HOMEWORK-4

In numerical analysis, the **Runge–Kutta methods** are a family of implicit and explicit iterative methods, which include the well-known routine called the Euler Method, used in temporal discretization for the approximate solutions of ordinary differential equations. These methods were developed around 1900 by the German mathematicians Carl Rungeand Wilhelm Kutta.

```
clc;
clear all;
%  fun=@(x,y)(-0.5*x^4 + 4*x^3 - 10*x^2 + 8.5*x +1)
%  Dy=@(x,y)(-2*x^3 + 12*x^2 - 20*x + 8.5);
%% Başlangıç koşulları
a=0;
b=2;
h=[0.05 0.025];
fun=@(x,y) ( exp ( (x^4) /4 -14*x) );
Dy=@(x,y)(y*x^3-14*y);
x=a:h:b; %x değeri
x_2=a:h/2:b; %x değeri midpoint için
step=length(x); %adım değeri normal euler için
step_2=length(x_2); % adım değeri midpoint
y(1)=1; % y başlangıç değeri
```

In my code, I began with a defining a function. After that, obtained the first derivative of the function because this value equal to the slope of the vector and that is the main character of our methods.

The **function handle is** used to make easier the calculations and to obtain the function handle. A function handle is a **MATLAB**® data type that stores an association to a function. Indirectly calling a function enables you to invoke the function regardless of where you call it from. For example, when I write down only x or y values into the bracket [Dy(x,y)], it makes calculation respect to the equation and comes out the result.

In addition, x values is assigned as two matrices. Because the question claims euler's solution for two different "h" value. "**h**" is equal to 0.05 and starts from 0 to 2 for first **"x"** matrix. "**h**" is equal to 0.025 and starts from 0 to 2 for second **"x"** matrix.

```
%% Euler Method
for i=1:step-1
    y(i+1)=y(i)+h(1)*Dy(x(i),y(i));
end
plot(x,y,'--bs','LineWidth',2);
hold on;
```

The first method is the **Euler's Method.** To build this method in MatLab, I benefited from **Steven Chappra Numerical Analysis Book** and I applied the formula on MatLab.

To plot the variables, I assigned two matices, x_plot and y_plot. These saves every **result** of y value.

```
%% Midpoint Method
for i=1:step-1
    y(i+1)=y(i)+ h(1)*Dy( x(i) + h(1)/2 , y(i) + (h(1)/2)* Dy(x(i),y(i)) );
end
plot(x,y,'-*','Color','g','LineWidth',2);
```

This method is solved similar to **Euler's Method.** Two matrices are assigned for plotting, results are saved as a matrix.

```
%% 4th Order Runge Kutta Method
for i=1:step-1
    k1=Dy(x(i),y(i));
    k2=Dy(x(i)+0.5*h(1),y(i)+0.5*k1*h(1));
    k3=Dy(x(i)+0.5*h(1),y(i)+0.5*k2*h(1));
    k4=Dy(x(i)+h(1),y(i)+k3*h(1));
    y(i+1)=y(i)+(1/6)*(k1+2*k2+2*k3+k4)*h(1);
end
plot(x,y,'--o','Color','c','LineWidth',2);
hold on;
```

4th order Runga Kutta method's needs too many calculations to obtain the exact value. I used **for loop** to calculate the **k(i)** values. There are for loop because I wanted to approximate the exact value with a **specific range** and a **specific accrual (h)**. End of the for loop, I obtained the **y(i+1) values** this calculation continues until complete every steps.

```
%% Analytic Solution
for i=1:step-1
    y(i)=fun(x(i),y(i));
end

plot(x,y,'Color','m','LineWidth',2);
hold on;
```

In analytic solution, I obtained the analytic result of equation and plotted as a graph.

```
%% Euler Method h=0.025
for i=1:step_2-1
    y(i+1)=y(i)+h(2)*Dy(x_2(i),y(i));
end
plot(x_2,y,'Color','y','LineWidth',2);
hold on
hold on;
grid on;
legend('Euler Method','4th RK method','Real Solution','Midpoint Method','Euler h=0.025');
```

The question claims two calculation for Euler's Method with different 'h' values. To make that, I used **two euler equations**. **'h'** is equal to 0.05 for the first calculation. For second calculation, **h** is 0.025 and two different graphs are plotted. "**plot"** command used to display the two graphs at the same time which are h is equal to 0.05(for euler and the another methods) and h is equal to 0.025(only for euler method, other methods is using 0.05).

At the bottom of every method I used **hold on** and **plot** command. Due to, every solution curve must be displayed at the same table.

Referances:

Steve Otto and James P Denier. An introduction to programming and numerical methods in MATLAB. Springer Science & Business Media, 2005.

mathworks.com