0.01ms
Vi=1; ILinit=0.0; VCinit=-0.0;                 % L current and C Voltage initial value is zero
R=100; L=0.1;    C=10^(-4);           % the value of R is 100Ohm, L is 0.1F, C is 100u
T=Tinit:Tstep:Tend;                 % Initializing time vector
IL = zeros([1,length(T)]);      % Initializing IL
VC = zeros([1,length(T)]);      % VC initializing
IL(1)=ILinit; VC(1)=VCinit;    % Set IL,VC initial vlaue

%Euler's Method
for j=2:length(T)                    % Euler's Method Calculation
        IL(j)=IL(j-1)+Tstep*(Vi-IL(j-1)*R-VC(j-1))/L;
        VC(j)=VC(j-1)+Tstep*IL(j-1)/C;
end
figure;
plot(T,IL); YLabel('IL'); Xlabel('time');      % IL plotting
Title('Output by Euler''s Method: IL');
figure;
plot(T,VC); YLabel('VC');Xlabel('time');       % VC plotting
Title('Output by Euler''s Method: VC');
```

The output from Euler's Method is shown in following:





Comparing above Matlab output with the simulation results from PSPICE, we could find they are very close. Both of them have maximum IL as about 8.4mA at about 2.5ms. Both 70Percent VC maximum value rising time is about 12 ms.

4. The Simulation with Trapezoidal rule :

In trapezoidal rule method, instead of calculating the gradient at t_{n-1}, it is calculating the gradient at $(t_{n-1}+t_n)/2$. The equation become to:

$$x_n = \Delta T\left\{ f\left(\frac{x_n + x_{n-1}}{2}, \frac{t_n + t_{n-1}}{2}\right) \right\} + x_{n-1}.$$

Applying it to target equations in this problem, the target differential equations become to:

