4 June 2019

NUMERICAL METHODS

ASSIGNMENT C
SOLVING ORDINARY DRIFFERENTIAL EQUATIONS

Michał Kurczak

285665

UNDER THE SUPERVISION OF ANDRZEJ MIĘKINA, PHD

Page od content

I. Formulation of the problem

A **differential equation** is a mathematical equation that relates some function with its derivatives. In applications, the functions usually represent physical quantities, the derivatives represent their rates of change, and the differential equation defines a relationship between the two. Because such relations are extremely common, differential equations play a prominent role in many disciplines including engineering, physics, economics, and biology. In my assignment, the **ordinary differential equation** (**ODE**) was to be solved.

$$9y'' + 6y' + 10y = 0 \text{ for } t \in [0,10],\ y(0) = 0 \text{ and } y'(0) = 2$$

Equation 1. Differential equation assigned to my project

The above equation was to be solved by means of three numerical algorithms:

- Own implementation of the **Lobatto IIID** order 4 method defined by the following Butcher table:

0	$\frac{1}{6}$	0	$-\frac{1}{6}$
$\frac{1}{2}$	$\frac{1}{12}$	$\frac{5}{12}$	0
1	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{6}$
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$

Table 1. Butcher table for Lobatto IIID

- MATLAB operator ***ode45***
- Own implementation of the **explicit Euler** method

Subsequently, the investigation of the dependence of the accuracy of the solutions on the integration step was to be carried out. For this purpose, the ***ode45*** solution was used as a point of reference, since we assumed this is the most accurate solution. As a measure of the accuracy, two indicators were taken for this purpose:

$$\delta_2(h) = \frac{\left\|\hat{\mathbf{y}}(t;h) - \dot{\mathbf{y}}(t,h)\right\|_2}{\left\|\dot{\mathbf{y}}(t,h)\right\|_2}$$

Equation 2. Root-mean-square error

$$\delta_\infty(h) = \frac{\left\|\hat{\mathbf{y}}(t;h) - \dot{\mathbf{y}}(t,h)\right\|_\infty}{\left\|\dot{\mathbf{y}}(t,h)\right\|_\infty}$$

Equation 3. Maximum error

II. Methodology

$$9y'' + 6y' + 10y = 0 \text{ for } t \in [0;10],$$

with the initial conditions

y(0)=0, y'(0)=2

In order so solve the above equation by means of **Lobatto IIID**, the following Butcher table was used:

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & 0 & -\frac{1}{6} \\ \frac{1}{2} & \frac{1}{12} & \frac{5}{12} & 0 \\ 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

The first step involves splitting one differential equation of the second order into two of the first order:

$$\begin{cases} 9y_1'' + 6y_1' + 10y_1 = 0 \\ y_1' = y_2 \end{cases}$$

after the substitution

$$\begin{cases} y_1' = y_2 \\ 9y_2' + 6y_2 + 10y_1 = 0 \end{cases}$$

Thus,

$$\begin{cases} y_1' = y_2 \\ y_2' = -\frac{10}{9}y_1 - \frac{2}{3}y_2 \end{cases}$$

Now, the matrix of coefficients can be obtained

$$\mathbf{A}=\begin{bmatrix} 0 & 1 \\ -\frac{10}{9} & -\frac{2}{3} \end{bmatrix}$$

According to the Butcher table

$$\begin{bmatrix} \boldsymbol{f_1} \\ \boldsymbol{f_2} \\ \boldsymbol{f_3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} \cdot [\boldsymbol{y_{n-1}} + h(\frac{1}{6}\boldsymbol{f_1} + 0 \cdot \boldsymbol{f_2} - \frac{1}{6}\boldsymbol{f_3})] \\ \boldsymbol{A} \cdot [\boldsymbol{y_{n-1}} + h(\frac{1}{12}\boldsymbol{f_1} + \frac{5}{12}\boldsymbol{f_2} + 0 \cdot \boldsymbol{f_3})] \\ \boldsymbol{A} \cdot [\boldsymbol{y_{n-1}} + h(\frac{1}{2}\boldsymbol{f_1} + \frac{1}{3}\boldsymbol{f_2} + \frac{1}{6}\boldsymbol{f_3})] \end{bmatrix}$$

hence

$$\begin{bmatrix} \boldsymbol{I} - \frac{1}{6}h\boldsymbol{A} & 0 & \frac{1}{6}h\boldsymbol{A} \\ -\frac{1}{12}h\boldsymbol{A} & \boldsymbol{I} - \frac{5}{12}hA & 0 \\ -\frac{1}{2}h\boldsymbol{A} & -\frac{1}{3}h\boldsymbol{A} & \boldsymbol{I} - \frac{1}{6}h\boldsymbol{A} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{f_1} \\ \boldsymbol{f_2} \\ \boldsymbol{f_3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} \cdot \boldsymbol{y_{n-1}} \\ \boldsymbol{A} \cdot \boldsymbol{y_{n-1}} \\ \boldsymbol{A} \cdot \boldsymbol{y_{n-1}} \end{bmatrix}$$

