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:

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

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,

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

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

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

Equation 4. Differential equation assigned to my project

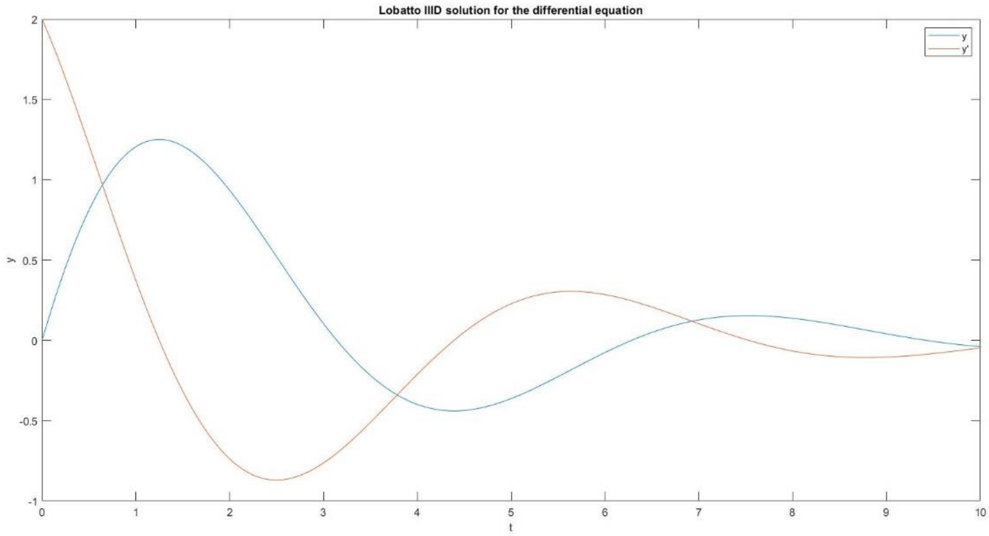


Figure 1. Lobatto solution to the ODE

