# NUMERİCAL METHODS/MATH214

*PROJECT 4*

Initial-Value Problems for Ordinary Differential Equations

- Euler's Method,
- Modifed Euler's Method,
- Midpoint Method,
- Runge-Kutta Method Order Four,

Muhammed Cemal Eryiğit
1801022024

Deadline 25/12/2020 – 17.00

Preparation Time 23-25/12/2020

**1- Problem definition and introduction formulas**

Using the equation of the Kirchoff voltage law, it is solving the equation (1) with 4 numerical method by using the current provided by the energy stored by the inductance in the circuit completed at the time t 0

$$Vs = L\frac{di(t)}{dt} + Ri(t), i(t_0) = i_0 \tag{1}$$

A) **Euler method** is a first-order numerical method for solving ordinary differential equations (ODEs) with a given initial value. It is the most elementary method for numerical derivation of initial value problems for ordinary differential equations and is the simplest Runge–Kutta method. The object of Euler's method is to obtain approximations to the well-posed initial-value with this formula;

$$y'(x) = f\big(x, y(x)\big)y(0) = y_0 y_{n+1} = y_n + h * f(x_n + y_n)$$

B) **Midpoint method** is a 2nd order Runge-Kutta method.It allows us to find a value at a point in ordinary differential equations, which is the first order initial value problem with this formula;

$$y'(x) = f\big(x, y(x)\big) \quad y(0) = y_0 \qquad y_{n+1} = y_n + h * f(x_n + \frac{h}{2}, y_n + \frac{h}{2} f(x_n, y_n)$$

C) **Modified Euler's method** is an improved version of the midpoint method. The difference is determined when the two formulas are compared. The modified Euler's method formula;

$$y'(x) = f\big(x, y(x)\big)y(0) = y_0 y_{n+1} = y_n + \frac{h}{2} * \big[f(x_n, y_n) + f\big(x_{n+1}, y_n + h * f(x_n, y_n)\big)\big]$$

D) **Runge-Kutta method order four:** The most common Runge-Kutta method in use is the fourth order. Its formula is given below.

$$y'(x) = f\big(x, y(x)\big) \quad y(0) = y_0$$

$$k_1 = h * f(x_n, y_n)$$

$$k_2 = h * f\left(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_1\right)$$

$$k_3 = h * f\left(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_2\right)$$

$$k_3 = h * f(x_{n+1}, y_n + k_3)$$

$$y_{n+1} = y_n + \frac{1}{6} * (k_1 + 2k_2 + 2k_3 + k_4)$$

The equation given in equation (1) cannot be used in this way so we transform the equation into a differentiable form. The new equation is given below.

$$\frac{d}{dt}i(t) = \frac{v_s - R * i(t)}{L}$$

If we want to do the analytical solution of this equation, we do as follows;

$$\frac{d}{dt}i(t) = \frac{v_s - R * i(t)}{L} \qquad i = \frac{V_s}{R}\left(1 - e^{-(R/L)t}\right)$$

Values of unknowns in the equation is: Vs = 12V, R=14.2Ω, *t* = 0.6 s, L = 0.98 H

$$i = \frac{12}{14.2}\left(1 - e^{-(14.2/0.98)0.6}\right)$$

The result is this $i = 0.8449$

**2- Code description and inputs**

All result and figures are generated using the code, which is given at the appendix section. The values given in the problem are Vs = 12V, R=14.2Ω, *t* = 0.6 s, L = 0.98 H. The h1 and h2 values used in the code are time intervals. h1=0.05, h2=0.025. Since h = (b-a) / n, the intervals are n1 = 12 n2 = 24. B and a values are the start and end time. a=0, b=0.6.