Performing left-hand side multiplication by the inverse matrix, we obtain vectors

$$\boldsymbol{y_n} = \boldsymbol{y_{n-1}} + h(\frac{1}{6}\boldsymbol{f_1} + \frac{2}{3}\boldsymbol{f_2} + \frac{1}{6}\boldsymbol{f_3})$$

of the form

$$\boldsymbol{y_n} = \begin{bmatrix} y_n \\ y_n{}' \end{bmatrix}$$

Explicit Euler method is a single step method described with the following equation

$$\boldsymbol{y_n} = \boldsymbol{y_{n-1}} + h \cdot f(\boldsymbol{t_{n-1}}, \boldsymbol{y_{n-1}})$$

where

$$f(\boldsymbol{t_{n-1}}, \boldsymbol{y_{n-1}}) = \boldsymbol{A} \cdot \boldsymbol{y_{n-1}}$$

hence

$$\boldsymbol{y_n} = \boldsymbol{y_{n-1}} + h \cdot \boldsymbol{A} \cdot \boldsymbol{y_{n-1}}$$

where

$$\boldsymbol{y_n} = \begin{bmatrix} y_n \\ y_n{}' \end{bmatrix}$$

III. Results

Figures 1-4 show the solutions to the ordinary differential equation assigned to my project.

$$9y'' + 6y' + 10y = 0 \text{ for } t \in [0,10],\ y(0) = 0 \text{ and } y'(0) = 2$$

Equation 4. Differential equation assigned to my project

Figures 1-3 obtained for the integration step h=0.01, Figure 4 obtained for the integration step h=0.05 in order to show the differences more clearly for the human eye.

