ELECTRONIC ENGINEERING DEPARTMENT

MATH 214 NUMERICAL METHODS

2020 – 2021 FALL

PROJECT 4

Student ID :1901022039

Name Surname :Alperen Arslan

Preperation date: 25.12.2020

Upload date: 25.12.2020

1. Problem Definition and Formulation

The Voltage formula at a RL circuit is given at equation (1).

$$V_S = L\frac{d}{dt}i(t) + Ri(t) \tag{1}$$

Since the voltage is constant at every time, the current flows through the circuit at any time can be calculated by solving this differential equation. The problem is to calculate the current flowing through the circuit using the Euler's method, the modified Euler method, the midpoint method, and the Runge-Kutta fourth order method, and then analyze the results to find which one approximates the best.

Let there is a differential equation such like:

$$\frac{dy}{dx} = f(x, y) \tag{2}$$

i. Euler's method

Euler's method is based on a simple idea. Assuming the slope of the graph of given function is constant at given $[h_i, h_{i+1}]$ interval. The derivative and the y value can be found by this assumption.

By using equation (2), the Euler's method formula can be written as:

$$y_i^0 = y_{i-1} + f(x_{i-1}, y_{i-1})h \tag{3}$$

i=step number, h=rate of change of x

The Euler's method is also known as first order Runge-Kutta method.

ii. Modified Euler's method

For applying the Modified Euler method, Euler's method is applied first. After That, the average of y_i and y_{i+1} is calculated. Drawing another slope with using this average value will reduce the error.

The formula of modified Euler's method is:

$$y_i^1 = y_{i-1}^1 + \frac{h}{2}\left(f(x_{i-1}, y_{i-1}) + f\left(x_i, y_i^0\right)\right) \tag{4}$$

iii. Midpoint method

Midpoint method is also known as second order Runge-Kutta method.

$$\begin{aligned} k_1 &= f(x_i, y_i) \\ k_2 &= f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right) \\ y_{i+1} &= y_i + h(k_2) \end{aligned} \tag{5}$$

K_1 is the slope at the beginning of the interval, using Euler's method,

K_2 is the slope at the midpoint of the interval

iv. Runge-Kutta method order four

$$\begin{aligned} k_1 &= f(x_i, y_i) \\ k_2 &= f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right) \\ k_3 &= f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right) \end{aligned}$$

$$k_4 = f(x_i + h, y_i + hk_3)$$
$$y_{i+1} = y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3k_4) \quad (6)$$

K_1 is the slope at the beginning of the interval, using Euler's method,
K_2 is the slope at the midpoint of the interval,
K_3 is the again midpoint, but now using y and k_2,
K_4 is the slope at the end of the interval, using y and k_3

v. Analytical Solution

Equation (1) is a linear 1st order differential equation therefore this equation can be solved by using integrating factors method.
In this problem V=12 v, R=14.2 Ω and L=0.98 H values are given.
Putting these variables into the equation (1):

$$0.98\frac{di}{dt} + 14.2i = 12 \quad (7)$$

If we take the coefficient of i for applying integrating factors method,

$$m = e^{\int 14{,}2dt}$$
$$m = e^{14.2t}$$

Multiplying both sides of equation (7) with m,

$$e^{14.2t}\left(0.98\frac{di}{dt} + 14.2i\right) = 12e^{14.2t}$$
$$\frac{d}{dt}(e^{14.2t}i) = 12e^{14.2t}$$
$$\int \frac{d}{dt}(e^{14.2t}i) = \int 12e^{14.2t}$$
$$e^{14.2t}i = \frac{12}{14.2}e^{14.2t} + c$$
$$i(t) = \frac{12}{14.2} + \frac{c}{e^{14.2t}}$$

Since i(0)=0.1 A is known, c can be found as:

$$i(0) = 0.1 = \frac{12}{14.2} + c$$
$$c = -\frac{529}{710} = -0.74507 \ldots$$

And the analytical solution of this differential equation is:

$$i(t) = \frac{12}{14.2} - \frac{529}{710}e^{-14.2t} \quad (8)$$

2. Codes and Inputs

All results and figures were generated with the code which is given at appendix section. First the given R, L, V variables and the function were assigned to proper variables. After that first step size(h) is assigned to 0.05 and 0.025. For each step size, the current value was found by using Euler's method in equation (3), modified Euler's method in equation (4), midpoint method in equation (4) and Runge-kutta method order four in equation (5). Then, the real Solution is calculated by using Equation (8). After that, the graphs of all results were plotted and error values were found.