In the introduction part of the code, there is the first part where all the values are defined. All methods are applied in order and the results are recorded in array the next sections. w array is reset after each method and prepared to apply and save the new method. After the methods are applied, the values are recorded in the table (table 1 and table 2) then the values are shown in the graph. (figure 1 and figure 2)

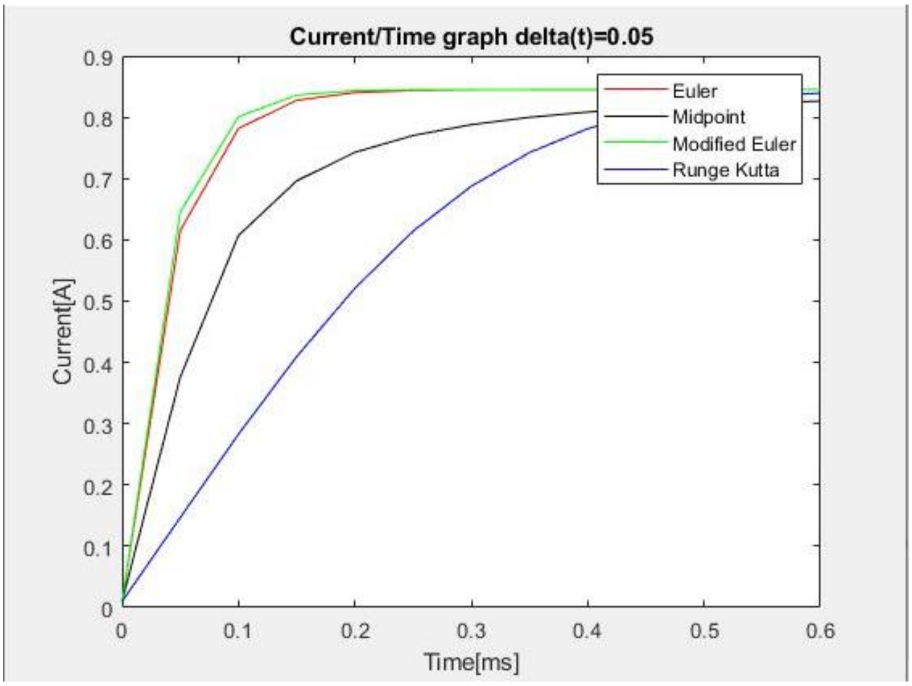
| Euler Method | Modified Euler Method | Midpoint Method | Runge-Kutta Method order four |
|---|---|---|---|
| 0.0100 | 0.0100 | 0.0100 | 0.0100 |
| 0.6150 | 0.6453 | 0.3760 | 0.1468 |
| 0.7817 | 0.8009 | 0.6070 | 0.2826 |
| 0.8276 | 0.8361 | 0.6962 | 0.4094 |
| 0.8403 | 0.8434 | 0.7427 | 0.5210 |
| 0.8437 | 0.8448 | 0.7702 | 0.6140 |
| 0.8447 | 0.8450 | 0.7879 | 0.6874 |
| 0.8450 | 0.8451 | 0.7999 | 0.7424 |
| 0.8450 | 0.8451 | 0.8085 | 0.7813 |
| 0.8451 | 0.8451 | 0.8148 | 0.8075 |
| 0.8451 | 0.8451 | 0.8196 | 0.8241 |
| 0.8451 | 0.8451 | 0.8263 | 0.8395 |

(Table-1 for h1=0.05)

