Analysis of Euler's and Modified Euler's Method
to Solve Initial Value Problems and Accelerate Convergence

By: Adam Headley 17 April 2015

Introduction

This program will solve any initial value problem

$$y' = f(t, y), \quad y(t_0) = y_0,$$

using Euler's Method (**em.M**) and Modified Euler's Method (**mem.M**). It is sufficiently general so that it can be used for an arbitrary function $f(t, y)$, with stepsize h and interval $[a, b]$. Both the code and commented explanations for these functions and the script used can be found in the appendix.

A

Using these functions the following initial value problem was solved:

$$y' = -4t^3y^2; \quad y(-a) = \frac{1}{(a^4+1)}; \quad t \in [-a, a]$$

The value of $a = 7$ was fixed for the analysis of this program. This value was chosen to observe the behavior of the methods.

To verify that $y(t) = \frac{1}{(t^4+1)}$ is the exact solution to the initial value problem, the derivative of the exact solution will be taken and then both $y(t)$ and $y'(t)$ will be plugged-in to their corresponding variables in the IVP and checked for equality. The computational knowledge engine, WolframAlpha, computed the derivative of the exact solution to be:

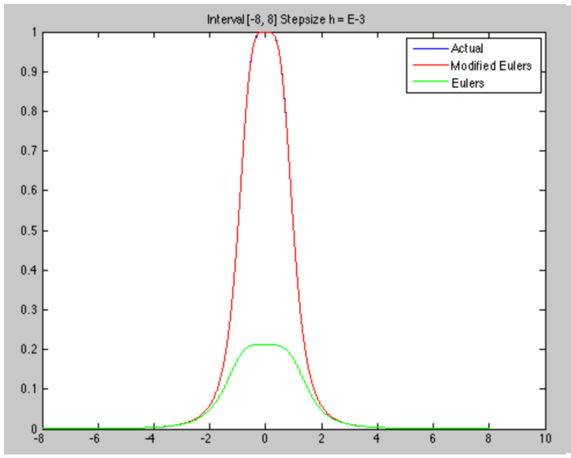
$$\frac{dy}{dt} = -\frac{4t^3}{(t^4+1)^2}$$
$$y' = -4t^3y^2 = -4t^3\left(\frac{1}{(t^4+1)}\right)^2 = -\frac{4t^3}{(t^4+1)^2}$$

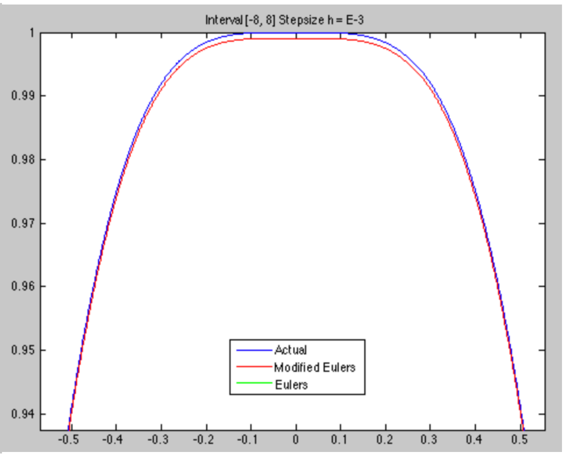
Obviously, the initial condition matches. Therefore, this is indeed the exact solution to the IVP. Also, notice how the solution is an even function, this implies the graph will be symmetric across the y-axis => $y(-a) = y(a)$.

Using values of stepsize $h = 10^{-3}, 10^{-4}, 10^{-5}$, Euler's Method and Modified Euler's Method calculated approximations $w(t_i)$ for actual values of $y(t_i)$. This is modeled in the following graphs:

Graph 1: Euler's and Modified Euler's Method Approximation $w(t_i)$ Compared to $y(t_i)$