$$\begin{cases} (IL_1 - IL_0)/\Delta T = \dfrac{1}{L}\left(V_i - \dfrac{IL_0 + IL_1}{2}\cdot R - \dfrac{VC_0 + VC_1}{2}\right) \\ (VC_1 - VC_0)/\Delta T = \dfrac{IL_1 + IL_0}{2C} \end{cases}$$

Above equations could finally transform to:

$$\begin{cases} \dfrac{\Delta T}{2L}VC_1 + (1+\dfrac{R\Delta T}{2L})IL_1 = \dfrac{\Delta T}{L}V_i + (1-\dfrac{R\Delta T}{2L})IL_0 - \dfrac{\Delta T}{2L}VC_0 \\ VC_1 - \dfrac{\Delta T}{2C}IL_1 = VC_0 + \dfrac{\Delta T}{2C}IL_0 \end{cases}$$

Above linear equations has the solution as: $\begin{cases} VC_1 = \dfrac{\Delta VC}{\Delta} \\ IL_1 = \dfrac{\Delta IL}{\Delta} \end{cases}$, where,

$$\Delta = \begin{vmatrix} \dfrac{\Delta T}{2L} & 1+\dfrac{R\Delta T}{2L} \\ 1 & -\dfrac{\Delta T}{2C} \end{vmatrix} = -1 - \frac{\Delta T^2}{4LC} - \frac{R\cdot\Delta T}{4L}$$

$$\Delta VC = \begin{vmatrix} \dfrac{\Delta T}{L}V_i + (1-\dfrac{R\Delta T}{2L})IL_0 - \dfrac{\Delta T}{2L}VC_0 & 1+\dfrac{R\Delta T}{2L} \\ VC_0 + \dfrac{\Delta T}{2C}IL_0 & -\dfrac{\Delta T}{2C} \end{vmatrix}$$

$$= -\frac{\Delta T}{2C}\left(\frac{\Delta T}{L}V_i + (1-\frac{R\Delta T}{2L})IL_0 - \frac{\Delta T}{2L}VC_0\right) - \left(VC_0 + \frac{\Delta T}{2C}IL_0\right)\left(1+\frac{R\Delta T}{2L}\right)$$

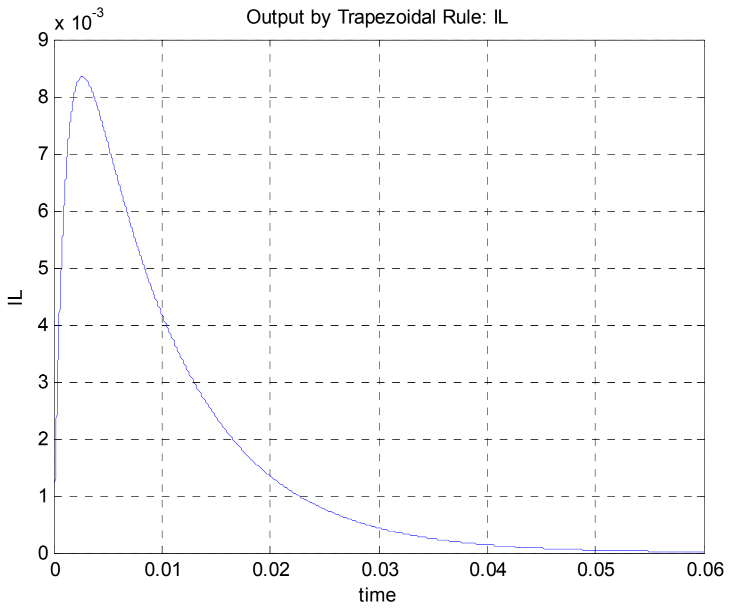
$$\Delta IL = \begin{vmatrix} \dfrac{\Delta T}{2L} & \dfrac{\Delta T}{L}V_i + (1-\dfrac{R\Delta T}{2L})IL_0 - \dfrac{\Delta T}{2L}VC_0 \\ 1 & VC_0 + \dfrac{\Delta T}{2C}IL_0 \end{vmatrix}$$