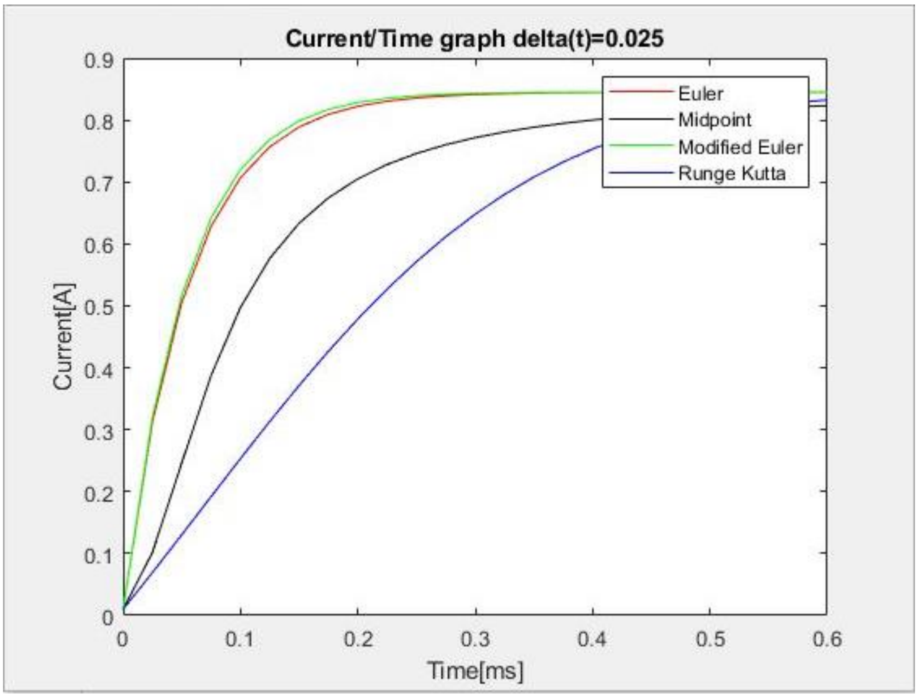
| Euler Method | Modified Euler Method | Midpoint Method | Runge-Kutta Method order four |
|---|---|---|---|
| 0.0100 | 0.0100 | 0.0100 | 0.0100 |
| 0.3125 | 0.3201 | 0.1015 | 0.0693 |
| 0.5054 | 0.5174 | 0.2466 | 0.1303 |
| 0.6285 | 0.6420 | 0.3876 | 0.1920 |
| 0.7069 | 0.7202 | 0.4974 | 0.2533 |
| 0.7570 | 0.7688 | 0.5167 | 0.3134 |
| 0.7889 | 0.7989 | 0.6334 | 0.3714 |
| 0.8092 | 0.8173 | 0.6746 | 0.4267 |
| 0.8222 | 0.8285 | 0.7051 | 0.4788 |
| 0.8305 | 0.8352 | 0.7282 | 0.5271 |
| 0.8391 | 0.8393 | 0.7461 | 0.5715 |
| 0.8413 | 0.8417 | 0.7603 | 0.6118 |
| 0.8427 | 0.8431 | 0.7716 | 0.6479 |
| 0.8435 | 0.8439 | 0.7884 | 0.6800 |
| 0.8441 | 0.8444 | 0.7947 | 0.7081 |
| 0.8444 | 0.8447 | 0.8000 | 0.7324 |
| 0.8447 | 0.8449 | 0.8046 | 0.7533 |
| 0.8448 | 0.8450 | 0.8084 | 0.7710 |
| 0.8449 | 0.8450 | 0.8118 | 0.7982 |
| 0.8450 | 0.8450 | 0.8147 | 0.8083 |
| 0.8450 | 0.8451 | 0.8172 | 0.8165 |
| 0.8450 | 0.8451 | 0.8195 | 0.8231 |
| 0.8450 | 0.8451 | 0.8214 | 0.8284 |
| 0.8451 | 0.8451 | 0.8232 | 0.8425 |

(Table-2 for h2=0.025)

When the values shown in the table are plotted on the figures




(Figure-1)




(Figure-2)

## 3- Project Result and Analysis

Comment on error rates and comparisons between methods will be evaluate in this section.

A- Error Rate

Since it is a function that we can calculate, it can compare analytically calculated values with numerically calculated values and obtain an error rate.

analytical solution $i = 0.8449$

Methods values for h=0.05

Euler Method: $i = 0.8451$

Modified Euler's Method: : $i = 0.8451$

Midpoint Method: : $i = 0.8232$

Runge-Kutta Method Order Four: : $i = 0.8395$

Methods values for h=0.025

Euler Method: $i = 0.8451$

Modified Euler's Method: : $i = 0.8451$

Midpoint Method: : $i = 0.8232$

Runge-Kutta Method Order Four: : $i = 0.8425$

The percentage error rate is calculated as :$\frac{|analytical\ solution - numerical\ solution|}{analytical\ solution} * 100$

| | **Time Interval h=0.05** | **Time Interval h=0.025** |
|---|---|---|
| Euler Method error rate | 0.0236 | 0.0236 |
| Modified Euler Method error rate | 0.0236 | 0.0236 |
| Midpoint Method Error rate | 2.568 | 2.568 |
| Runge-Kutta Method Order Four Error rate | 0.6391 | 0.2840 |

(Table-3 Percentage Error Rate)

B- Analysis

When the error rates are examined, the rates of the Euler method and the modified Euler method are the same and have the least error rates. When the error rates are examined, the rates of the Euler method and the modified Euler method are the same and have the least error rates.

