GEBZE TECHNICAL UNIVERSITY

ELECTRONIC ENGINEERING

MATH214 NUMERICAL ANALYSIS

Alican Bayındır
200102002087

Project - 3

Preparation date
14.12.2020

Upload date
16.10.2020

Abstract:

Following report includes the numerical analysis course project three, in this project we are given an inductor to find voltage, current, power values by using different types of methods we learnt in class so far. As methods, backward difference method is used to find the derivative of voltage values. Additionally, to make numerical integration, composite Simpson's method, composite midpoint formula, composite trapezoidal formula was used. First, data extracted from pr3data.dat file which is given to us in advance within project guide. We have loaded all the datas to MATLAB and tried to process all operations due to project guideline. Obtained data such as derivative of the voltage values and integration of functions on the plots, are ascertained by the methods plotted on the screen to make comments on them freely.

Introduction:

In the first place the circuit given in this project can be seen in the Figure 1 below;

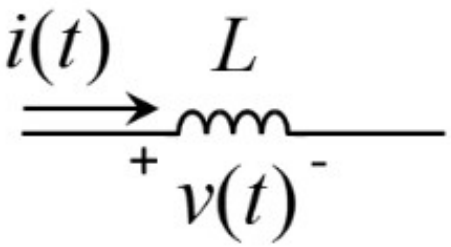

Figure 1: Current and voltage of an inductor.

The methods Composite Simpson's method, Composite midpoint method and Composite Trapezoidal method have been used to solve the problem. All these formulas can be used to converge. Here, Main idea is to divide integration interval $[a, b]$ into subintervals and use simple integration rule for each subinterval. If the current reference is in the direction of the voltage drop across the terminals of the inductor as shown in Fig. 1, the power;

$$p(t)=v(t)xi(t)$$

where v(t) is the voltage in Volts [V] and i(t) is the current in Amperes [A]. The power p(t) is in Watts [W]. Knowing that the power is the time rate of expending energy and assuming a reference for a zero energy corresponds to zero current in the inductor, the stored energy w(t) in Joules [J] can be given as;

$$w(t) = \int_0^t p(\tau)d\tau = \int_0^t v(\tau)i(\tau)d\tau.$$

Using the voltage-current relation of the inductor, the stored energy can also be found as;

$$w(t)=\frac{1}{2}L\,i^{2}(t)$$

Composite Simpson's Method;

If the interval of integration [a, b] is in some sense "small", then Simpson's rule with n = 2 subintervals will provide an adequate approximation to the exact integral. By small, what we really mean is that the function being integrated is relativelty smooth over the interval [a, b]. For such a function, a smooth quadratic interpolant like the one used in Simpson's rule will give results.

However, it is often the case that the function we are trying to integrate is not smooth over the interval. Typically, this means that either the function is highly oscillatory, or it lacks derivatives at certain points. In these cases, Simpson's rule may give really poor results. One common way of handling this problem is by breaking up interval [a, b] into n > 2 small subintervals. Simpson's rule is then applied to each subinterval, with the results being summed to produce an approximation for the integral over the entire interval. This sort of approach is termed the composite Simpson's rule. Suppose that the interval [a, b] is split up into n sub-intervals, with n an even number. Then, the composite simpson's rule is given by;

$$\int_a^b \approx \frac{h}{3}\left[f(x_0)+2\sum_{j=1}^{n/2-1} f(x_{2j})+4\sum_{j=1}^{n/2} f(x_{2j-1})+f(x_n)\right]$$

where $x_j = a + jh$ for j = 0, 1, 2, ..., n-1, n with $h=\frac{b-a}{n}$; in particular, $x_0 = a$ and $x_n = b$.
This composite rule with n= 2 correspons with the regular Simpson's Rule.

Composite Trapezoidal Method;

The trapezoidal rule gives us a technique to approximate the integral on a given interval [a, b], but we cannot reduce the error because the error depends on the width of the interval over which we are integrating.
By dividing the interval [a, b] into many smaller intervals, and applying the trapezoidal rule to each, this allows us to find a better approximation the integral.

$$\int_a^b f(x)\,dx \approx \frac{h}{3}[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+\cdots+2f(x_{n-2})+4f(x_{n-1})+f(x_n)],$$

Composite Midpoint Method;

Instead of approximating the territory under a bend by trapezoids, we can utilize plain square shapes. It might sound less exact to utilize flat lines and not slant lines following the capacity to be incorporated, yet a joining strategy dependent on square shapes (the

midpoint technique) is indeed marginally more precise than the one dependent on trapezoids!

$$\int_a^b f(x).dx \approx h\sum_{n=1}^{n-1} f(x_i)$$

Conclusion:

As a result, the error rates are so accurate as we see in the Table 1 below:

Table 1: Includes converge error rates of methods while finding stored energy.