The formula for calculating the error is:

$$error = \frac{|real - experimental|}{real} \tag{9}$$

3. Project Results and Discussions

At this section, the results which generated by the code were explained and discussed.

A) Current Values for All Cases

The current values for all cases are shown in Figure 1 and Figure 2.
It can be seen in Figure 1 that the Euler's method is very far away from the real solution especially at [0,0.3] s time interval.
And since the modified Euler's method is using two slopes instead of one, it can be expected that the converge rate is closer to the actual value than Euler's method.
It is also be expected that the midpoint method converges better than both Euler's method and modified Euler's method because it uses more iterations than both two methods. And since the Runge-Kutta order four method has the most iteration number, the convergence rate will be the best.
Also since the lower time step sizes means more detailed data sets it can be expected that the convergence rate is much more accurate at h=0.025 than h=0.05.

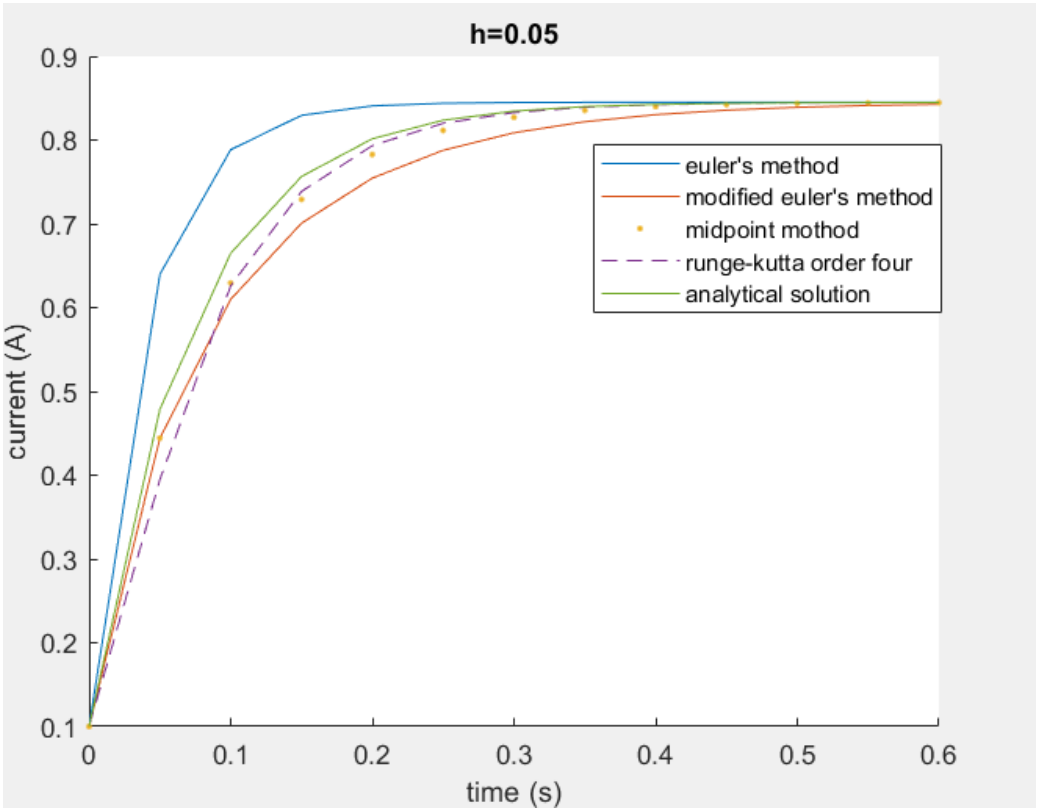

Figure 1. current versus time graph for h=0.05 step size

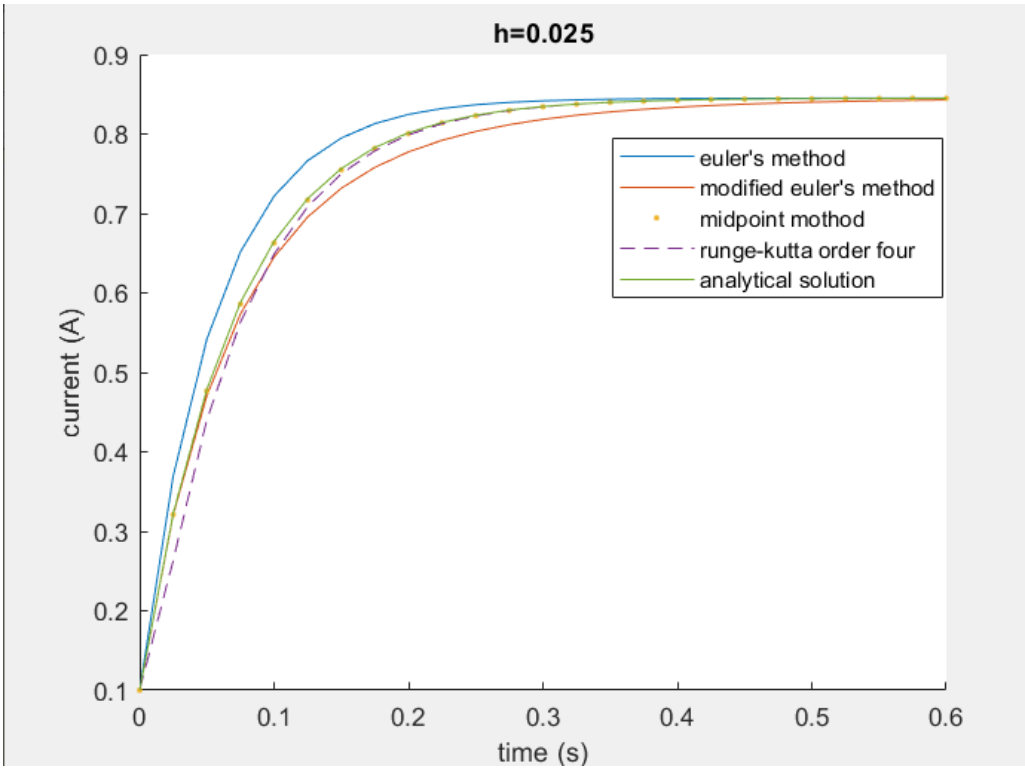