We observe that the Runge-Kutta method has less error rates after the Euler and modified Euler methods. We reached more accurate result with decrease of interval in Runge-Kutta method order four. (Number of terms increases when the interval decreases)

It is the midpoint method with the highest error rate.

## 4- Appendix Section

# Code

```
clear
clear all
clc
%definition
a=0; b=0.6;
h1=0.05; h2=0.025;
n1=(b-a)/h1; n2=(b-a)/h2;
L=0.98; Vs=12; R=14.2;
w=zeros(12,1);
w(1,1)=0.01; t1= a:h1:b; t2= a:h2:b;
%euler method
for k=1: length(w)
    w(k+1)=w(k)+h1*((Vs-R*w(k))/L);
end
figure
plot(t1,w,'r-');
hold on
w=zeros(12,1);
w(1,1)=0.01;
```

```
%midpoint method
for k=1: length(w)
    w(k+1)=w(k)+h1*((h1*k*((Vs-R*w(k))/L))*(Vs-R*w(k))/L);
end
plot(t1,w,'k-');
hold on
w=zeros(12,1);
w(1,1)=0.01;
%modified euler method
for k=1: length(w)
    w(k+1)=w(k)+h1/2*(((Vs-R*w(k))/L)+(((h1*k+1)+h1)*(Vs-R*w(k))/L));
end
plot(t1,w,'g-');
hold on
w=zeros(12,1);
w(1,1)=0.01;
%runge-kutta method order four
for k=1: length(w)
    k1=h1*((Vs-R*w(k))/L);
    k2=h1*((h1*k*((Vs-R*w(k))/L))+(k1/2));
    k3=h1*((h1*k*((Vs-R*w(k))/L))+(k2/2));
    k4=h1*(h1*(k+1)*((Vs-R*w(k))/L)+k3);
    w(k+1)=w(k)+1/6*(k1+2*k2+2*k3+k4);
end
plot(t1,w,'b-');
hold on
xlabel('Time[ms]')
ylabel('Current[A]')
title('Current/Time graph delta(t)=0.05');
legend('Euler','Midpoint','Modified Euler','Runge Kutta');
w=zeros(24,1);
w(1,1)=0.01;

%euler method
for k=1: length(w)
    w(k+1)=w(k)+h2*((Vs-R*w(k))/L);
end
figure
plot(t2,w,'r-');
hold on
w=zeros(24,1);
w(1,1)=0.01;
%midpoint method
for k=1: length(w)
    w(k+1)=w(k)+h2*((h2*k*((Vs-R*w(k))/L))*(Vs-R*w(k))/L);
end
plot(t2,w,'k-');
hold on
w=zeros(24,1);
w(1,1)=0.01;
%modified euler method
for k=1: length(w)
    w(k+1)=w(k)+h2/2*(((Vs-R*w(k))/L)+(((h2*k+1)+h2)*(Vs-R*w(k))/L));
end
plot(t2,w,'g-');
hold on
w=zeros(24,1);
w(1,1)=0.01;
%runge-kutta method order four
for k=1: length(w)
    k1=h2*((Vs-R*w(k))/L);
```

```
    k2=h2*((h2*k*((Vs-R*w(k))/L))+(k1/2));
    k3=h2*((h2*k*((Vs-R*w(k))/L))+(k2/2));
    k4=h2*(h2*(k+1)*((Vs-R*w(k))/L)+k3);
    w(k+1)=w(k)+1/6*(k1+2*k2+2*k3+k4);
end
plot(t2,w,'b-');
hold on
xlabel('Time[ms]')
ylabel('Current[A]')
title('Current/Time graph delta(t)=0.025');
legend('Euler','Midpoint','Modified Euler','Runge Kutta');
```