PROJECT ASSIGNMENT C: SOLVING ORDINARY DIFFERENTIAL EQUATIONS

Author: Minh Hieu Do (ID: 288414)

Course: Numerical Methods (ENUME)

Advisor: Andrzej Miekina, Ph.D., Assistant Professor

Warsaw, 12/06/2018

Contents

1. The concise description of numerical algorithms

a. *Butcher's tableau of coefficients*

$$
\begin{array}{c|cccc}
c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,K} \\
c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,K} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
c_K & a_{K,1} & a_{K,2} & \cdots & a_{K,K} \\
\hline
 & w_1 & w_2 & \cdots & w_K
\end{array}
$$

b. *Implicit Runge-Kutta formula using Butcher's tableau of coefficients*

ODE systems:

$$y'(t) = A \cdot y(t) + b \cdot e(t) \quad for\ t \in [0, T]$$

may be solve by using the following algorithm:

$$y_n = y_{n-1} + h \sum_{k=1}^{K} w_k f_k$$

Where:

$$h = t_n - t_{n-1}$$

$$
\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_K \end{bmatrix} =
\begin{bmatrix}
f\left(t_{n-1} + c_1 h, y_{n-1} + h \sum_{k=1}^{K} a_{1,k} f_k\right) \\
f\left(t_{n-1} + c_2 h, y_{n-1} + h \sum_{k=1}^{K} a_{2,k} f_k\right) \\
\vdots \\
f\left(t_{n-1} + c_K h, y_{n-1} + h \sum_{k=1}^{K} a_{K,k} f_k\right)
\end{bmatrix}
$$

c. *Explicit Euler method*

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

d. *Accuracy indictors*

$$\delta_2(h) = \frac{\|\hat{y}(t;h) - \dot{y}(t,h)\|_2}{\|\dot{y}(t,h)\|_2} \ and\ \delta_\infty(h) = \frac{\|\hat{y}(t;h) - \dot{y}(t,h)\|_\infty}{\|\dot{y}(t,h)\|_\infty}$$

with $\dot{y}(t,h)$ is the most accurate solution and $\hat{y}(t;h)$ is the numerical solution obtained of ODE equation for the integration step h.

2. The methodology for testing numerical algorithms

- Design a MATLAB procedure to solve the equation

$$y' = -\mathrm{y} + 2\mathrm{t}e^{-t+2} \; for \; t \in [0, T] \; and \; y(0) = 0$$

by applying the implicit method Lobatto IIID of order 2.
- Make the graphs of the obtained results of the above equation using method Lobatto IIID, explicit Euler method and MATLAB function ***ode45*** for some integration step *h*.
- For the MATLAB function ***ode45***, repeat several times to obtain the values of its parameters *AbsTol and RelTol* to maximize the exactness of the solution, which are the smallest values they can be.
- Perform the dependence of the accuracy indicators $\delta_2(h)$ and $\delta_\infty(h)$ of Lobatto IIID method and explicit Euler method on the integration step *h*.
- Plot graphs of the dependence of $\delta_2(h)$ and $\delta_\infty(h)$ of both methods.