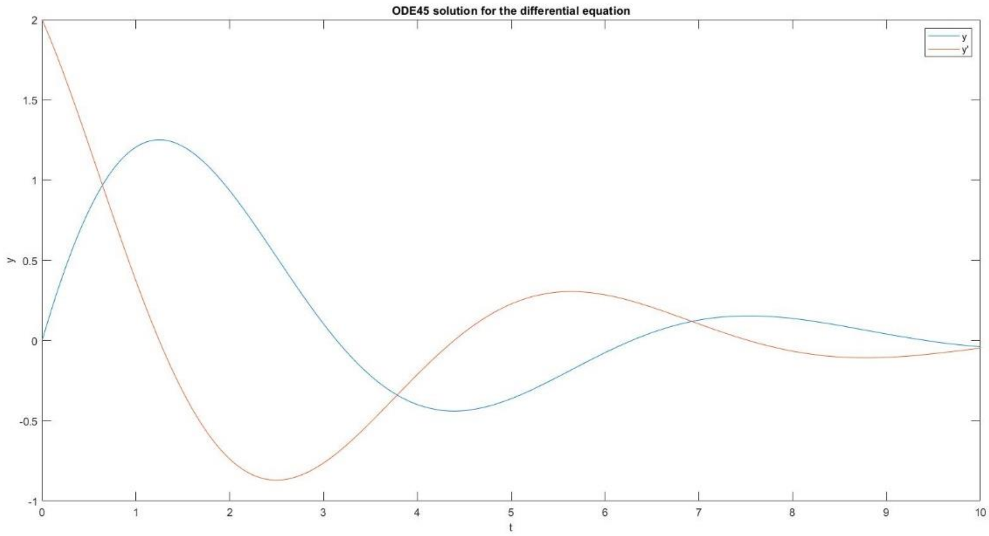


Figure 2. MATLAB ode45 solution to the ODE

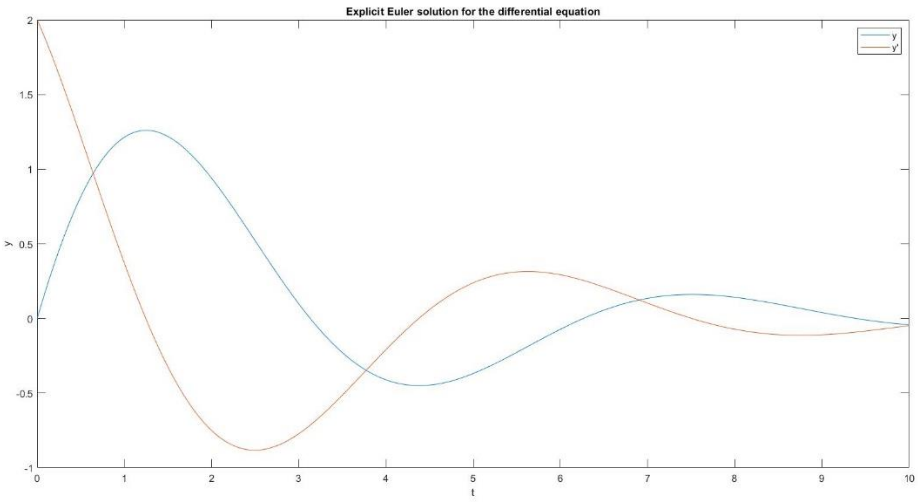


Figure 3. Explicit Euler solution to the ODE

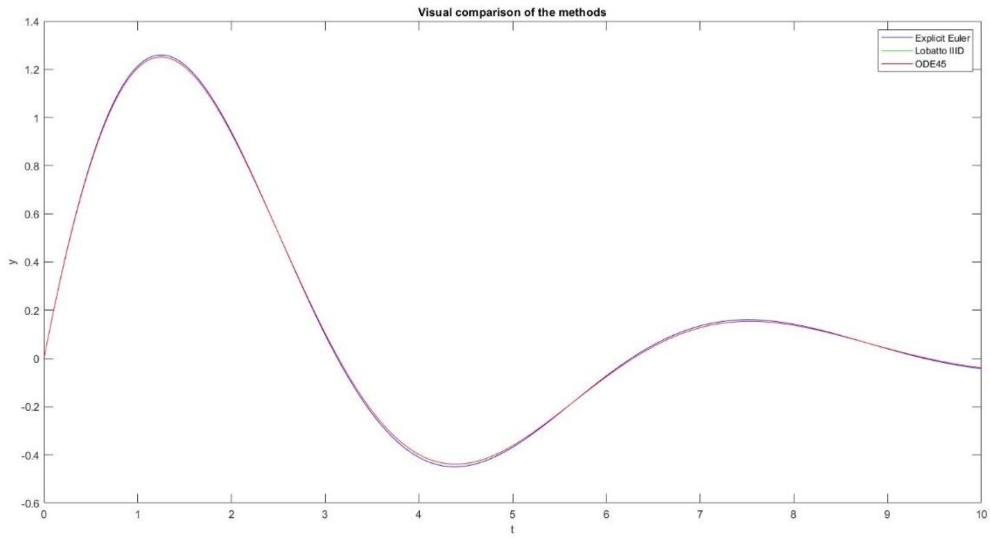


Figure 4. Visual comparison of the explicit Euler, Lobatto IIID and MATLAB ode45 methods

At the first glance, there are some differences between methods. Since the human eye is not the best quality of algorithm indicator, further numerical analysis is required.