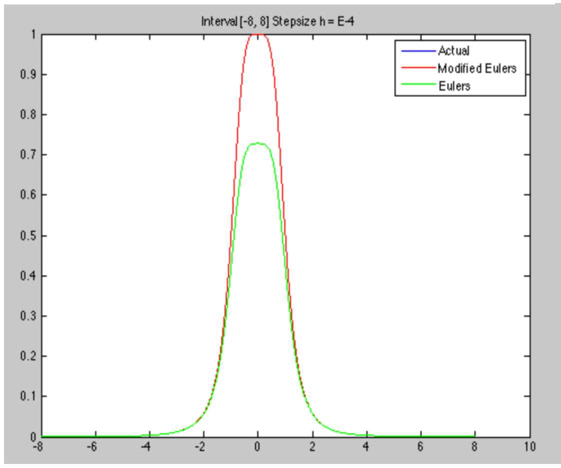
$h = 10^{-3}$

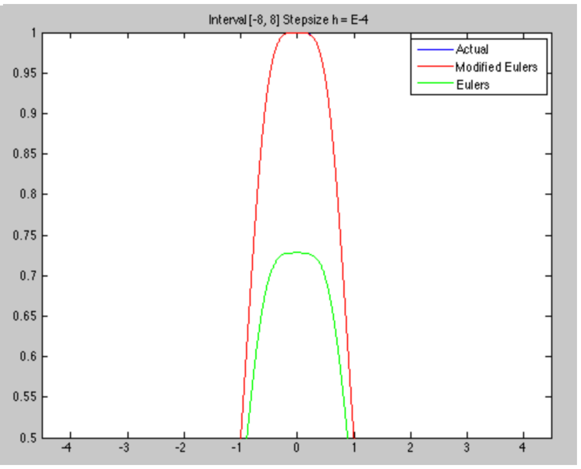




Graph 2: Euler's and Modified Euler's Method Approximation $w(t_i)$ Compared to $y(t_i)$

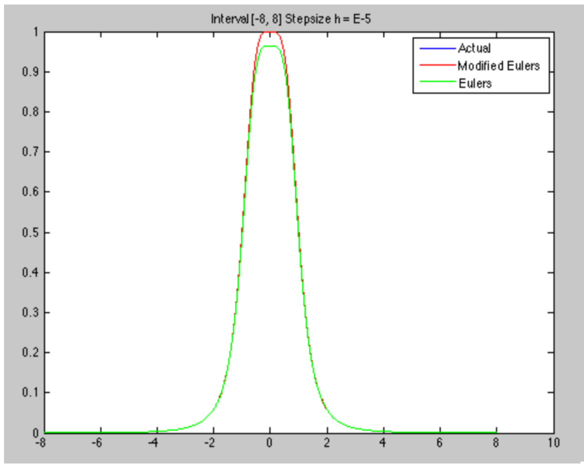
$h = 10^{-4}$

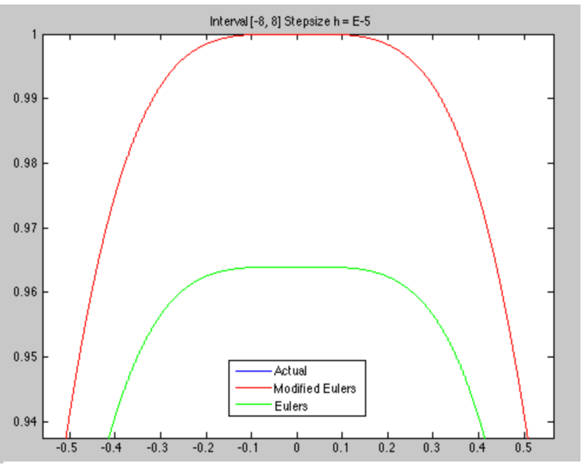




Graph 3: Euler's and Modified Euler's Method Approximation $w(t_i)$ Compared to $y(t_i)$

$h = 10^{-5}$





B By analyzing the output it is obvious that the smaller the stepsize h the smaller the error. The largest part of the error occurs in the middle of the graph causes the larger intervals with same stepsizes to be more spread out. By closing in on $t = 0$ with the interval the central error will have a more sufficient amount of points to be smaller and the endpoint error will be larger than with using a larger interval. By choosing larger intervals in the further away each point will be to the next causing central error to be greater than with a smaller interval and endpoint error will be better than with a smaller interval.