Data files ↓ / Methods →	Composite Simpson's Method	Composite Trapezoidal Method	Composite Midpoint Method
pr3data.dat	0.077019	0.079698	0.198802

The plots of voltage, current and converged voltage values we have gathered with MATLAB code can be seen in the Figure 2 below;

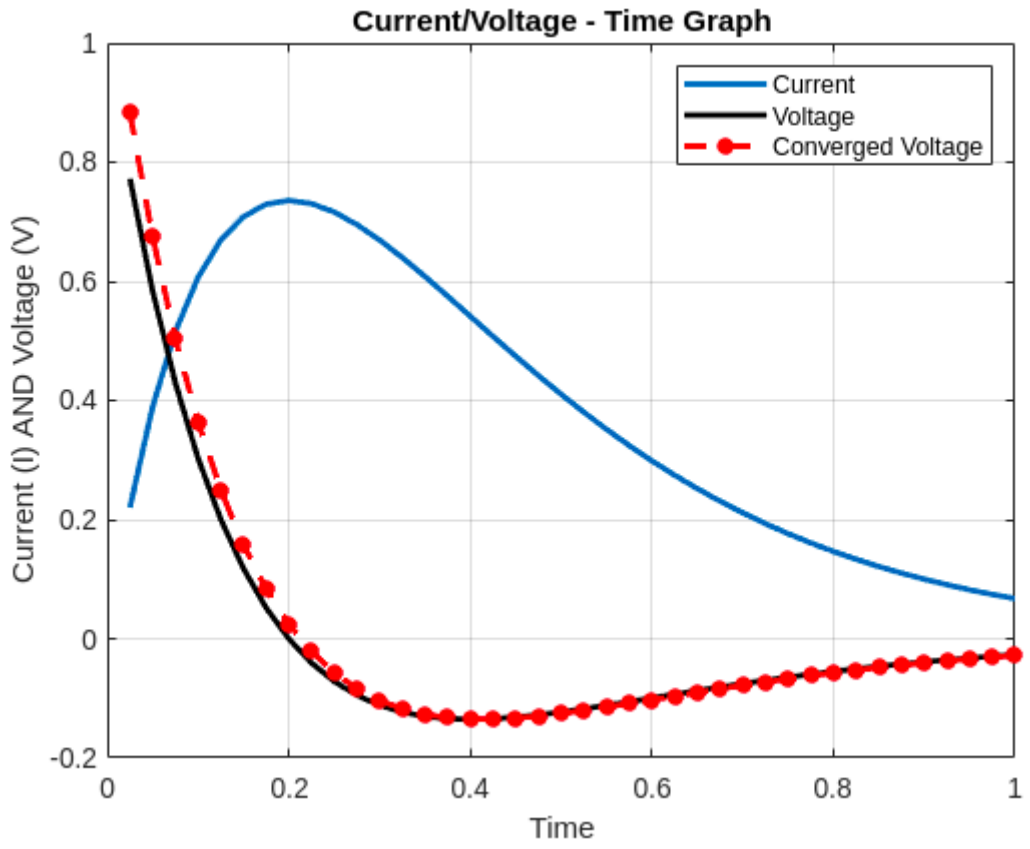

Figure 2: Current, Voltage and Converged Voltage plot with respect to time.

As specified in first part backward difference formula is used to find the converged voltage values with numerical methods. The results of the method are really accurate to proceed with them. Voltage values can be seen in Table 2 below;

Time	Voltage	Voltage (with B.D.)
0.500	0.5841	0.6751
0.225	-0.0406	-0.0212
0.4750	-0.1317	-0.1332
0.7250	-0.0700	-0.0727
1.000	-0.0270	-0.0283

The data on the Table 2 is taken randomly to compare voltage values from variable voltage_values_conc after running the MATLAB code.

