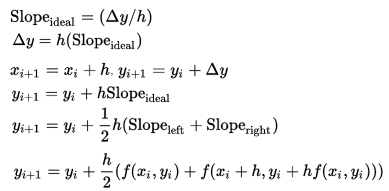
Euler’s method is used to calculated the slope at a point x and then estimate the value of next nearby point by the formula Y(n+1)=Y+f(x0,y0)\*h, Where f(x0,y0) is the slope obtained at point (x0,y0) and h is the step interval(x1-x2). It is easy to see that Euler’s method is a linear approximation. The smller value of h, the smaller the error and better the approximation. However, even tiny error can acculmulates to huge global error. For example, when a curve convaves up, its tangent line will underestimate the vertical coordinate of the next point and vice versa for a concave down solution. There is no way to know wether the curve concaves up or down. So we apply Heun’s method, which is second-order numerical method.

Heun ‘s method, normally recogonised as improved Euler’s method. It can be combined with the Euler method estimate of the next point with step size h to give the slope of the tangent line at the right end-point. Next the average of two slopes is used to find the corrected corrdinates of the right end interval.



The Heun.m script has seven input and two output parameters, where f is a function represents the input waveform defined by the question. i0 is the current value at t0, t0 is the starting time which is normally zero, tf is the final time, N is the number of intervals, R and L are resistance and inductance respectively.

We kenow that current of a inductor does not change instantenously. Thus in this RL circuit problem, we use the slope of inductor current as a predictor to calculate out an array of inductor currents IL(t) by N iterations. The slope of inductor current (vin-R\*i\_in)/L. Set this formula as a function call in the script, then evaluate this function at a variety values of vin and i\_in as N increments. In each iteration, we apply Heun’s method to an array of inductor current IL(t). Vout is obtained by Vout=Vin-R\*IL(t). Simultaneously, an array of time with size N and step size h is recorded.

Finally we have two arrays as outputs. MATLAB command plot(T,Vout)gives the numerical solution of Vout.

MATLAB code for Heun’s

function [ t,vout ] = Heun(f,i0,t0,tf,N,R,L)

%takes as input f, the function describing vin(t)

increment = @(vin,i\_in)(vin-R\*i\_in)/L;

%di/dt as a function of vin and i

h=((tf-t0)/N); %get the step size

t=t0:h:tf; %generate an array of each of the times to plot

vout=zeros(size(t)); %generate an output array for Vout

i\_L = zeros(size(t)); %array of inductor currents at t

vout(1) = f(t0) - R\*i0; %starting values for Vout

i\_L(1) = i0; %starting values for current i

for n=1:N % loop for N steps

%define temporary variables to limit array accessing

V=feval(f, t(n)) %evaluate value of input signal at t

Vp=feval (f, t(n+1)); %evaluate value of input signal at t+h

i\_t = i\_L(n); %get the current value of i

i\_pred=i\_t + h\*increment(V,i\_t); % predict for next value of i

grad1=increment(V,i\_t); % gradient at t

grad2=increment(Vp,i\_pred); % gradient at t+h

grad\_ave=0.5\*(grad1+grad2); % average gradient

i\_L(n+1)=i\_t+h\*grad\_ave; % predict i at t+h again, store it in an array

vout(n+1) = Vp - R\*i\_L(n+1); %get the output voltage

end

end

Error analysis

With input a cosine wave of period T=150µs and amplitude Vin=6V, we solve the differnential equation using MATLAB command:

syms y(t)

L=1.5e-3;

R=0.5;

f=1/(150e-6);

K=6\*cos(2\*pi\*f\*t)

dsolve(L\*diff(y) + R\*y == K, y(0) == 1)

Exact solution:

exact\_i = @(t) (12\*pi\*T\*L\*sin(2\*pi\*t/T)+6\*T^2\*R\*cos(2\*pi\*t/T))/(T^2\*R^2+4\*pi^2\*L^2)+ exact\_c\*exp(-R\*t/L); where exact\_c = -(6\*T^2\*R)/(T^2\*R^2+4\*pi^2\*L^2);