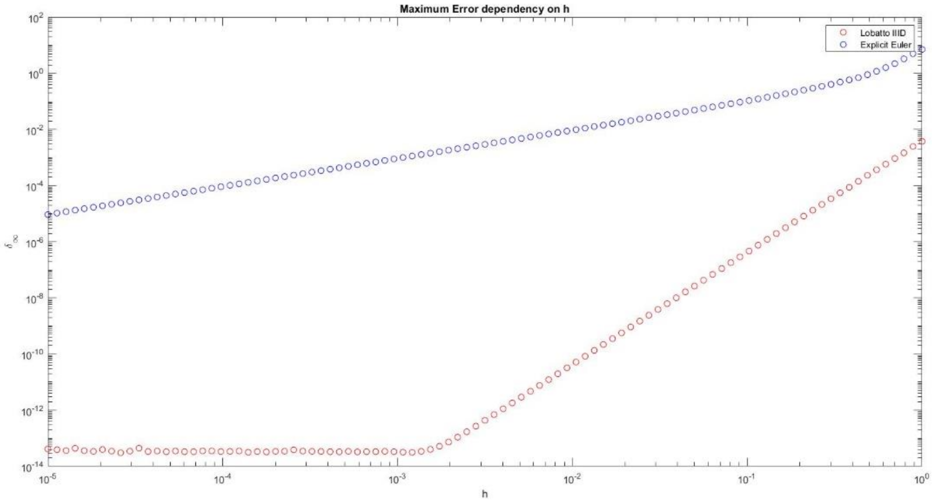


Figure 5. Maximum error dependency on the integration step h

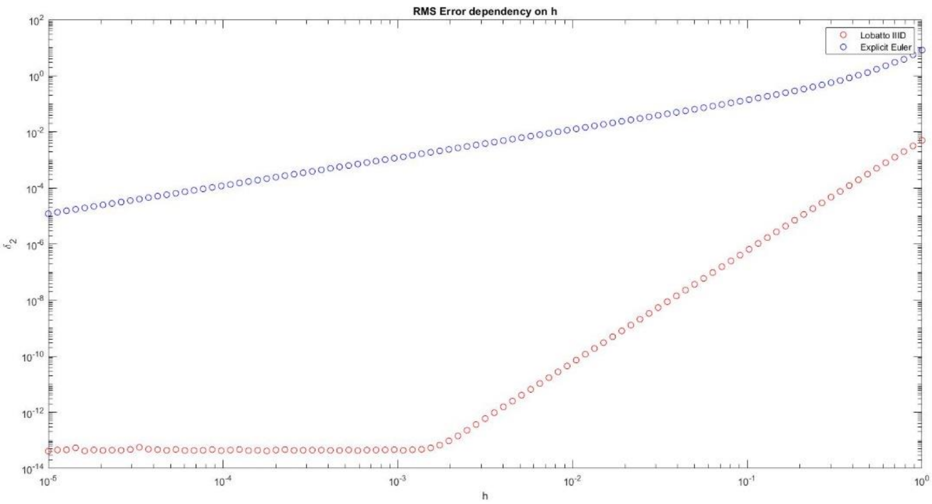
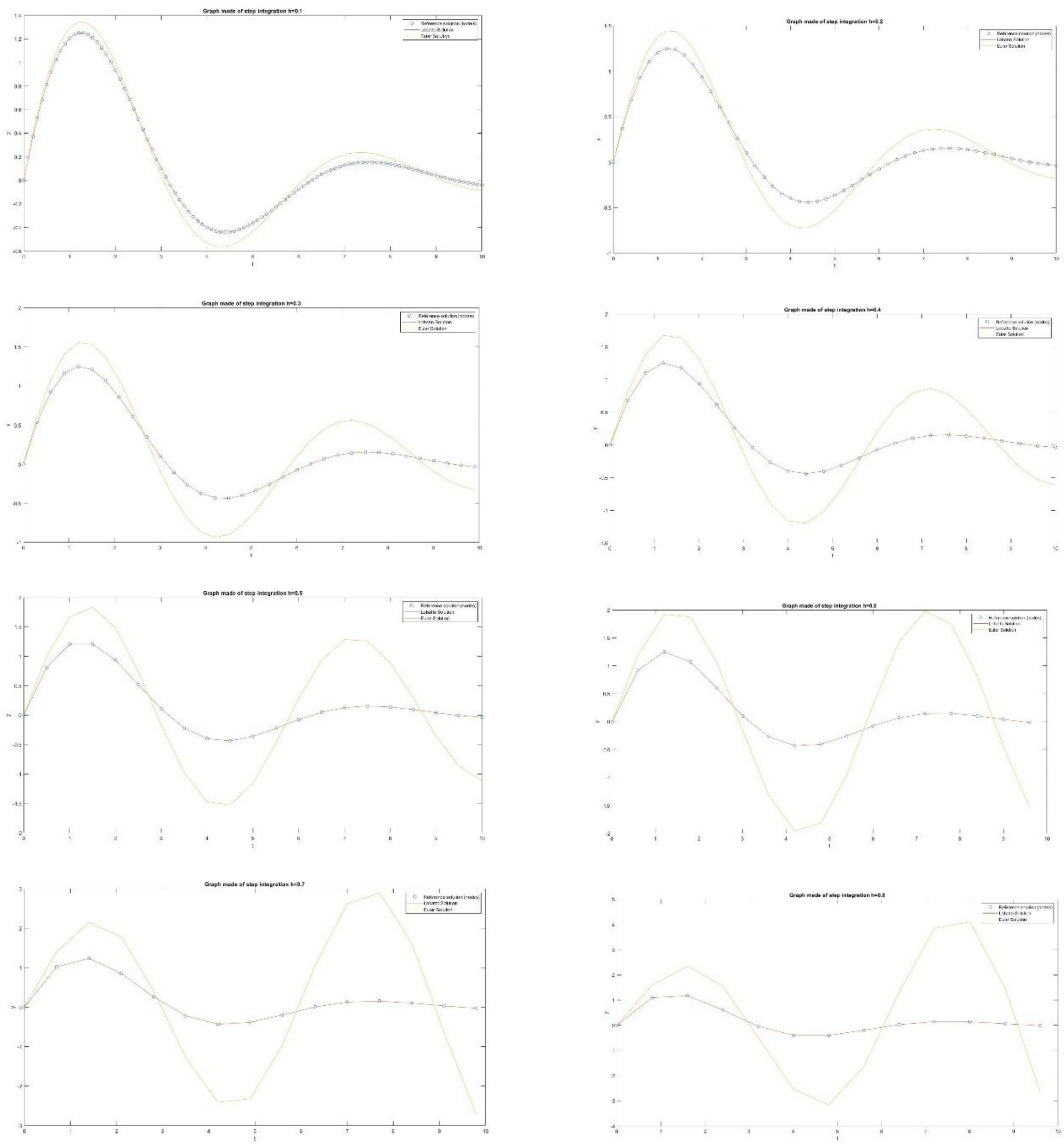