The error rates of Simpson's Method and Trapezoidal Method are really precise Composite Midpoint Method has more error rate than others as seen on the table [Table 1]. Additionally, the converge success of each method can be seen in Figure 3 below;

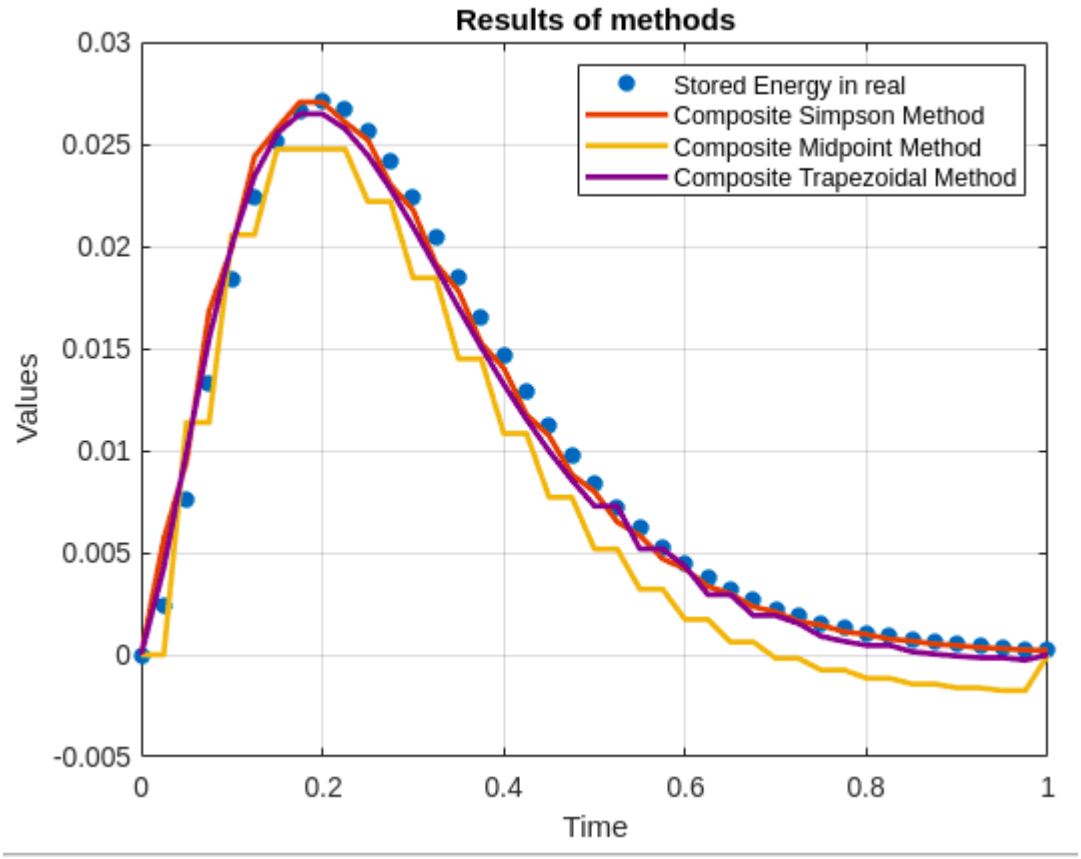

Figure 3: Results of all methods with respect to time.

Moreover, as previously discussed under the table 1, the success of each method can be seen in Figure 2 visually above. The composite Simpson's Method and composite trapezoidal method are really close to real values of energy stored in the inductor and the composite midpoint method is failed a bit. The midpoint method converges on the real value by passing through both two time periods. After that, even if that converges on the real value, its success chance reduces continuously.

There is one more thing I want to mention before jumping onto last part.