Figure 2. current versus time graph for h=0.025 step size

B) Error Analysis for All Cases with Respect to The Analytical Solution

The real value of the current could be determined by using equation (8). If the error is calculated by using absolute error formula which given at equation (9), the error rates are given at Table 1 and Table 2. From comparing Table 1 and Table 2, it could be seen that the convergence rates are much better when h is 0.025 than 0.05.

Even though midpoint method gives more accurate results than Runge-Kutta method at some points, from Table 1 and Table 2, it can be said that Runge-Kutta method order four is the most accurate method in generally.

Table 1. error rates for h=0.05

time(s)	euler	modified euler	Midpoint method	Runge-Kutta
0.00	0.000000	0.000000	0.000000	0.000000
0.05	0.539796	0.344258	0.344258	0.294685
0.10	0.309754	0.131176	0.150691	0.147398
0.15	0.506842	0.378110	0.406431	0.416021
0.20	0.175801	0.089612	0.117695	0.128377
0.25	0.365127	0.309033	0.332742	0.341172
0.30	0.088216	0.052132	0.070484	0.076324
0.35	0.256757	0.233658	0.247132	0.250908
0.40	0.043507	0.028752	0.038305	0.040645
0.45	0.180087	0.170671	0.177283	0.178691
0.50	0.021400	0.015393	0.019890	0.020720
0.55	0.126277	0.122445	0.125464	0.125946