Assuming the true answer is not known, it is possible to estimate the accuracy of Euler's Method for this problem by comparing the Modified Euler's Method to Euler's.

Table 1: Comparing Modified Euler's Method to Euler's Method ($h = 10^{-3}$)

Stepsize	$h = 10^{-3}$		
Method	Euler's	Modified Euler's	Difference
Central Error $\left(t = \frac{N}{2}\right)$	0.787479721389898	0.000921042531047	-0.786558678858851
Boundary Error $(t = h)$	-0.000000122010730	-0.000000122048838	-0.000000000038108
Boundary Error $(t = N + 1)$	-0.000000122048838	-0.000000000000106	0.000000122048732

Table 2: Comparing Modified Euler's Method to Euler's Method ($h = 10^{-4}$)

Stepsize	$h = 10^{-4}$		
Method	Euler's	Modified Euler's	Difference
Central Error $\left(t = \frac{N}{2}\right)$	0.271916413298950	0.000009227004581	-0.271907186294369
Boundary Error $(t = h)$	-0.000000012201073	-0.000000012201454	-0.000000000000381
Boundary Error $(t = N + 1)$	-0.000000012201454	0.000000000000000	0.000000012201454

Table 3: Comparing Modified Euler's Method to Euler's Method ($h = 10^{-5}$)

Stepsize	$h = 10^{-5}$		
Method	Euler's	Modified Euler's	Difference
Central Error $\left(t = \frac{N}{2}\right)$	0.036037320189496	0.000000092234865	-0.036037227954631
Boundary Error $(t = h)$	-0.000000001220107	-0.000000001220111	-0.000000000000004
Boundary Error $(t = N + 1)$	-0.000000001220111	0.000000000000000	0.000000001220111

As seen in the "Difference" columns, Modified Euler's Method is extremely accurate.

Another way the accuracy of Euler's Method can be estimated for this problem is by comparing different stepsizes of h. The difference between Euler's Method's approximate solutions for different stepsizes (second column of the above tables) can be seen in the following table. Note: the correct value of t must be chosen in order to perform this comparison.

Table 4: Comparing Different Stepsizes of Euler's Method (the "Stepsize(#) – Stepsize(#)" corresponds to the Euler values in the second columns (Euler's Method) in the above tables)

Method	Euler's		
Stepsize(#) – Stepsize(#):	$10^{-3} - 10^{-5}$	$10^{-3} - 10^{-4}$	$10^{-4} - 10^{-5}$
Central Error $\left(t = \frac{N}{2}\right)$	0.751442401200402	0.515563308090948	0.235879093109454
Boundary Error $(t = h)$	-0.000000120790623	-0.000000109809657	-0.000000010980966
Boundary Error $(t = N + 1)$	-0.000000120828727	-0.000000109847384	-0.000000010981343

The previous tables show that the smaller the stepsize, the more steps, the smaller the error. If the Difference column is observed more closely it can be seen that the Modified Euler's Method with sufficient stepsize, can produce a fairly accurate solution.

Using the error formulas, an error bound can be obtained analytically for Euler's Method. First, the continuous function f needs to be checked for satisfying a Lipschitz condition on the convex set $D = \{(t, y) \mid a \leq t \leq b \text{ and } -\infty < y < \infty\}$ by taking the partial derivative of f with respect to y. Then by plugging in the exact solution in for y and finding the value of t that makes the partial derivative as large as it can be, a Lipschitz constant L can be found.

$$\left|\frac{\partial f}{\partial y}\right| = |8t^{3y}| = \left|\frac{8t^3}{t^4 + 1}\right|$$

WolframAlpha, computed the max partial derivative of f w.r.t. y at $t = \pm\sqrt[4]{3} \approx \pm 1.32$.

$$\left|\frac{8t^3}{t^4 + 1}\right| \leq \left|\frac{8\sqrt[4]{3}^3}{\sqrt[4]{3}^4 + 1}\right| = \left|\frac{8\sqrt[4]{3}^3}{3 + 1}\right| = \left|\frac{8\sqrt[4]{3}^3}{4}\right| = \left|2\sqrt[4]{3}^3\right| \approx 4.559 \leq 4.6 = L$$