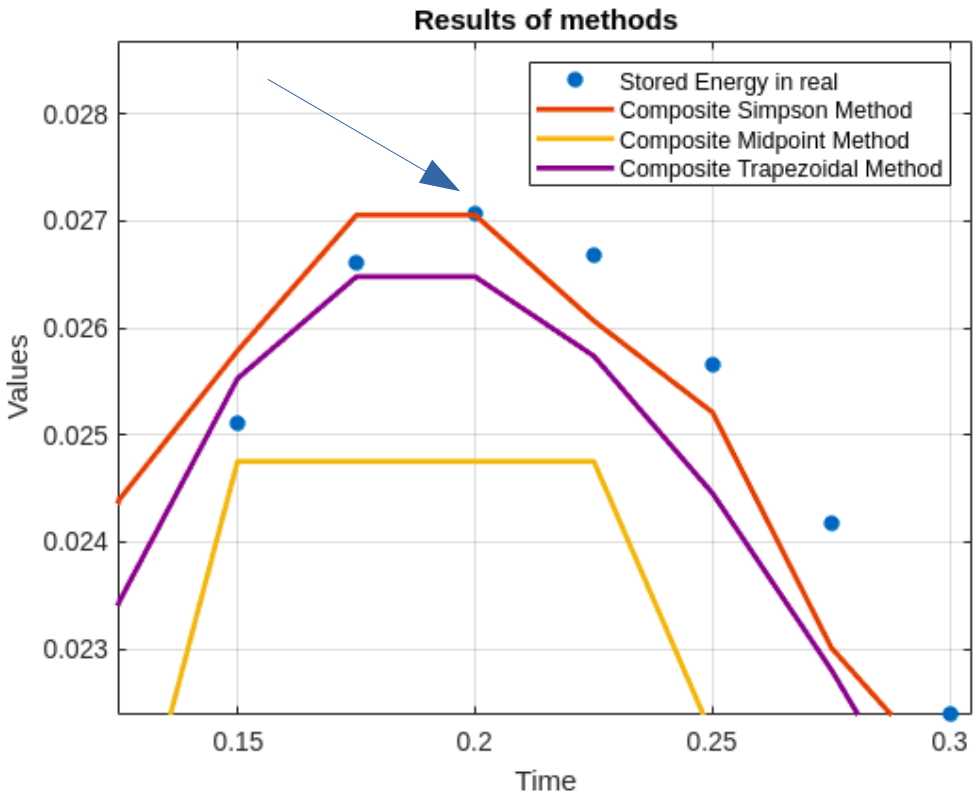

Figure 4: Simpson's method catches real value.

The Simpson's Method cathces the real value on the inflection point. After this point, while the graph is going curved The Simpson's method and Trapezoidal method catches or converges to real values more than midpoint formula do at some time intervals as seen in figure 5 and 6.

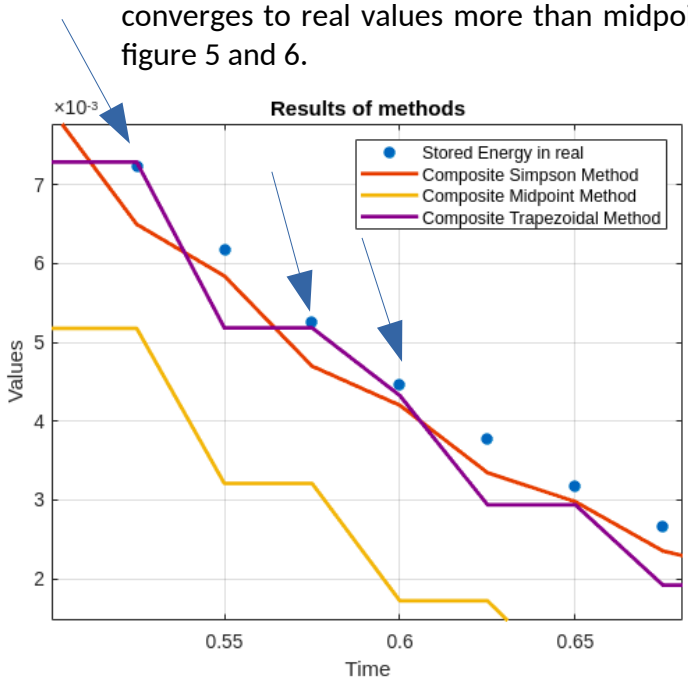

Figure 5: Curved part of the graph.

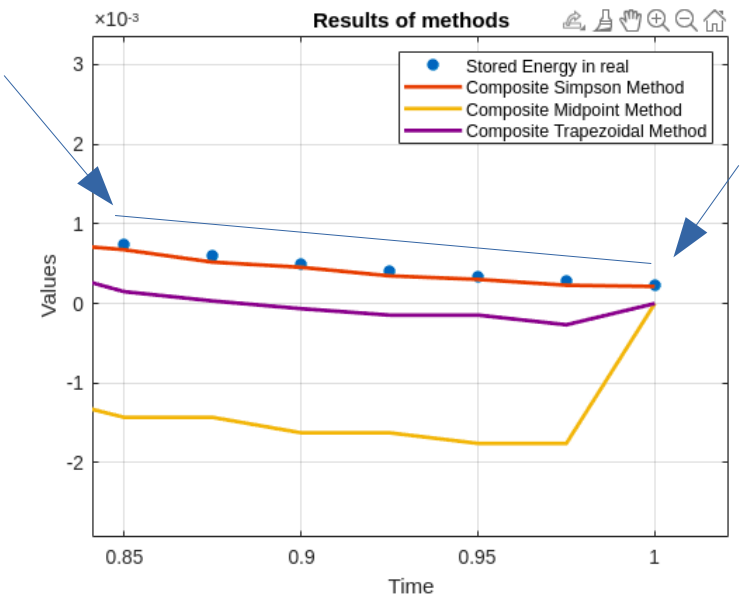

Figure 6: Last part of the graph.