3. The results of testing numerical algorithms

a. Problem 1, 3

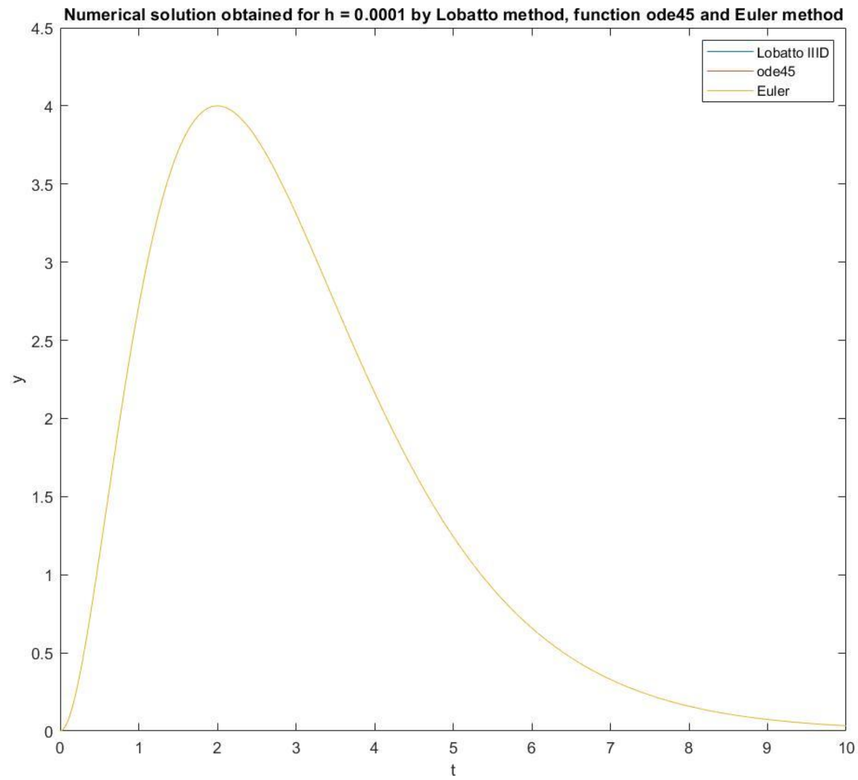

*Figure 1. Numerical solution obtained for **h = 0.0001***

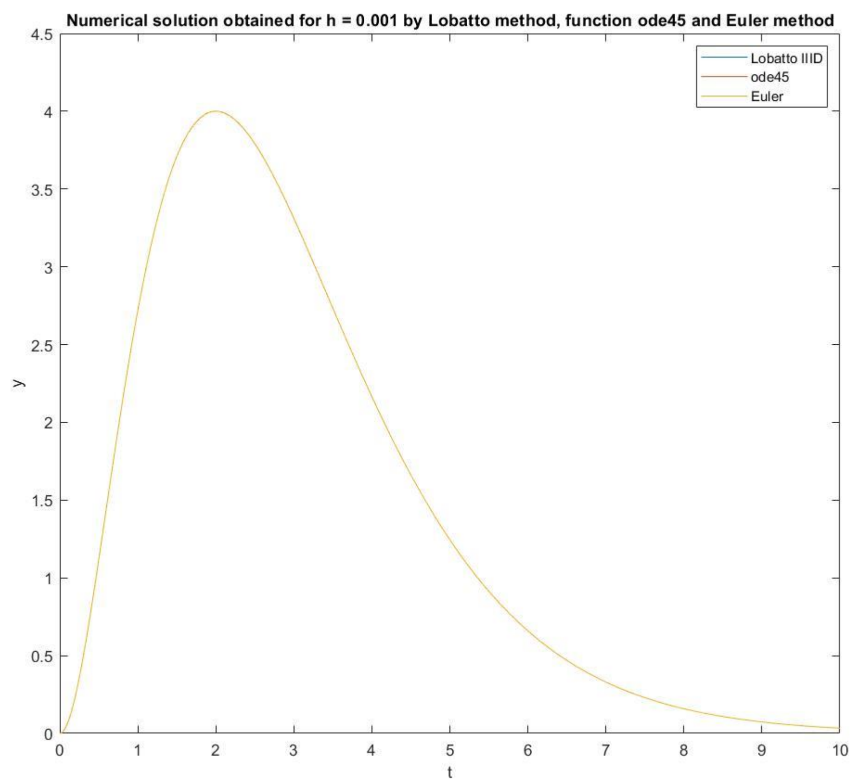

*Figure 2. Numerical solution obtained for **h = 0.001***

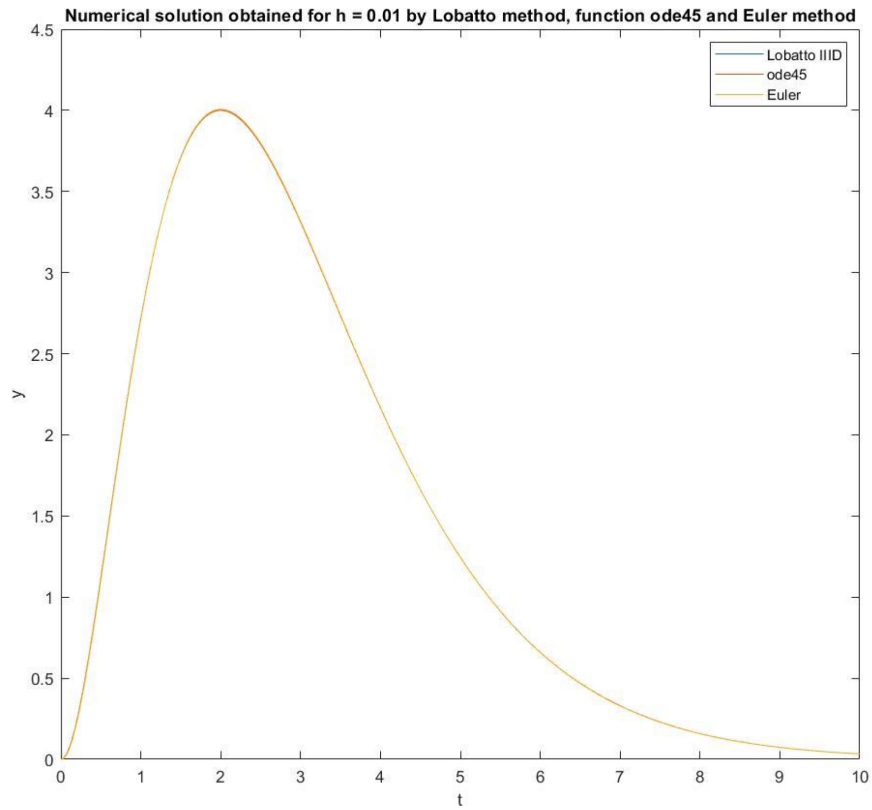

*Figure 3. Numerical solution obtained for **h = 0.01***

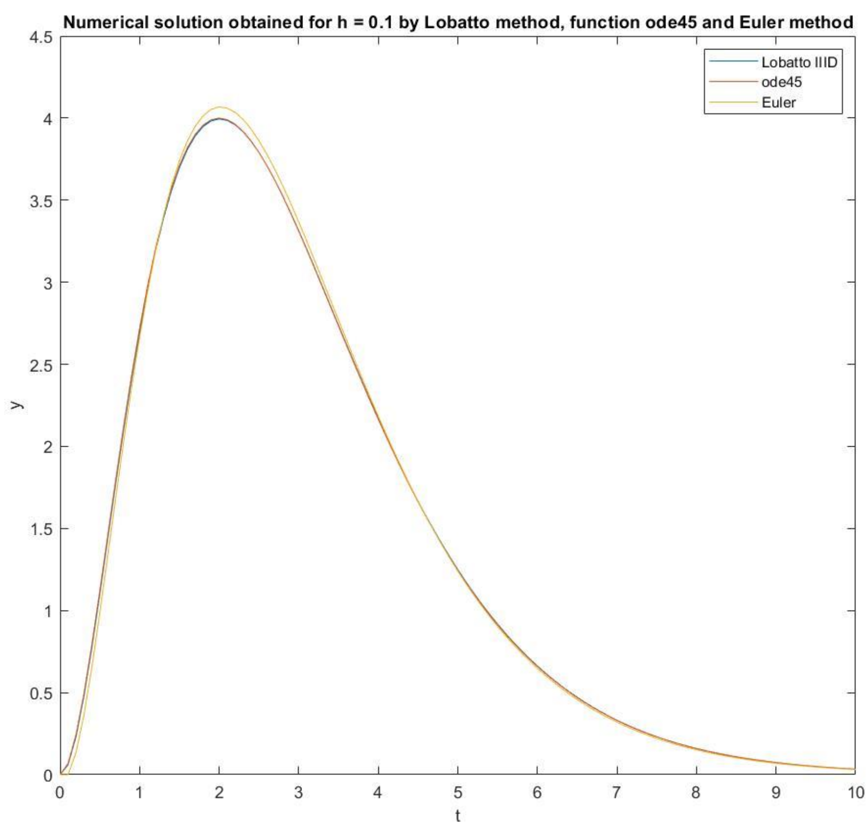

*Figure 4. Numerical solution obtained for **h = 0.1***

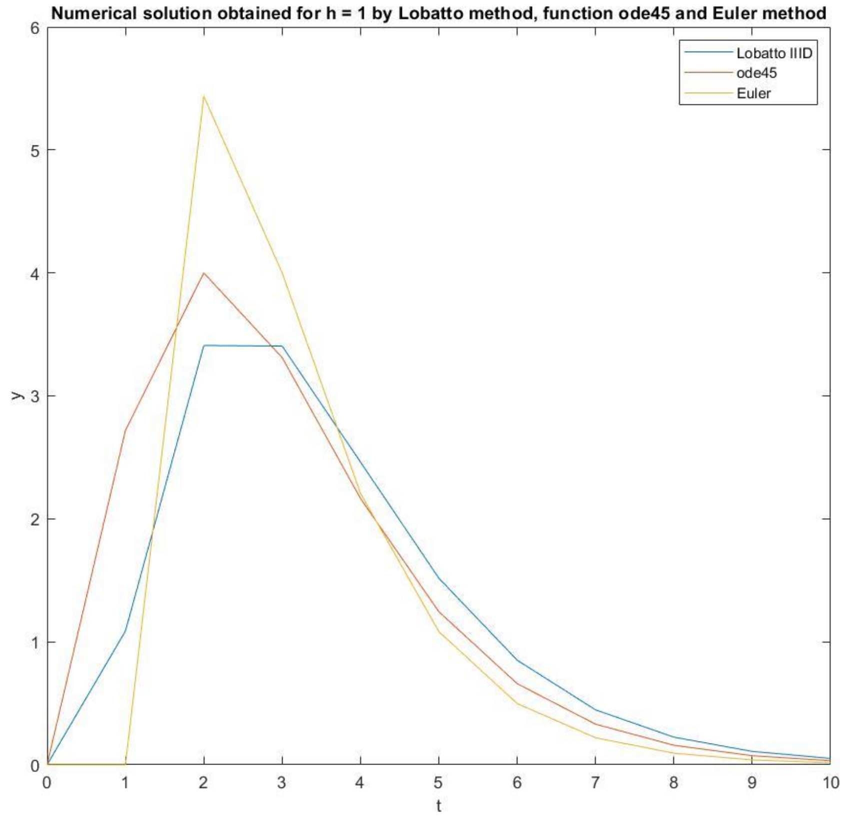

*Figure 3. Numerical solution obtained for **h = 1***

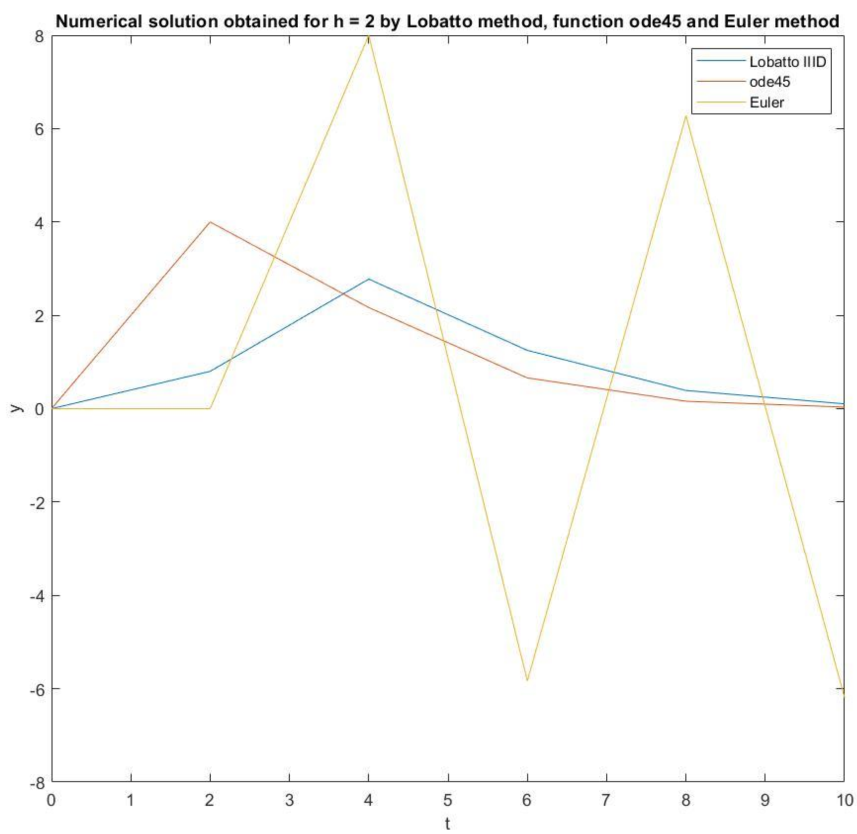

*Figure 4. Numerical solution obtained for **h = 2***

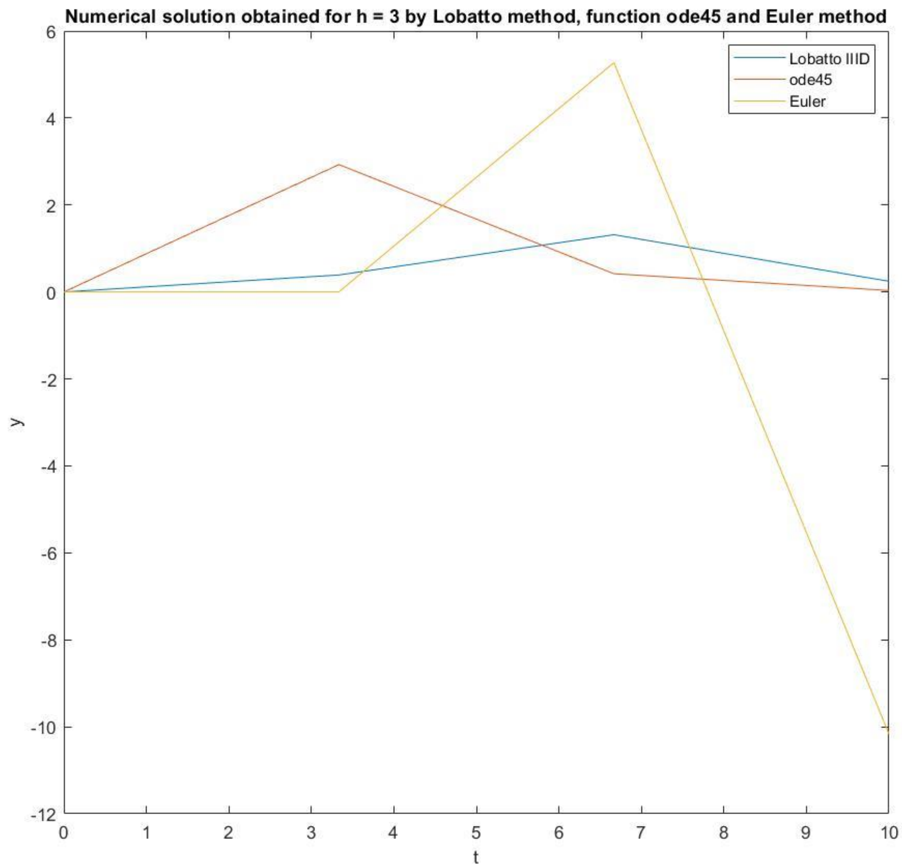