Second, WolframAlpha computed the bound, M, for the second derivative of the unique solution to the initial value problem at $t \approx \pm 0.57$.

$$|y''| = \left|\frac{(4\,t^2\,(-3+5\,t^4))}{(t^4+1)^3}\right| = \left|\frac{(4*0.57^2\,(-3+5*0.57^4))}{(0.57^4+1)^3}\right| \approx 2.377 \le 2.4 = M$$

Third, since f now satisfies a Lipschitz Condition with L on D and the second derivative of the unique solution is bounded by M, the following error formula can be used:

$$|y(t_i) - w_i| \le \frac{hM}{2L}\left(e^{L(t_i-a)} - 1\right)$$

The largest error occurs at $t = 0$ in the interval $[a,b] = [-8,8]$ when $h = 10^{-3}$.

$$\text{Actual error} = |y(t_i) - w_{\mathrm{i}}| = |1 - 0.787| = 0.223$$

$$\text{Euler's error bound} = \frac{hM}{2L}\left(e^{L(-a)} - 1\right) = \frac{10^{-3} * 2.4}{2 * 4.6}\left(e^{4.6(8)} - 1\right) = 2.503\times10^{12}$$

The error computed from the approximations was much better than the analytical bound.

C Assuming that the true solution $y(t)$ and the approximate solution $y(t,h)$ obtained from Euler's Method with step size h satisfies a relation of the form

$$y(t) = y(t,h) + m_1 h + m_2 h^2 + O(h^3) \qquad (1)$$

Richardson's Extrapolation process can be used to show how $O(h^2)$ and $O(h^3)$ approximations to the true solution can be obtained using the results from Euler's method for step sizes h, $0.1h$, and $0.01h$. Specific formulas for these improved approximations can be seen below.

By changing h in (1) to $0.1h$:

$$y(t) = y\left(t,\frac{h}{10}\right) + m_1\frac{h}{10} + m_2\left(\frac{h}{10}\right)^2 + O(\mathrm{h}^3) \qquad (2)$$

By $10 * (2) - (1)$:

$$9y(t) = 10y\left(t,\frac{h}{10}\right) - y(t,h) - \frac{99}{100}m_2(h^2) + O(h^3)$$

$$y(t) = \frac{10}{9}y\left(t,\frac{h}{10}\right) - \frac{1}{9}y(t,h) - \frac{11}{100}m_2(h^2) + O(h^3)$$

where the $O(h^2)$ approximation formula is: $y_2(t,h) = \frac{10}{9}y\left(t,\frac{h}{10}\right) - \frac{1}{9}y(t,h)$

This means that: $y(t) - y_2(t,h) = -\frac{11}{100}m_2(h^2) + O(h^3)$ now,
$y(t) = y_2(t,h) - \frac{11}{100}m_2(h^2) + O(h^3)$ (3)

By changing h in (3) to $0.1h$:
$y(t) = y_2\left(t,\frac{h}{10}\right) - \frac{11}{100}m_2\left(\frac{h}{10}\right)^2 + O(h^3)$ (4)

By $100 * (4) - (3)$:

$$99y(t) = 100y_2\left(t,\frac{h}{10}\right) - y_2(t,h) + O(h^3)$$
$$y(t) = \frac{100}{99}y_2\left(t,\frac{h}{10}\right) - \frac{1}{99}y_2(t,h) + O(h^3)$$