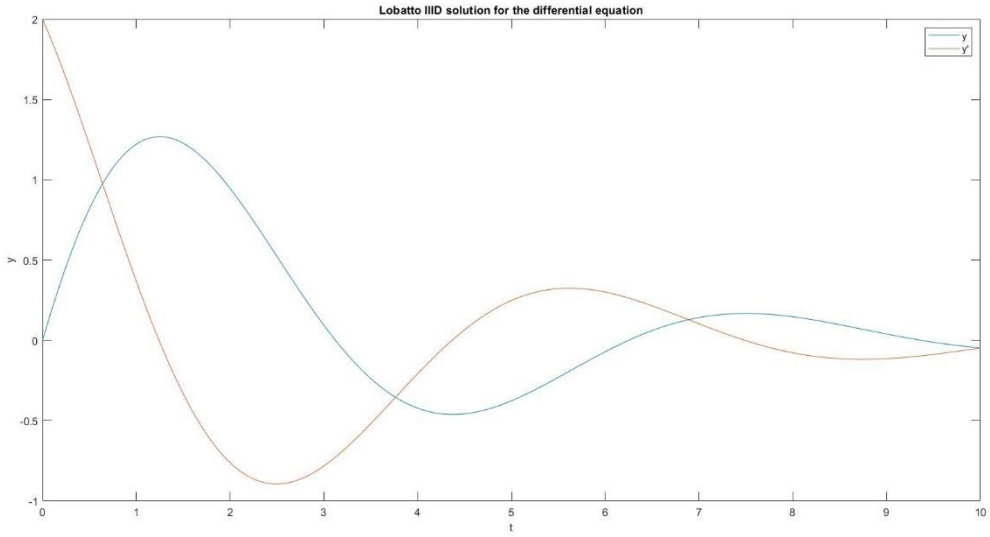

Figure 1. Lobatto IIID solution to the ODE

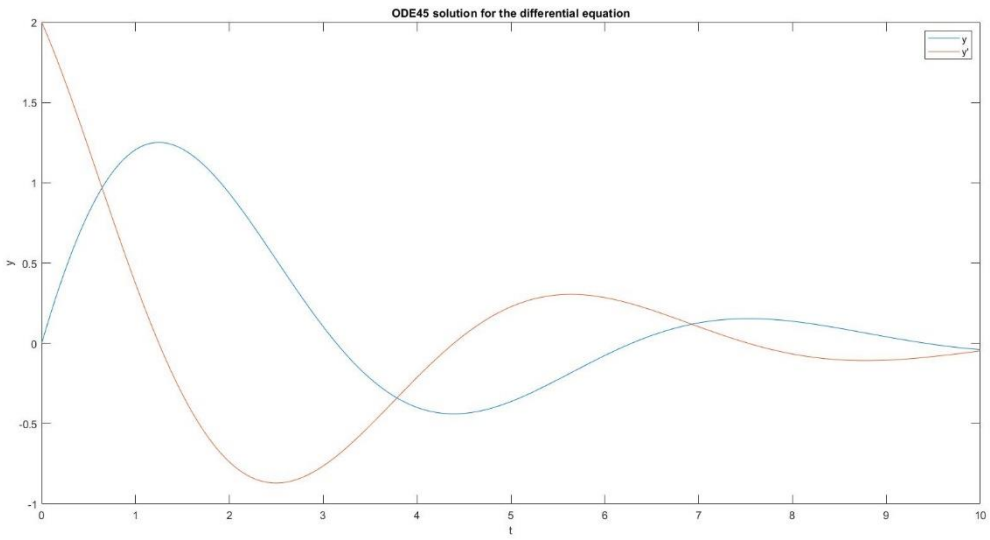

Figure 2. MATLAB's ode45 solution to the ODE

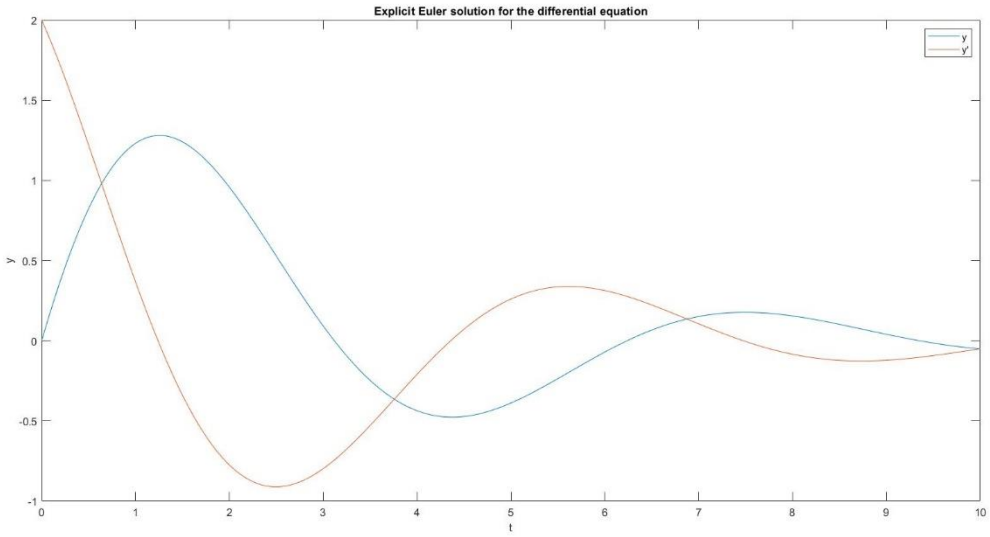

Figure 3. Explicit Euler solution to the ODE

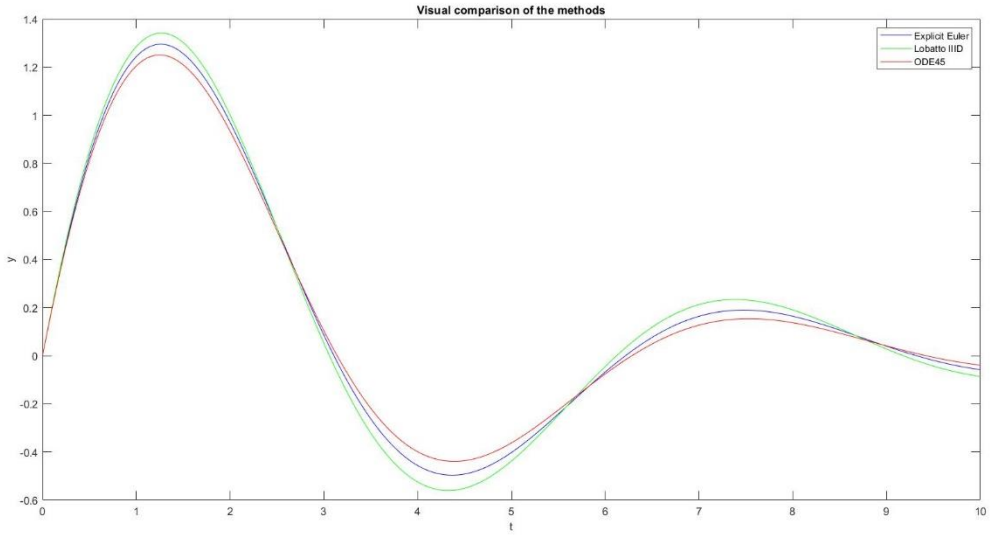

Figure 4. Visual comparison of the three methods. h=0.05

Figures 5-6 show the dependency of the errors on the step integration h value.

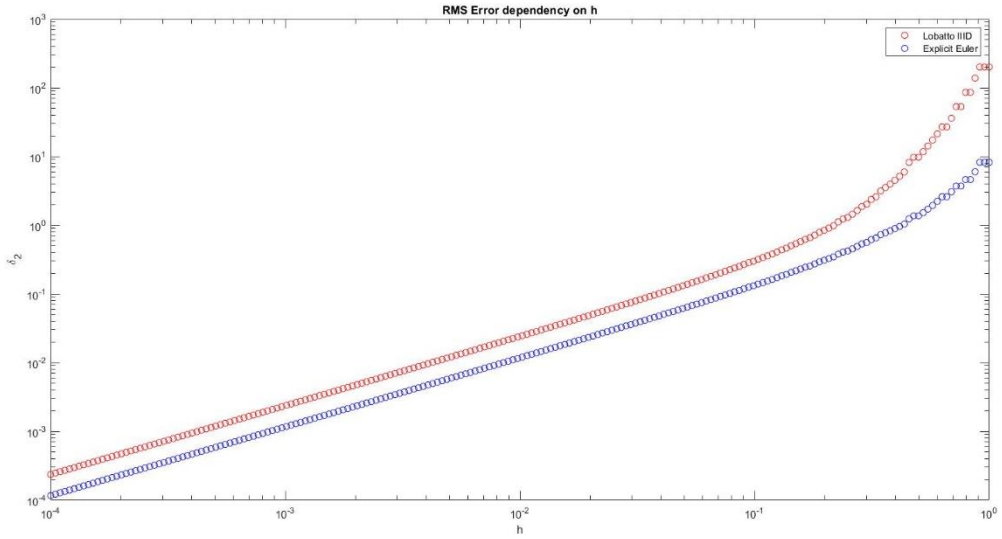

Figure 5. RMS dependency on the integration step value h

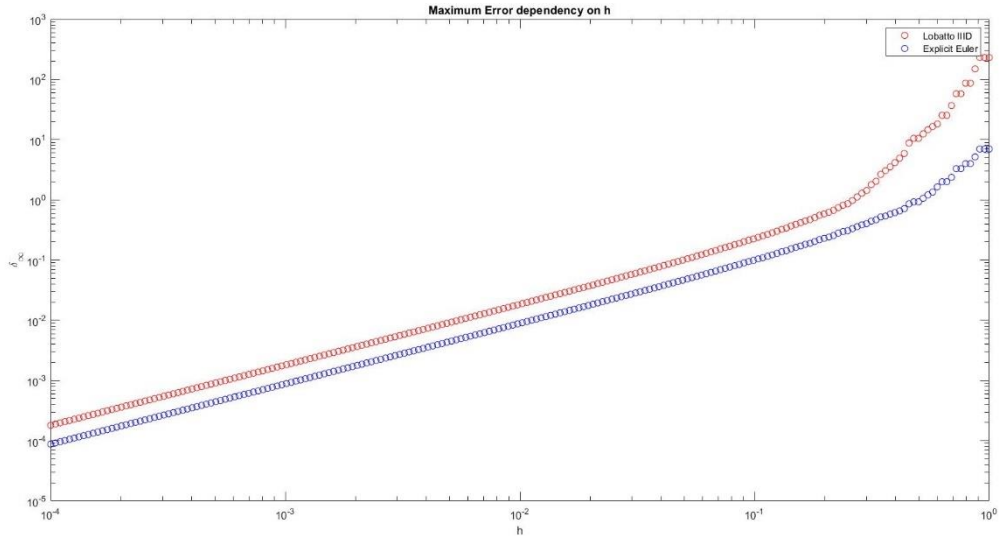

Figure 6. Maximum error dependency on the integration step value h

As we can see, the closer the integration step gets to the value of 0, the better the accuracy of both algorithms. Even simple observation of the figure comparing the methods can lead one into the conclusion that the explicit Euler method is more accurate on most of the interval. Those assumptions are confirmed with the investigation of the errors dependency on the step integration parameter h. Both the RMS and the Maximum Error values were smaller for the explicit Euler method.