0.60	0.010522	0.008078	0.010084	0.010361

Table 2. error rates for h=0.025

time(s)	euler	modified euler	Midpoint method	runge kutta
0.000	0.000000	0.000000	0.000000	0.000000
0.025	0.269898	0.221013	0.221013	0.163637
0.050	0.063266	0.007941	0.002294	0.038462
0.075	0.329156	0.250963	0.263160	0.240638
0.100	0.056837	0.019874	0.002263	0.016074
0.125	0.287701	0.216798	0.238046	0.229745
0.150	0.038409	0.024809	0.001674	0.006529
0.175	0.224874	0.169816	0.193392	0.190662
0.200	0.023141	0.024043	0.001101	0.002544
0.225	0.167090	0.127115	0.148701	0.148018
0.250	0.013108	0.020480	0.000679	0.000928
0.275	0.120988	0.092939	0.110746	0.110732
0.300	0.007149	0.016167	0.000402	0.000300
0.325	0.086391	0.067077	0.080859	0.081008
0.350	0.003801	0.012156	0.000231	0.000072
0.375	0.061208	0.048052	0.058282	0.058432
0.400	0.001985	0.008846	0.000130	0.000002
0.425	0.043175	0.034269	0.041651	0.041762
0.450	0.001023	0.006293	0.000072	0.000019
0.475	0.030378	0.024372	0.029593	0.029665
0.500	0.000522	0.004405	0.000040	0.000018
0.525	0.021343	0.017302	0.020942	0.020986
0.550	0.000265	0.003048	0.000021	0.000013
0.575	0.014982	0.012267	0.014779	0.014805
0.600	0.000133	0.002092	0.000012	0.000008

4. Conclusion

From the final results, it can be said that smaller step sizes give more accurate results. Also it can be said that the Runge-Kutta method order four gives the most accurate results. If a comparison between these four methods is apply it can be said that the best method is the Runge-Kutta method order four, midpoint method is the second best method, there is small difference between these methods. The third best method is the modified Euler's method and least accurate method is Euler's method.

5. Appendix

The code is given below:

```
clear all;
format long;
clc;
R=14.2;
L=0.98;
V=12;
f = @(x) (V-R*x)/L;
```

6. euler's method for h=0.05

```
h=0.05;
current_euler(1,1)=0.1;% first row is for h=0.05

derivative_current_euler(1,1)=(V-R*current_euler(1,1))/L;
for i=2:13
current_euler(1,i)=current_euler(1,i-1)+derivative_current_euler(1,i-1)*h;
derivative_current_euler(1,i)=(V-R*current_euler(1,i))/L;
end
```

7. modified euler's method for h=0.05

```
h=0.05;
modified_euler(1,1)=0.1;% first row is for h=0.05
modified_euler(1,13)=0;
derivative_m_euler(1,1)=(V-R*modified_euler(1,1))/L;
derivative_m_euler(1,13)=0;
for i=2:13
modified_euler(1,i)=modified_euler(1,i-1)+(h/2)*(derivative_m_euler(1,i-
1)+derivative_current_euler(1,i));
derivative_m_euler(1,i)=(V-R*modified_euler(1,i))/L;
end
```

8. midpoint method for h=0.05

```
h=0.05;
midpoint(1,1)=0.1;
k1(1)=(V-R*midpoint(1,1))/L;
k2(1)=f(midpoint(1,1)+0.5*h*k1(1));
for i=2:13
   midpoint(1,i)= midpoint(1,i-1) + k2(i-1)*h;
   k1(i)=f(midpoint(1,i));
k2(i)=f(midpoint(1,i)+0.5*h*k1(i));
end
```

9. runge kutta method order four for h=0.05

```
h=0.05;
for i=1:25
    k1(i)=0;k2(i)=0; %just to be sure dont get anything wrong
end
```

```
runge_kutta(1,1)=0.1;
k1(1)=f(runge_kutta(1,1));
k2(2)=f(runge_kutta(1,1)+0.5*k1(1)*h);
k3(1)=f(runge_kutta(1,1)+0.5*k2(1)*h);
k4(1)=f(runge_kutta(1,1)+k3(1)*h);

for i=2:13
runge_kutta(1,i)=runge_kutta(1,i-1)+(1/6)*(k1(i-1) + 2*k2(i-1) + 2*k3(i-1) + k4(i-
1))*h;
k1(i)=f(runge_kutta(1,i));
k2(i)=f(runge_kutta(1,i)+0.5*k1(i)*h);
k3(i)=f(runge_kutta(1,i)+0.5*k2(i)*h);
k4(i)=f(runge_kutta(1,i)+k3(i)*h);
end
```