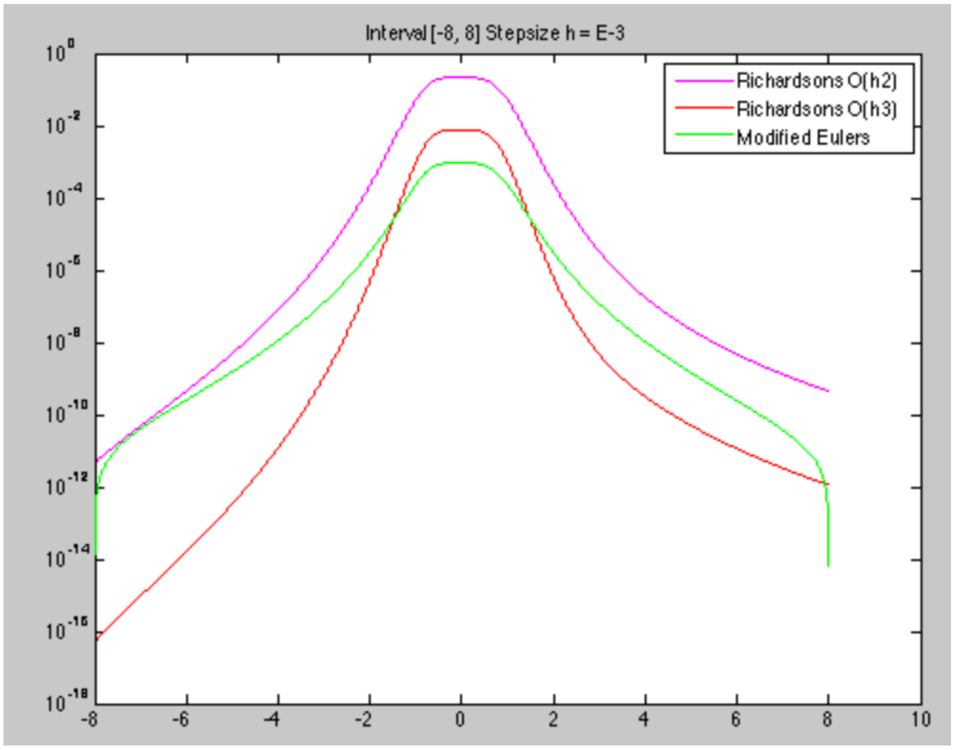
where the $O(h^3)$ approximation formula is: $y_3(t,h) = \frac{100}{99}y_2\left(t,\frac{h}{10}\right) - \frac{1}{99}y_2(t,h)$

$$y_3(t,h) = \frac{100}{99}\left(\frac{10}{9}y\left(t,\frac{\frac{h}{10}}{10}\right) - \frac{1}{9}y\left(t,\frac{h}{10}\right)\right) - \frac{1}{99}\left(\frac{10}{9}y\left(t,\frac{h}{10}\right) - \frac{1}{9}y(t,h)\right)$$
$$y_3(t,h) = \frac{100}{891}\left(10y\left(t,\frac{h}{100}\right) - y\left(t,\frac{h}{10}\right)\right) - \frac{1}{891}\left(10y\left(t,\frac{h}{10}\right) - y(t,h)\right)$$
$$y_3(t,h) = \frac{1}{891}\left(1000y\left(t,\frac{h}{100}\right) - 110y\left(t,\frac{h}{10}\right) + y(t,h)\right)$$