Figure 6. Root-mean-square error dependency on the integration step h

As we can see, Lobatto IIID method gives better results on the whole interval. One can observe the improvement of the Lobatto algorithm up to the vicinity of the integration step equal to ~10^{-3}. Approaching value of zero after that point gives us no observable improvement of the algorithm. Explicit Euler algorithm on the other hand, improves with the decreasing of the integration step, yet it is significantly less precise than the Lobatto IID algorithm.



Figures 7-14. Graphs of Lobatto and Euler solutions made for different values of the integration step parameter

As we can see, for relatively big values of the integration step Lobatto IIID method gives us fairly good results, whereas the explicit Euler algorithm solutions are far from the real ones.

```
Command Window

LobattoTime =

   28.5914


EulerTime =

   11.1426
```

Figure 15. Time of calculating RMS and maximum error for all of the values of h

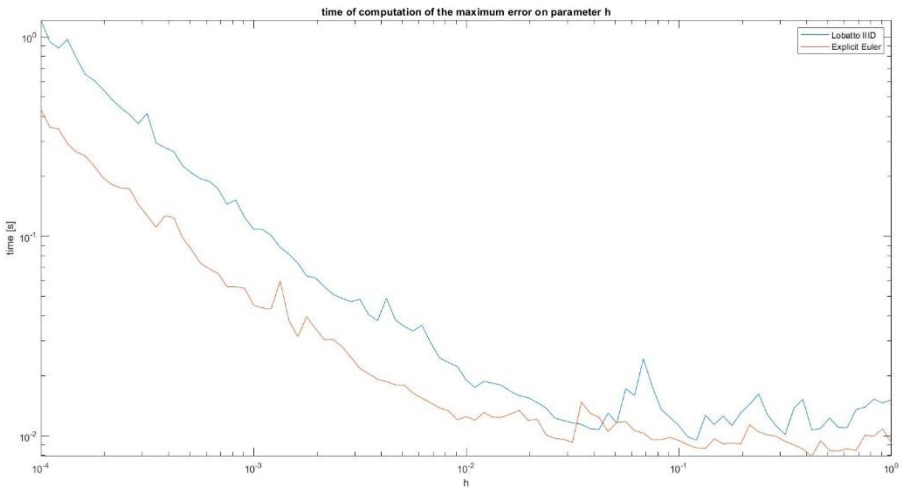


Figure 16. time of computation of the maximum error on parameter h

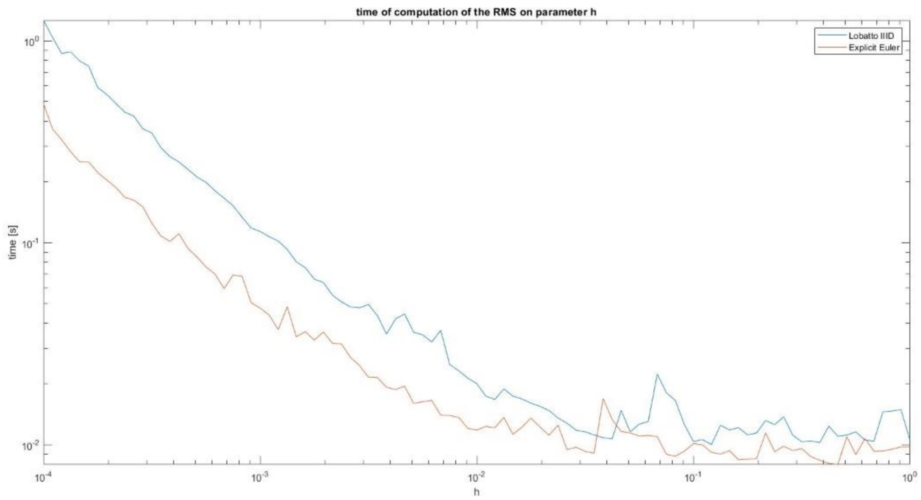


Figure 17. Time of computation of the RMS on parameter h

Since the computation speed is also a very important parameter of the numerical algorithm, I decided to compare the times of calculating errors for both Lobatto and Euler algorithms. As we can see, computations for the Euler algorithm required less time for most of the values of the integration step parameter.