$$= \frac{\Delta T}{2L}\left(VC_0 + \frac{\Delta T}{2C}IL_0\right) + \frac{\Delta T}{2L}VC_0 - \frac{\Delta T}{L}V_i - \left(1 - \frac{R\Delta T}{2L}\right)IL_0$$

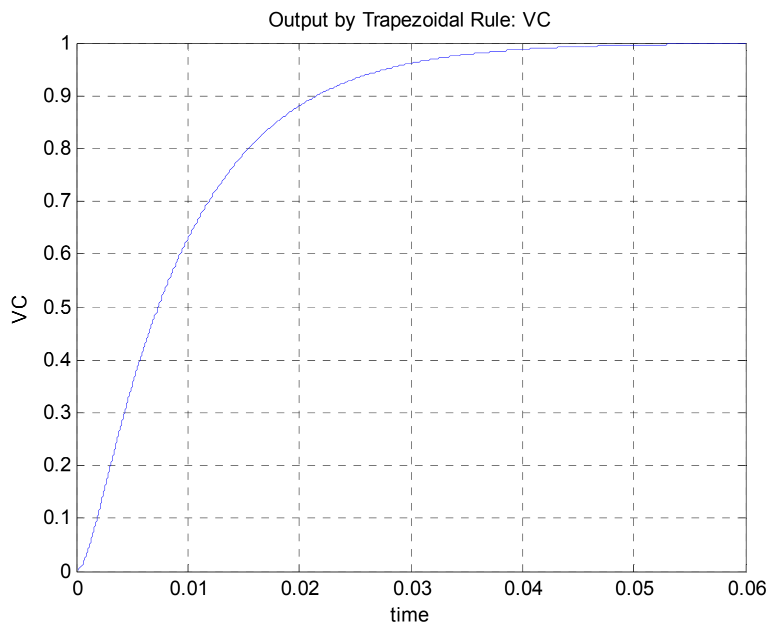
Bases on above equations, we could use following Matlab codes to simulate the target equations, where the initialization and time step are same as in Euler's Method which have been omitted:

```
%Trapezoidal's Rule
Delta=-Tstep^2/4/L/C-1-R*Tstep/2/L;      % Delta
IL(1)=ILinit; VC(1)=VCinit;    % Set IL,VC initial vlaue
for j=2:length(T)                     %Trapezoidal's Rule Calculation
    DeltaVC=-Tstep/2/C*(Tstep/L*Vi+(1-R*Tstep/2/L)*IL(j-1)-Tstep/2/L*VC(j-1))-(VC(j-1)+
    … Tstep*IL(j-1)/2/C)*(1+R*Tstep/2/L);
    DeltaIL=Tstep/2/L*(VC(j-1)+Tstep*IL(j-1)/2/C)+Tstep/2/L*VC(j-1)-Tstep/L*Vi-
    … (1-R*Tstep/2/L)*IL(j-1);
    IL(j)=DeltaIL/Delta;
    VC(j)=DeltaVC/Delta;
end
figure;
plot(T,IL); YLabel('IL'); Xlabel('time');    %IL plotting
Title('Output by Trapezoidal Rule: IL'); grid on;
figure;
plot(T,VC); YLabel('VC');Xlabel('time');      %VC plotting
Title('Output by Trapezoidal Rule: VC'); grid on;
```

The output from this simulation could be shown in following:



and:

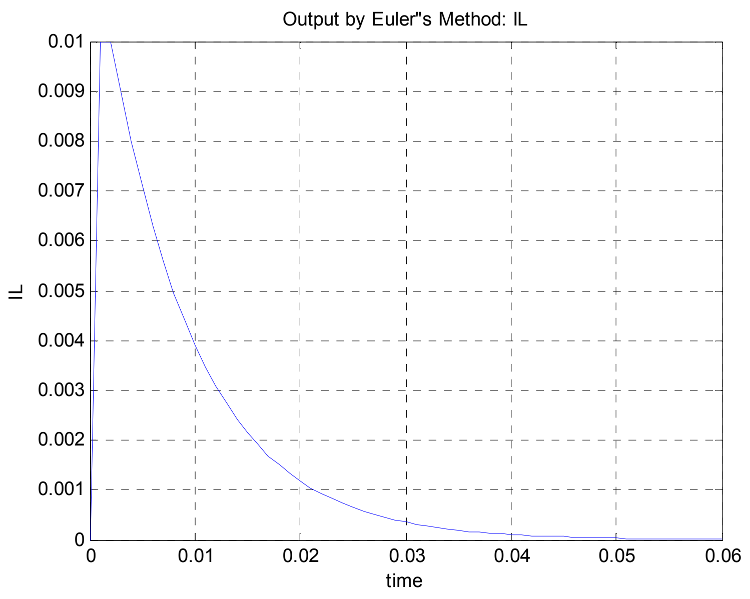


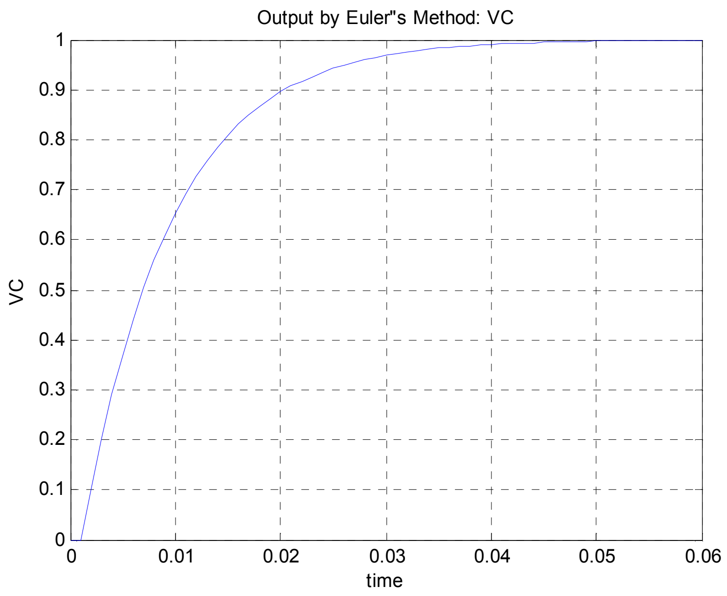
It could see that output from trapezoidal rule method has the same results as PSPICE simulation and Euler's Method. That is: for IL, they all have a maximum value as about 8.4mA at about 2.5ms; for VC, the 70Percent VC maximum value rising time is about 12 ms.