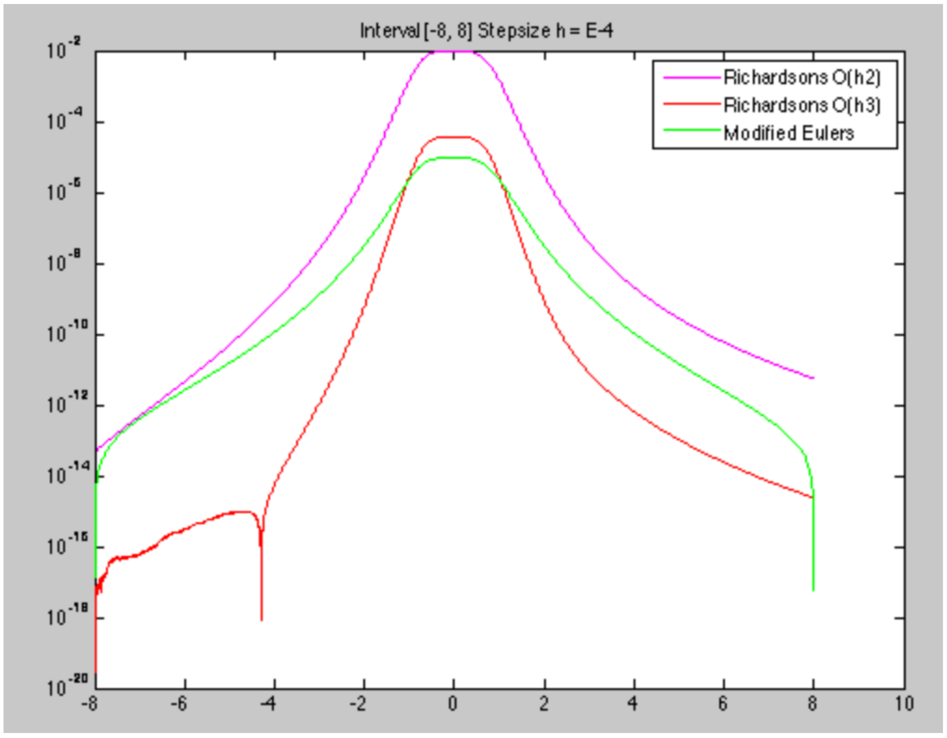
D

The Richardson approximations to the numerical solution of the stated IVP can be computed using the results obtained in (A). The following graphs compare the results of Modified Euler's from part (A) to the Richardson approximations.

Graph 4: Error of Richardson's and Modified Euler's Method
$h = 10^{-3}$

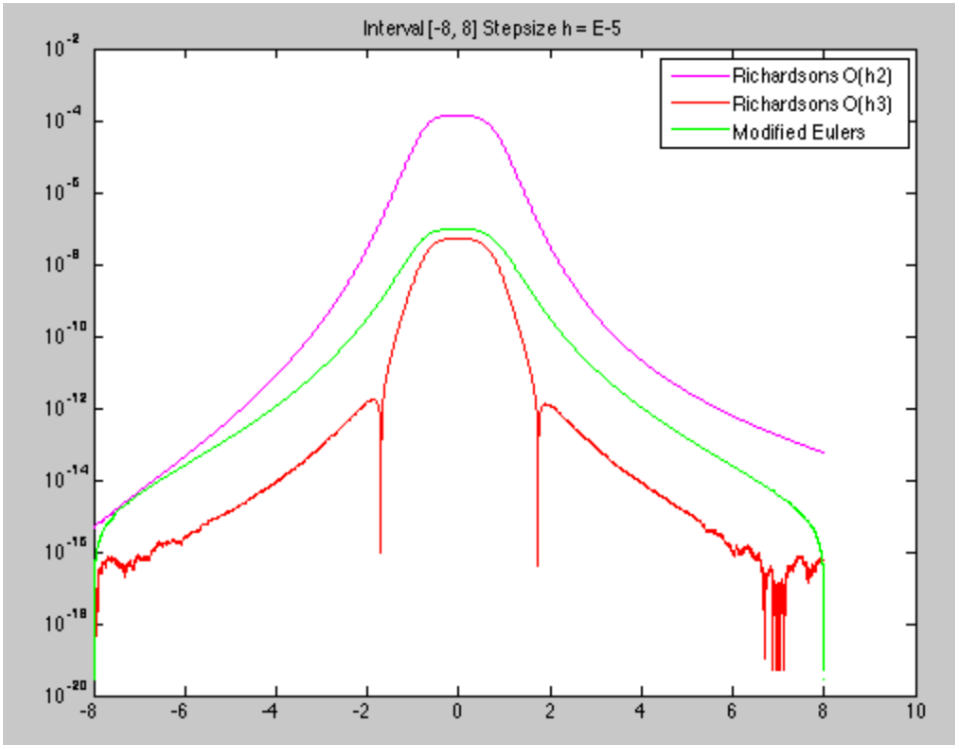


Graph 5: Error of Richardson's and Modified Euler's Method
$h = 10^{-4}$



Graph 6: Error of Richardson's and Modified Euler's Method

$h = 10^{-5}$



Richardson's extrapolation (**eRsons.M**) generates high-accuracy results from low-order formulas. This is a result of the truncation error being dependent on h, and in turn having a predictable form. The object of extrapolation is to find an easy way to combine the $O(h)$ approximations in such a way that the result is a formula with a higher-order truncation error. As shown in the graphs above, Modified Euler's is still extremely accurate, even more accurate with central error than Richardson's $O(h^3)$ formula until $h = 10^{-5}$, Richardson's $O(h^3)$ formula becomes even more accurate than Modified Euler's. The error at the beginning of the interval is better than the end of the interval because the truncation error propagates throughout the method.