In this report, all the methods are compared with their error rates and values on the same problem. The methods were pretty simple and we tested them all. The Simpson's method is more accurate than others as seen in Table 1. I did not quiet understand that the Compostie Midpoint Method converged more at the last part of the graph [Figure 6].

All the code I wrote for this project belongs to me and can be seen in the APPENDIX part.

APPENDIX

```
close all; clear all; clc;

L = 0.1; % 100 mH = 0.1 H
DT = 0.025;
stored_energy_simpsons(1,41) = zeros;
stored_energy_midpoint(1,41) = zeros;
stored_energy_trapezoidal(1,41) = zeros;

load pr3data.dat;

time_values = pr3data(:, 1)';
current_values = pr3data(:, 2)';
voltage_values = pr3data(:, 3)';

for i = 2:length(time_values)
   derivat_cur1_backward(i) = (current_values(i) - current_values(i-1)) / DT;
   voltage_e_cur1_backward(i) = (derivat_cur1_backward(i) * L);
end

% COmposite Simpson's rule
for j = 1:length(time_values)
   power(j) = voltage_values(j) * current_values(j);
   for k = 1:j
      if (k==1 || k==i)
         stored_energy_simpsons(j) = stored_energy_simpsons(j) + power(k);
      elseif(rem(k, 2) == 0)
         stored_energy_simpsons(j) = stored_energy_simpsons(j) + 4*power(k);
      elseif(rem(k, 2) == 1)
         stored_energy_simpsons(j) = stored_energy_simpsons(j) + 2*power(k);
      end
   end
   stored_energy_simpsons(j) = stored_energy_simpsons(j) * DT / 3;
end

% Composite Midpoint
for x = 1:length(time_values) - 1
   for t = 1:2:x
      stored_energy_midpoint(x) = stored_energy_midpoint(x) + 2*DT*power(t);
   end
end
```

```
for p = 2:length(time_values) - 1
    n = time_values(p)/DT;
    h = (time_values(p) - time_values(1)) / n;
    for y = 1:(n+1)
        if (j==1 || j==(n+1))
            stored_energy_trapezoidal(p) = stored_energy_trapezoidal(p) + power(y);
        else
            stored_energy_trapezoidal(p) = stored_energy_trapezoidal(p) + 2 * power(y);
        end
    end
    stored_energy_trapezoidal(p) = (h/2) * stored_energy_trapezoidal(p);
end

voltage_values_conc = vertcat(time_values, voltage_values, voltage_e_cur1_backward)';

% For the fourth question
stored_energy_eq3 = (L .* (current_values).^2) ./ 2;

plot(time_values(1,2:end), current_values(1,2:end), time_values(1,2:end), ...
voltage_values(1,2:end),'k', time_values(1,2:end), voltage_e_cur1_backward(1,2:end), 'r--*',
'LineWidth', 2);
grid on;
title('Current/Voltage - Time Graph');
xlabel('Time'); ylabel('Current (I) AND Voltage (V)');
legend('Current', 'Voltage', 'Converged Voltage');

figure(2);
plot(time_values, stored_energy_eq3, '*', time_values, stored_energy_simpsons, time_values,
stored_energy_midpoint, time_values, stored_energy_trapezoidal, 'LineWidth', 2);
grid on;
title('Results of methods');
xlabel('Time'); ylabel('Values');
legend('Stored Energy in real', 'Composite Simpson Method', 'Composite Midpoint Method', 'Composite
Trapezoidal Method');

% Error Calculations
fprintf('Error rate of composite Simpson formula = %f\n', (norm(stored_energy_simpsons -
stored_energy_eq3) / norm(stored_energy_simpsons)));
fprintf('Error rate of composite midpoint formula = %f\n', (norm(stored_energy_midpoint -
stored_energy_eq3) / norm(stored_energy_midpoint)));
fprintf('Error rate of composite trapezoidal method = %f\n', (norm(stored_energy_trapezoidal -
stored_energy_eq3) / norm(stored_energy_trapezoidal)));
```