5. Stability study:

From above Matlab simulation results, we could get that when the time step is small enough, f.g. in this case which is 0.01ms, the simulation results from Euler's Method and trapezoidal rule don't have the difference. In this section, we will set the time step as 1ms to see is there any difference between them.

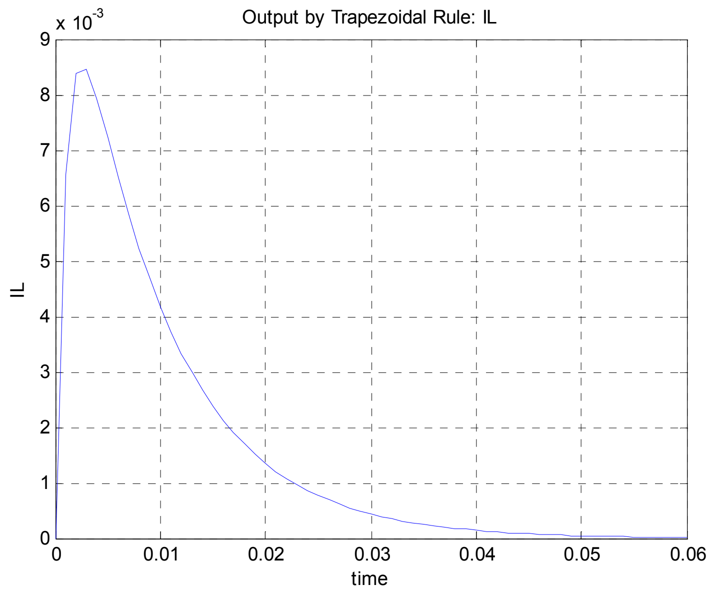
The output from Euler's method in this time step is following:

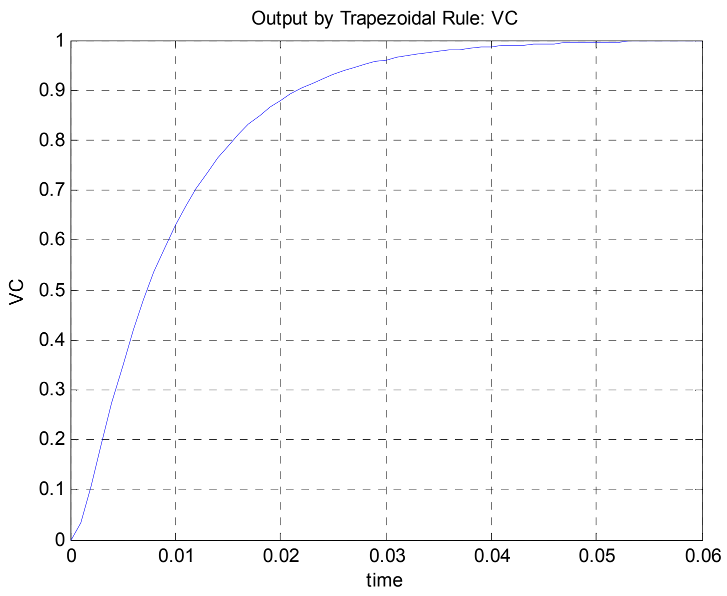




From above output, we could see that the maximum value of the IL in this simulation has become to above 10ms in 2ms. In VC plot there appears a death area at the first about 1.5 ms where the value of VC is zero volts.