*Figure 5. Numerical solution obtained for **h = 3***

b. Problem 2, 3

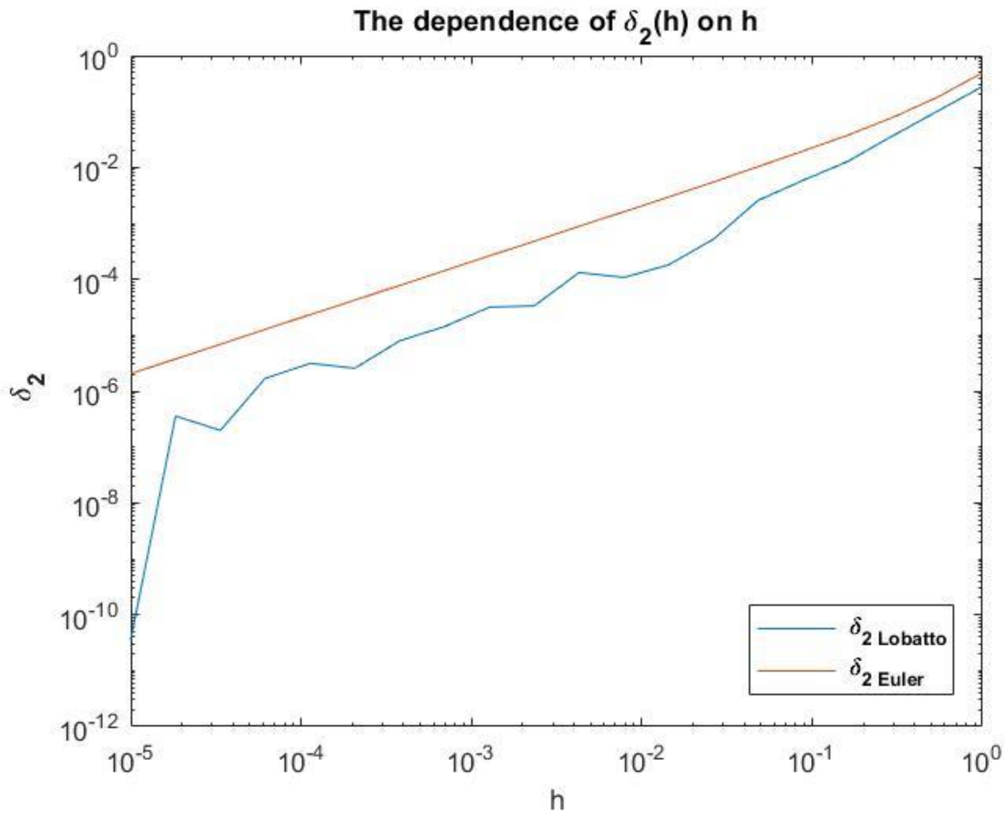

Figure 6. The dependence of $\delta_2(h)$ on h

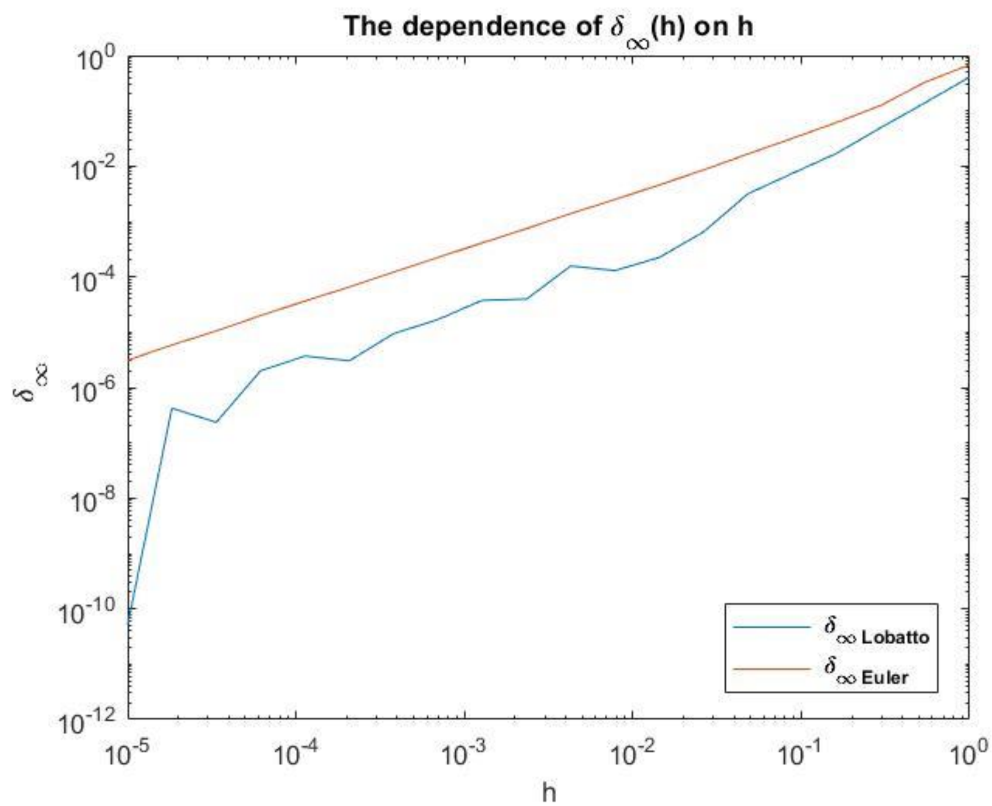

Figure 7. The dependence of $\delta_\infty(h)$ on h

4. Conclusion

- After using the implicit method Lobatto IIID of order 2, the explicit Euler method and the MATLAB function ***ode45*** to solve ordinary differential equation and analyzing the graphs of the dependence of $\delta_2(h)$ and $\delta_\infty(h)$ on h, it can be seen that:
    - The obtained results are very close to each other for the small integration step h $(h \leq 0.01)$.