Define max\_ind the number of iterations. Creat an array of maxium errors with size max\_ind. N=10^ind to boost the number of intervals. For each value of N, a corresponding h is calculated and stored in an array. Obtain the numerical solution array Vout by calling the function script. Moreover, exact solution is subtracting from the numerical solution, getting an array of errors. Then we pick up the maximum absolute value of the error array, store it in the max\_error array. Finally, do the log-log plot of step size h and max\_error.

The local error at each step of the midpoint method is of order {\displaystyle O\left(h^{3}\right)}, giving a global error of order {\displaystyle O\left(h^{2}\right)}. Thus, while more computationally intensive than [Euler's method](https://en.wikipedia.org/wiki/Euler%27s_method), the midpoint method's error generally decreases faster as {\displaystyle h\to 0}.

%error analysis

R=0.5;

L=1.5e-3;

%Sine wave 150

t0 = 0; %set start time

T = 150e-6; %set time period

fre=1/T; %frequency

tf = T; %set finish time

N = 100; %set number of intervals

i\_0 = 0; %initial condition of current

Input = @(t) 6\*cos(t\*2\*pi\*fre); %define input signal

exact\_c = -(6\*T^2\*R)/(T^2\*R^2+4\*pi^2\*L^2);

exact\_i = @(t) (12\*pi\*T\*L\*sin(2\*pi\*t/T)+6\*T^2\*R\*cos(2\*pi\*t/T))/(T^2\*R^2+4\*pi^2\*L^2)+ exact\_c\*exp(-R\*t/L);

max\_ind = 5; %define maxindex fo array size

h\_a = zeros(max\_ind,1); %array for step sizes

me\_h = zeros(max\_ind,1); %arrays for max errors

%error function for Heun's

for ind=1:max\_ind

N=10^ind; %boost N to get a large number of intervals

h=((tf-t0)/N); %define step size

h\_a(ind) = h; %creat an array for all values of h

[t,vout] = Heun(Input,i\_0,t0,tf,N,R,L); %get the output voltage array

vin=arrayfun(Input,t); %input voltages

actual\_i = arrayfun(exact\_i,t); %exact current at t

actual\_vout = vin - R\*actual\_i; %exact Vout at t

error = actual\_vout - vout;

[maximum,index] = max(abs(error)); %find the maximum error

me\_h(ind) = maximum;

end

figure;

subplot(3,1,1);

plot(log(h\_a),log(me\_h));

xlabel('log(h)') % x-axis label

ylabel('log(Max Error)') % y-axis label

title('Error analysis Heun');

grad = polyfit(log(h\_a),log(me\_h),1); %calculate the gradient of the line

text(-20,-10,['Gradient = ' num2str(grad(1))]);

%error function for Ralston

for ind=1:max\_ind

N=10^ind; %boost N to get a large number of intervals

h=((tf-t0)/N); %define step size

h\_a(ind) = h; %creat an array for all values of h

[t,vout] = Ralston( Input, R, L, i\_0, tf, N ); %get the output voltage array

vin=arrayfun(Input,t); %input voltages

actual\_i = arrayfun(exact\_i,t); %exact current at t

actual\_vout = vin - R\*actual\_i; %exact Vout at t

error = actual\_vout - vout;

[maximum,index] = max(abs(error)); %find the maximum error

me\_h(ind) = maximum;

end

subplot(3,1,2);

plot(log(h\_a),log(me\_h));

xlabel('log(h)') % x-axis label

ylabel('log(Max Error)') % y-axis label

title('Error analysis Ralston');

grad = polyfit(log(h\_a),log(me\_h),1); %calculate the gradient of the line

text(-20,-10,['Gradient = ' num2str(grad(1))]);

Reference

https://en.wikipedia.org/wiki/Heun's\_method