IV. Conclusions

Differential equations are widely used for describing real-world phenomena in the engineering language. Finding an algorithm that solves them accurately and fast is the desired tool for many engineers.

Both methods of solving the ODE improved their accuracies with the integration step approaching zero value. Computation time strongly depends on the integration step, as the algorithm is a loop iterating through $\frac{10}{h} - 1$ loops. Being aware of that fact, one can assess that the computation time is inversely proportional to the integration step and find the consensus between the accuracy and the computation time.
What is more, better results were obtained for the explicit Euler method than the Lobatto IID method. Taking into consideration this, and the fact that the explicit Euler method requires fewer operations, one may conclude that the explicit Euler method is a better tool for solving ODEs.

V. Bibliography

[1] Roman Z. Morawski – ENUME 2019 – Lecture Notes

[2] https://en.wikipedia.org/wiki/Differential_equation

[3] https://en.wikipedia.org/wiki/Ordinary_differential_equation

VI. Appendix

```
clear all
close all
%t=[0,10]
t=linspace(0,10,1000); %h=10/length(t)



%task1

%Graph of the Lobatto IIID solution
lobatto_solution=lobatto_solved_ode(t);
figure()
plot(t, lobatto_solution);
xlabel('t');
ylabel('y');
title('Lobatto IIID solution for the differential
equation');
legend('y',"y'");

%Graph of the ode45 MatLab's function solution
ode45_solution=ode45_solved_ode(t);
figure()
plot(t,ode45_solution);
xlabel('t');
ylabel('y');
title('ODE45 solution for the differential equation');
legend('y',"y'");

%Graph of the Explicit Euler method solution
euler_solution=euler_solved_ode(t);
figure()
plot(t,euler_solution);
xlabel('t');
ylabel('y');
title('Explicit Euler solution for the differential
equation');
legend('y',"y'");

%Visual comparison of the methods
%Not required in the assignment, just for my own curiosity
figure()
plot(t, euler_solution(1,:),'b');
hold on
plot(t,lobatto_solution(1,:),'g');
```

```
figure()
plot(t, lobatto_solution);
xlabel('t');
ylabel('y');
title('Lobatto IIID solution for the differential
equation');
legend('y',"y'");

%Graph of the ode45 MatLab's function solution
ode45_solution=ode45_solved_ode(t);
figure()
plot(t,ode45_solution);
xlabel('t');
ylabel('y');
title('ODE45 solution for the differential equation');
legend('y',"y'");

%Graph of the Explicit Euler method solution
euler_solution=euler_solved_ode(t);
figure()
plot(t,euler_solution);
xlabel('t');
ylabel('y');
title('Explicit Euler solution for the differential
equation');
legend('y',"y'");

%Visual comparison of the methods
%Not required in the assignment, just for my own curiosity
figure()
plot(t, euler_solution(1,:),'b');
hold on
plot(t,lobatto_solution(1,:),'g');
hold on
plot(t,ode45_solution(:,1),'r');
xlabel('t');
ylabel('y');
title('Visual comparison of the methods');
legend('Explicit Euler','Lobatto IIID','ODE45');


%task2 & task3
num_of_points=200; %How many values of h
h=logspace(-4,0,num_of_points);
```

```
%data preallocation
rms=zeros(1,num_of_points);
rms_euler=zeros(1,num_of_points);
mx=zeros(1,num_of_points);
mx_euler=zeros(1,num_of_points);
%counting errors
for i=1:length(h)
rms(i)=RMS(h(i));
mx(i)=MxError(h(i));
rms_euler(i)=RMS_euler(h(i));