Appendix (4 functions + 1 script)

```
% function f(t,y)
function fty = fty(t,y)
    fty = -4 * t^3 * y^2;
end


function [ approximations,iterations ] = em( a, b, h, c )
%em Euler's Method for solving IVPs
%       To approximate the solution to the IVP
%       y' = f(t,y)  ,  a <= t <= b  ,  y(a) = c
%       at N+1 equally spaced numbers in the interval [a,b]

%   INPUT: a: left endpoint
%          b: right endpoint
%          h: stepsize
%          c: initial condition (alpha)
%   OUTPUT: approximations: approximation to y at time t
%           iterations: N + 1 iterations of time

%   Step 1: Set N = (b-a)/h;
%               t = a;
%               w = c;
%           OUTPUT (t,w)
%   Step 2: For i = 1:N % do Steps 3, 4
%       Step 3: Set w(i) = w + h*f(t,w); % Compute w(i)
%                   t(i) = a + i*h; % Compute t(i)
%       Step 4: OUTPUT t(i),w(i)
%   Step 5: STOP

format long;

% Step 1
N = round((b-a)/h);
w = zeros(N + 1,1);
w(1) = c;
t = zeros(N+1,1);
```

```
t(1) = a;

% Step 2
for i = 2:N+1 % do Steps 3
    % Step 3: Compute w(i) and t(i)
    w(i) = w(i-1) + h*fty(t(i-1),w(i-1));
    t(i) = a + i*h;
end

% Step 4
approximations = w;
iterations = t;
end



function [ approximations,iterations ] = mem( a, b, h, c )
%mem Modified Euler's Method for solving IVPs
%       To approximate the solution to the IVP
%       y' = f(t,y)  ,  a <= t <= b  ,  y(a) = c
%       at N+1 equally spaced numbers in the interval [a,b]

%   INPUT: a: left endpoint
%          b: right endpoint
%          h: stepsize
%          c: initial condition (alpha)
%   OUTPUT: approximations: approximation to y at time t
%           iterations: N + 1 iterations of time

%   Step 1: Set N = (b-a)/h;
%               t = a;
%               w = c;
%           OUTPUT (t,w)
%   Step 2: For i = 1:N % do Steps 3, 4
%       Step 3: % Compute w(i) and t(i)
%               Set w(i) = w(i-1) + (h/2)*(fty(t(i-1), w(i-1)) +
%                   fty(t(i-1)+h, w(i-1) + h*fty(t(i-1), w(i-1))));
%                   t(i) = a + i*h;
%       Step 4: OUTPUT t(i),w(i)
%   Step 5: STOP

format long;

% Step 1
N = round((b-a)/h);
w = zeros(N+1,1);
w(1) = c;
t = zeros(N+1,1);
t(1) = a;

% Step 2
for i = 2:N+1 % do Steps 3
    % Step 3:  Compute w(i) and t(i)
    w(i) = w(i-1) + (h/2)*(fty(t(i-1), w(i-1)) + fty(t(i-1)+h, w(i-1) + h*fty(t(i-1), w(i-
1))));
    t(i) = a + (i-1)*h;
end

% Step 4
approximations = w;
iterations = t;
end

function [ y2,y3,iterations ] = eRsons( a,h,c )
%eRsons Richardsons Extrapolation Process on Eulers Method for solving IVPs
%   Detailed explanation is in part C

%   INPUT: a: interval
%              h: stepsize
%              c: initial value
%   OUTPUT: y2: Richardson O(h2) approximation
%                 y3: Richardson O(h3) approximation
%                 iterations: N + 1 iterations of time

format long;

% Step 1
```

```
N = round((a--a)/h);
yn = c;

[we1,te1] = em(-a,a,h,yn);
[we2] = em(-a,a,0.1*h,yn);
[we3] = em(-a,a,0.01*h,yn);
y2 = zeros(length(te1),1);
y3 = zeros(length(te1),1);

for i = 1:length(te1)
    y2(i) = (10*we2(10*(i-1)+1)-we1(i))/9;
    y3(i) = (1000*we3(100*(i-1)+1)-110*we2(10*(i-1)+1)+we1(i))/891;
end

iterations = te1;
end
```