10. analytical solution for h=0.05

```
h=0.05;
y=@(x) 12/14.2-(529/710)*exp(-14.2*x);
for i=0:12
    analytical(1,i+1)=y(h*i);
end
```

11. euler's method for h=0.025

```
h=0.025;
current_euler(2,1)=0.1;% second row is for h=0.025
derivative_current_euler(2,1)=(V-R*current_euler(2,1))/L;
for i=2:25
   current_euler(2,i)=current_euler(2,i-1)+derivative_current_euler(2,i-1)*h;
   derivative_current_euler(2,i)=(V-R*current_euler(2,i))/L;
end
```

12. modified euler's method for h=0.025

```
h=0.025;
modified_euler(2,1)=0.1;
derivative_m_euler(2,1)=(V-R*modified_euler(2,1))/L;
for i=2:25
   modified_euler(2,i)=modified_euler(2,i-
1)+(h/2)*(derivative_current_euler(2,i)+derivative_m_euler(2,i-1));
   derivative_m_euler(2,i)=(V-R*modified_euler(2,i))/L;
end
```

13. midpoint method for h=0.025

```
h=0.025;
midpoint(2,1)=0.1;
midpoint(2,25)=0;
a1(1)=(V-R*midpoint(2,1))/L;
a2(1)=f(midpoint(2,1)+0.5*h*k1(1));
a1(25)=0;
a2(25)=0;
for i=2:25
   midpoint(2,i)=midpoint(2,i-1)+a2(i-1)*h;
```

```
    a1(i)=f(midpoint(2,i));
    a2(i)=f(midpoint(2,i)+0.5*h*a1(i));
end
```

14. runge-kutta order four for h=0.025

```
h=0.025;

for i=1:25
    k1(i)=0;k2(i)=0; %just to be sure dont get anything wrong
end
runge_kutta(2,1)=0.1;
runge_kutta(2,25)=0;
k1(1)=f(runge_kutta(2,1));
k2(2)=f(runge_kutta(2,1)+0.5*k1(1)*h);
k3(1)=f(runge_kutta(2,1)+0.5*k2(1)*h);
k4(1)=f(runge_kutta(2,1)+k3(1)*h);

for i=2:25
runge_kutta(2,i)=runge_kutta(2,i-1)+(1/6)*(k1(i-1) + 2*k2(i-1) + 2*k3(i-1) + k4(i-
1))*h;
k1(i)=f(runge_kutta(2,i));
k2(i)=f(runge_kutta(2,i)+0.5*k1(i)*h);
k3(i)=f(runge_kutta(2,i)+0.5*k2(i)*h);
k4(i)=f(runge_kutta(2,i)+k3(i)*h);
end
```

15. analytical solution for h=0.025

```
h=0.025
for i=0:24
   analytical(2,i+1)=y(h*i);
end
```

16. graphs

```
 x_1=linspace(0,0.6,13);
 x_2=linspace(0,0.6,25);
 figure('name','h=0.05');
 hold on;
 plot(x_1,current_euler(1,1:13));
 plot(x_1,modified_euler(1,1:13));
 plot(x_1,midpoint(1,1:13),".");
 plot(x_1,runge_kutta(1,1:13),"--");
 plot(x_1,analytical(1,1:13));
 xlabel("time (s)");
 ylabel("current (A)");
 hold off;
 legend("euler's method","modified euler's method","midpoint mothod","runge-kutta
order four","analytical solution");
 title("h=0.05");

 figure("name","h=0.025");
 title("h=0.025");
 hold on;
```

```
  plot(x_2,current_euler(2,:));
  plot(x_2,modified_euler(2,:));
  plot(x_2,midpoint(2,:),".");
  plot(x_2,runge_kutta(2,:),"--");
  plot(x_2,analytical(2,1:25));
  xlabel("time (s)");
  ylabel("current (A)");
  hold off;
  legend("euler's method","modified euler's method","midpoint mothod","runge-kutta
order four","analytical solution");
```