    - The smaller the integration step is, the more accurate the obtained results of both Lobato IIID and Euler method are.
    - The obtained solutions of the implicit method Lobatto IIID of order 2 are more precise than the one of the explicit Euler method.
- To maximize the accuracy of the solution of the MATLAB function ***ode45***, $2.22045e^{-14}$ and e^{-20} were chosen as the values of its parameters *RelTol* and *AbsTol* respectively.

List of references

R. Z. Morawski, Lecture notes for ENUME students

A. Miękina, ENUME MatLab Intro 2018

MathWorks, MATLAB Documentation, https://www.mathworks.com/help/index.html

MATLAB Code

```
clear all
close all

%%Problem 1
H = [0.0001, 0.001, 0.01, 0.1, 1, 2, 3];
hl = length(H);
for i = 1 : hl
    [y, yy, T] = Lobatto(H(i));
    yE = Euler(H(i));
    figure(i)
    plot (T, y, T, yy, T, yE)
    xlabel('t')
    ylabel('y')
    legend('Lobatto IIID', 'ode45', 'Euler')
    title(['Numerical solution obtained for h = 
',num2str(H(i)),' by Lobatto method, function ode45 and Euler 
method']);
end

%Problem 2
H = logspace(-5, 0, 20);
hl = length(H);

accuracy = zeros(size(H));
accuracyInf = zeros(size(H));
accuracyEuler = zeros(size(H));
accuracyInfEuler = zeros(size(H));

for i = 1 : hl
    h = H(i);
    [y, yy, T] = Lobatto(h);
    accuracy(i) = norm(y - yy) / norm(yy);
    accuracyInf(i) = norm(y - yy, Inf) / norm(yy, Inf);

    yEuler = Euler(h);
    accuracyEuler(i) = norm(yEuler - yy) / norm(yy);