mx_euler(i)=MxError_euler(h(i));
end

%RMS dependency on h
figure()
loglog(h,rms,'or');
hold on
loglog(h,rms_euler,'ob');
xlabel('h');
ylabel('\delta_2');
title('RMS Error dependency on h');
legend('Lobatto IIID', 'Explicit Euler');

%Maximum error dependency on h
figure()
loglog(h,mx,'or');
hold on
loglog(h,mx_euler,'ob');
xlabel('h');
ylabel('\delta_\infty');
title('Maximum Error dependency on h');
legend('Lobatto IIID', 'Explicit Euler');


%----functions-----

%Just for checking corectness of the solution
%at the very beginning of writing the code
%Not used in tasks completion
function y=manually_solved_ode(t)
 y=zeros(1,length(t));
 for i=1:(length(t))
   y(i)=2*exp(-t(i)/3)*sin(t(i));
 end
```

```
end

%Solving differential equation using ode45 MatLab function
function y=ode45_solved_ode(t)
    %9y''+6y'+10y=0
    y0=0;
    dy0dt=2;


    opts = odeset('RelTol',1e-13,'AbsTol',1e-13);
    [t,y]=ode45( @rhs, t, [y0, dy0dt], opts);

    function dydt=rhs(t,y)
        dydt = [y(2); -(6*y(2) + 10*y(1))/9];
    end
end

%Solving differential equation using explicit Euler method
function y=euler_solved_ode(t)
y=zeros(2,length(t));
y(1,1)=0; %y0
y(2,1)=2; %dy0dt
h=10/length(t);   %step
A=[0,1;-10/9,-2/3];

for i=2:length(t)
    y(:,i)=y(:,i-1)+h.*A*y(:,i-1);
end
end

%Solving differential equation using LobattoIIID method
function y=lobatto_solved_ode(t)
F=zeros(6,length(t));
y=zeros(2,length(t));
y(1,1)=0; %y0
y(2,1)=2; %dy0dt
h=10/length(t);   %step

A=[0,1;-10/9,-2/3];
L=[eye(2)-1/6*h*A,zeros(2),1/6*h*A;
   -1/12*h*A, eye(2)-5/12*h*A, zeros(2);
   -1/2*h*A, -1/3*h*A, eye(2)-1/6*h*A];
for i=2:length(t)
    R=[A*y(:,i-1);A*y(:,i-1);A*y(:,i-1)];
```

```
F(:,i-1)=1\L*R;
    f1=F(1:2,i-1);
    f2=F(3:4,i-1);
    f3=F(5:6,i-1);
    y(:,i)=y(:,i-1)+h*(1/6*f1+2/3*f2+1/6*f3);
end
end

%---------Errors calculation----

%Root mean square error
%Ode45 solution as the accurate solution
%Lobatto IIID as the approximated solution
function y=RMS(h)
num_of_pts=10/h;
t=linspace(0,10,num_of_pts);
y_lobatto=lobatto_solved_ode(t).';
y_ode45=ode45_solved_ode(t);
    nominator=norm(y_lobatto(:,1)-y_ode45(:,1),2);
    denominator=norm(y_ode45(:,1),2);
    y=nominator/denominator;
end

%Maximum error
%Ode45 solution as the accurate solution
%Lobatto IIID as the approximated solution
function y=MxError(h)
num_of_pts=10/h;
t=linspace(0,10,num_of_pts);
y_lobatto=lobatto_solved_ode(t).';
y_ode45=ode45_solved_ode(t);
    nominator=norm(y_lobatto(:,1)-y_ode45(:,1),inf);
    denominator=norm(y_ode45(:,1),inf);
    y=nominator/denominator;
end

%----Explicit Euler investigation---

%Root mean square error
%Ode45 solution as the accurate solution
%Explicit Euler method as the approximated solution
function y=RMS_euler(h)
num_of_pts=10/h;
t=linspace(0,10,num_of_pts);
```

```
y_euler=euler_solved_ode(t).';
y_ode45=ode45_solved_ode(t);
    nominator=norm(y_euler(:,1)-y_ode45(:,1),2);
    denominator=norm(y_ode45(:,1),2);
    y=nominator/denominator;
end

%Maximum error
%Ode45 solution as the accurate solution
%Explicit Euler method as the approximated solution
function y=MxError_euler(h)
num_of_pts=10/h;
t=linspace(0,10,num_of_pts);
y_euler=euler_solved_ode(t).';
y_ode45=ode45_solved_ode(t);
    nominator=norm(y_euler(:,1)-y_ode45(:,1),inf);
    denominator=norm(y_ode45(:,1),inf);
    y=nominator/denominator;
end
```