IV. Conclusions

Although the explicit Euler method is a very simple to both understand and implement method, it is much less accurate method than the Lobatto IID. With the use of Lobatto IIID, we are able to obtain very accurate answers (errors of the order of 10^{-14}).

What is more, for the fairly big steps of integration, the explicit Euler method is of no use, whereas Lobatto IIID handles the problems fairly well.

Yet, we have to point out that Lobatto IIID is a more complex algorithm, hence it takes more time to obtain the answer. It can be easily observed in the computation time of errors. The Lobatto IIID required more than twice as much time than the Euler algorithm for the completion of all the loops, yet one has to be aware that this is just a simple analysis showing that there is a difference, but it is not sufficient to conclude any further information.

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]
h=0.01;
t=0:h:10;



%task1

%Graph of the Lobatto IIID solution
lobatto_solution=lobatto_solved_ode(h);
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(h);
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(h);
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=97; %How many values of h
(3*N+4)
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
tic
for i=1:length(h)
tic
rms(i)=RMS(h(i));
rms_time(i)=toc;
tic
mx(i)=MxError(h(i));
mx_time(i)=toc;
end
LobattoTime=toc
tic
for i=1:length(h)
tic
rms_euler(i)=RMS_euler(h(i));
rms_euler_time(i)=toc;
tic;
mx_euler(i)=MxError_euler(h(i));
mx_euler_time(i)=toc;
end
EulerTime=toc

%rms times
figure()
loglog(h,rms_time);
hold on
plot(h,rms_euler_time);
xlabel('h');
ylabel('time [s]');
title('time of computation of the RMS on
parameter h');
legend('Lobatto IIID', 'Explicit
Euler');

%maximum error times
figure()
loglog(h,mx_time);
hold on
plot(h,mx_euler_time);
xlabel('h');
ylabel('time [s]');
title('time of computation of the
maximum error on parameter h');
legend('Lobatto IIID', 'Explicit
Euler');



%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');


%Graphs investigation on h
h=0.1:0.1:1;
for i=1:length(h)
t=0:h(i):10;
manual_solution=manually_solved_ode(t);
euler_solution=euler_solved_ode(h(i));
lobatto_solution=lobatto_solved_ode(h(i)
);
figure()
plot(t,manual_solution,'o');
hold on
plot(t,lobatto_solution(1,:));
hold on
plot(t,euler_solution(1,:));
xlabel('t');
ylabel('y');
ttle=strcat('Graph made of step
integration h=', num2str(h(i)));
title(ttle);
legend('Reference solution (nodes)',
"Lobatto Solution","Euler Solution");
end


%----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(h)
    %9y''+6y'+10y=0
    y0=0;
    dy0dt=2;
    t=0:h:10;

    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(h)
 t=0:h:10;
y=zeros(2,length(t));
y(1,1)=0; %y0
y(2,1)=2; %dy0dt
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(h)
 t=0:h:10;
y=zeros(2,length(t));
y(1,1)=0; %y0
y(2,1)=2; %dy0dt

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=L\R;
    f1=F(1:2);
    f2=F(3:4);
    f3=F(5:6);
    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)
y_lobatto=lobatto_solved_ode(h).';
y_ode45=ode45_solved_ode(h);
    nominator=norm(y_lobatto(:,1)-
y_ode45(:,1),2);
    denominator=norm(y_ode45(:,1),2);
    y=nominator/denominator;
end

%Maximum error
```

```matlab
%Ode45 solution as the accurate solution
%Lobatto IIID as the approximated solution
function y=MxError(h)
y_lobatto=lobatto_solved_ode(h).';
y_ode45=ode45_solved_ode(h);
    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)
y_euler=euler_solved_ode(h).';
y_ode45=ode45_solved_ode(h);
    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)
y_euler=euler_solved_ode(h).';
y_ode45=ode45_solved_ode(h);
    nominator=norm(y_euler(:,1)-y_ode45(:,1),inf);
    denominator=norm(y_ode45(:,1),inf);
    y=nominator/denominator;
end
```