error analysis

```
%error for h=0.05
for i=1:13
   error_euler(1,i)=abs(analytical(i)-current_euler(1,i));
   error_modified(1,i)=abs(analytical(i)-modified_euler(1,i));
   error_midpoint(1,i)=abs(analytical(i)-midpoint(1,i));
   error_runge_kutta(1,i)=abs(analytical(i)-runge_kutta(1,i));
end
%error for h=0.025
for i=1:25
   error_euler(2,i)=abs(analytical(i)-current_euler(2,i));
   error_modified(2,i)=abs(analytical(i)-modified_euler(2,i));
   error_midpoint(2,i)=abs(analytical(i)-midpoint(2,i));
   error_runge_kutta(2,i)=abs(analytical(i)-runge_kutta(2,i));
end
   fprintf("errors for h=0.05\n");
   fprintf("%s\t  %-6s %10s  %8s  %-8s\n","time(s)","euler","modified euler","midpoint","runge
kutta");
for i=0:12
fprintf("%.2f\t%f\t%f\t%f\t%f\n",0.05*i,error_euler(1,i+1),error_modified(1,i+1),error_midpoin
t(1,i+1),error_runge_kutta(1,i+1));
end
fprintf("\n\n\n");
  fprintf("errors for h=0.025\n");
   fprintf("%s\t  %-6s %10s  %8s  %-8s\n","time(s)","euler","modified euler","midpoint","runge
kutta");
for i=0:24
fprintf("%.3f\t%f\t%f\t%f\t%f\n",0.025*i,error_euler(2,i+1),error_modified(2,i+1),error_midpoi
nt(2,i+1),error_runge_kutta(2,i+1));
end
analytical(1,1:25))/sqrt(25));
fprintf(" average error for h=0.025  runge kutta order four:%f\n",norm(runge_kutta(1,1:25)-
analytical(1,1:25))/sqrt(25));
```

```
errors for h=0.05
time(s)   euler       modified euler  midpoint    runge kutta
0.00    0.000000       0.000000       0.000000       0.000000
0.05    0.539796       0.344258       0.344258       0.294685
0.10    0.309754       0.131176       0.150691       0.147398
0.15    0.506842       0.378110       0.406431       0.416021
0.20    0.175801       0.089612       0.117695       0.128377
0.25    0.365127       0.309033       0.332742       0.341172
0.30    0.088216       0.052132       0.070484       0.076324
0.35    0.256757       0.233658       0.247132       0.250908
0.40    0.043507       0.028752       0.038305       0.040645
```

0.45	0.180087	0.170671	0.177283	0.178691
0.50	0.021400	0.015393	0.019890	0.020720
0.55	0.126277	0.122445	0.125464	0.125946
0.60	0.010522	0.008078	0.010084	0.010361

errors for h=0.025

time(s)	euler	modified euler	midpoint	runge kutta
0.000	0.000000	0.000000	0.000000	0.000000
0.025	0.269898	0.221013	0.221013	0.163637
0.050	0.063266	0.007941	0.002294	0.038462
0.075	0.329156	0.250963	0.263160	0.240638
0.100	0.056837	0.019874	0.002263	0.016074
0.125	0.287701	0.216798	0.238046	0.229745
0.150	0.038409	0.024809	0.001674	0.006529
0.175	0.224874	0.169816	0.193392	0.190662
0.200	0.023141	0.024043	0.001101	0.002544
0.225	0.167090	0.127115	0.148701	0.148018
0.250	0.013108	0.020480	0.000679	0.000928
0.275	0.120988	0.092939	0.110746	0.110732
0.300	0.007149	0.016167	0.000402	0.000300
0.325	0.086391	0.067077	0.080859	0.081008
0.350	0.003801	0.012156	0.000231	0.000072
0.375	0.061208	0.048052	0.058282	0.058432
0.400	0.001985	0.008846	0.000130	0.000002
0.425	0.043175	0.034269	0.041651	0.041762
0.450	0.001023	0.006293	0.000072	0.000019
0.475	0.030378	0.024372	0.029593	0.029665
0.500	0.000522	0.004405	0.000040	0.000018
0.525	0.021343	0.017302	0.020942	0.020986
0.550	0.000265	0.003048	0.000021	0.000013
0.575	0.014982	0.012267	0.014779	0.014805
0.600	0.000133	0.002092	0.000012	0.000008