    accuracyInfEuler(i) = norm(yEuler - yy, Inf) / norm(yy, 
Inf);

end

figure(8)
loglog(H, accuracy, H, accuracyEuler);
xlabel('h');
ylabel('\bf \delta_{2}');
legend('\bf \delta_{2 Lobatto}', '\bf \delta_{2 Euler}');
title(['The dependence of \delta_{2}(h) on h']);
```

```
figure(9)
loglog(H, accuracyInf, H, accuracyInfEuler);
xlabel('h');
ylabel('\bf \delta_{\infty}');
legend('\bf \delta_{\infty Lobatto}', '\bf \delta_{\infty
Euler}');
title(['The dependence of \delta_{\infty}(h) on h']);


%Solving using the implicit method Lobatto IID of order 2
function [y, yy, T] = Lobatto(h)
    T = linspace (0, 10, 10/h + 1);
    y = zeros(size(T));

    a = 1/2*h;
    A = [1 + a, a;
        -a, 1 + a];

    B = zeros(2, 1);
    F = zeros(2, 1);

    N = length(y);

    for n = 2 : N
        B = [-y(n-1) + 2 * T(n-1) * exp(-T(n-1) + 2);
            -y(n-1) + 2 * (T(n-1) + h) * exp(-T(n-1) - h + 2)];
        F = A\B;
        y(n) = y(n-1) + h * (1/2*F(1) + 1/2*F(2));
    end
    y = y';

    opts = odeset('RelTol',2.22045e-14,'AbsTol',1e-20);
    yy0 = 0;
    [~, yy] = ode45(@(a, x) -x + 2*a*exp(-a+2), T, yy0, opts);
end


%Solving using the explicit Euler method
function [y] = Euler(h)
    T = linspace (0, 10, 10/h + 1);
    y = zeros(size(T));
    N = length(y);

    for n = 2 : N
        y(n) = y(n-1) + h * (-y(n-1) + 2 * T(n-1) * exp(-T(n-1)
+ 2));
    end
    y = y';
end
```