%% SCRIPT

```
clear all;
format long;

% Stepsize (10^(-3, -4, and -5))
h = 10^-3;

% BC: 5 <= a <= 10
% a = 5;
a = 8;

% Number of steps
N = round((a--a)/h);

% IC % y(-a) = y(a) since y is an even function
yn = 1/((-a)^4 + 1);

% Actual Value
actual = zeros(round(N+1),1);
i = 1;
for x = -a:h:a
    actual(i) = 1/(x.^4 + 1);
    i = i+1;
end

% Euler's Method
tic
[we,te] =  em(-a,a,h,yn);
toc

% Modified Euler's Method
tic
[wm,tm] = mem(-a,a,h,yn);
toc

% Richardson's Extrapolation
tic
[R2,R3,tr] = eRsons(a,h,yn);
toc
```

```
% Graphs
% Actual vs Euler's vs Modified Euler's
figure
plot(tm,actual,'b')
hold on
plot(tm,wm,'r')
hold on
plot(te,we,'g')
hold on
legend('Actual','Modified Eulers','Eulers')
AMEtitle = ['Interval [', num2str(-a), ', ', num2str(a), '] Stepsize h = E',
num2str(log10(h))];
title(AMEtitle)

% Error of Actual vs Richardson's and Modified Euler's
figure
semilogy(tr,abs(R2-actual),'m')
hold on
semilogy(tr,abs(R3-actual),'r')
hold on
semilogy(tr,abs(wm-actual),'g')
hold on
legend('Richardsons O(h2)','Richardsons O(h3)','Modified Eulers')
ARMtitle = ['Interval [', num2str(-a), ', ', num2str(a), '] Stepsize h = E',
num2str(log10(h))]
title(ARMtitle)

% error @ center
ce  = 1 - we(N/2);
cm  = 1 - wm(N/2);
cr2 = 1 - R2(N/2);
cr3 = 1 - R3(N/2);
fprintf('            Eulers Central  Error: b = %16.15f \n',ce)
fprintf('  Modified Eulers Central  Error: b = %16.15f \n',cm)
fprintf('Richardsons O(h2) Central  Error: b = %16.15f \n',cr2)
fprintf('Richardsons O(h3) Central  Error: b = %16.15f \n',cr3)
```

```
% error @ boundary
bbe = yn - we(2);
bee = yn - we(N+1);
bbm = yn - wm(2);
bem = yn - wm(N+1);
bbr2 = yn -  R2(2);
ber2 = yn -  R2(N+1);
bbr3 = yn -  R3(2);
ber3 = yn -  R3(N+1);
fprintf('             Eulers Boundary Error: bbe = %16.15f \n',bbe)
fprintf('             Eulers Boundary Error: bee = %16.15f \n',bee)
fprintf('  Modified Eulers Boundary Error: bbm = %16.15f \n',bbm)
fprintf('  Modified Eulers Boundary Error: bem = %16.15f \n',bem)
fprintf('Richardsons O(h2) Boundary Error: bbr2 = %16.15f \n',bbr2)
fprintf('Richardsons O(h2) Boundary Error: ber2 = %16.15f \n',ber2)
fprintf('Richardsons O(h3) Boundary Error: bbr3 = %16.15f \n',bbr3)
fprintf('Richardsons O(h3) Boundary Error: ber3 = %16.15f \n',ber3)
```