The output from trapezoidal rule as following:





From above trapezoidal rule output we could find the output lines are not as smooth as in the small time step case. But the maximum value of the IC is still could be estimated as 8.5mA at about 2.5ms and the death area in VC plot disappears. So the outputs from trapezoidal rule are much closer to PSPICE output in this case. This means the trapezoidal rule method has a better stability than Euler's Method.

6. Conclusion:

In this project, we studied a differential equations. The solution for this differential equations is with the PSPICE circuit simulation and Matlab program. The Matlab program is coded with Euler's Method and trapezoidal rule method.

In the first round simulations, we set a small time step as 0.01ms in the Matlab program. In this setting, all the simulation results are very close. This means both the PSPICE simulation and numerical methods could be used to solve differential equations.

Then we change the time step to 1ms. In this simulation, both numerical methods' output plots are not as smooth as in first simulation. But the plot characteristic values from trapezoidal rule method don't show a notable difference from the first simulation while the outputs from Ruler's Method have some notable difference. This shows that trapezoidal rule method has a better stability.

7. Appendix:

Matlab simulation code:

```
%    Differential Equations Simulation with Euler's Method and Trapezoidal Rule Method
%    Author: Wei Han;    Date: 16/10/2009

clear; clc; % Clear Memory
% Initialization
Tinit=0; Tend=6*10^(-2); Tstep=10^(-3);    % simulating 0-60ms, time step is 0.01ms
Vi=1; ILinit=0.0; VCinit=-0.0;                  % L current and C Voltage initial value is zero
R=100; L=0.1;    C=10^(-4);            % the value of R is 100Ohm, L is 0.1F, C is 100u
T=Tinit:Tstep:Tend;                   % Initializing time vector
IL = zeros([1,length(T)]);      % Initializing IL
VC = zeros([1,length(T)]);      % VC initializing
IL(1)=ILinit; VC(1)=VCinit;    % Set IL,VC initial vlaue

%Euler's Method
for j=2:length(T)                     % Euler's Method Calculation
      IL(j)=IL(j-1)+Tstep*(Vi-IL(j-1)*R-VC(j-1))/L;
      VC(j)=VC(j-1)+Tstep*IL(j-1)/C;
end
figure;    plot(T,IL); YLabel('IL'); Xlabel('time');      % IL plotting
Title('Output by Euler''s Method: IL'); grid on;
figure; plot(T,VC); YLabel('VC');Xlabel('time');       % VC plotting
Title('Output by Euler''s Method: VC'); grid on;

%Trapezoidal's Rule
Delta=-Tstep^2/4/L/C-1-R*Tstep/2/L;      % Delta
IL(1)=ILinit; VC(1)=VCinit;    % Set IL,VC initial vlaue
for j=2:length(T)                          %Trapezoidal's Rule Calculation
      DeltaVC=-Tstep/2/C*(Tstep/L*Vi+(1-R*Tstep/2/L)*IL(j-1)-Tstep/2/L*VC(j-1))-(VC(j-1)+Tstep*IL(j-1)/2/C
      )*(1+R*Tstep/2/L);
      DeltaIL=Tstep/2/L*(VC(j-1)+Tstep*IL(j-1)/2/C)+Tstep/2/L*VC(j-1)-Tstep/L*Vi-(1-R*Tstep/2/L)*IL(j-1);
      IL(j)=DeltaIL/Delta;
      VC(j)=DeltaVC/Delta;
end
figure; plot(T,IL); YLabel('IL'); Xlabel('time');    %IL plotting
Title('Output by Trapezoidal Rule: IL'); grid on;
figure;plot(T,VC); YLabel('VC');Xlabel('time');      %VC plotting
Title('Output by Trapezoidal Rule: VC'); grid on;
```