Numerical Analysis Project: Euler's Method, Modified Euler's Method and Richardson's Extrapolation

Cameron Weigel

May 6, 2016

This project will demonstrate how to numerically and analytically solve first order initial-value problems using Euler's Method and Modified Euler's Method. The initial-value problems are of the form:

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

1 Analysis

The purpose of this project was to analytically and numerically show how Euler's Method for approximating solutions to differential equations is implemented. We begin by finding the exact solution to our initial-value problem. In many cases, finding the actual solution to the differential equation in nearly impossible. To overcome this issue we apply methods of approximating the solution; in this project we achieve this by implementing Euler's Method and Modified Euler's Method. When implementing Euler's method, we discuss the error bounds on the approximations in a formal manner. Finally, we apply a Richardson's extrapolation process to the data to show how we can get improved approximations to the solution.

1.1 Definitions

Before discussing Euler's Method and its variants, certain definitions should be stated:

1. A function $f(t, y)$ is said to satisfy a **Lipschitz condition** in the variable y on a set $D \subset \mathbb{R}^2$ if a constant $L < 0$ exists with

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|,$$

whenever (t, y_1) and (t, y_2) are in D. The constant L is called a **Lipschitz constant** for f.[1]

2. A set $D \subset \mathbb{R}^2$ is said to be **convex** if whenever (t_1, y_1) and (t_2, t_2) belong to D, then $((1-\lambda)t_1 + \lambda t_2, (1-\lambda)y_1 + \lambda y_2)$ also belongs to D for every λ in $[0, 1]$.[2]

3. The initial-value problem

$$\frac{dy}{dt} = f(t, y), \quad a \leq t \leq b, \quad y(a) = \alpha$$

is said to be a **well-posed problem** if:

- A unique solution, $y(t)$, to the problem exists, and
- There exist constants $\epsilon_0 > 0$ and $k > 0$ such that for any ϵ, in $(0, \epsilon_0)$, whenever $\delta(t)$ is continuous with $|\delta(t)| < \epsilon$ for all t in $[a, b]$, and when $|\delta_0| < \epsilon$, the initial value problem

$$\frac{dz}{dt} = f(t, z) + \delta(t), \quad a \leq t \leq b, \quad z(a) = \alpha + \delta_0,$$

has a unique solution $z(t)$ that satisfies

$$|z(t) - y(t)| < k\epsilon \quad \text{for all } t \text{ in } [a, b].^{(3)}$$

Note: We have assumed well-posedness for the initial-value problem given but there are many cases where this is not true. Therefore, the definitions posed above are still worth mentioning since this criterion is met only on occasion.

1.2 Solution to the initial-value problem

This section contains the details showing that the exact solution to

$$y' = (-1/25)(t-2)^3 y^2; \quad y(-a) = \frac{1}{((-a-2)^4 + 1)}; \quad t \in [-a, a]$$

$$\text{is} \quad y(t) = \frac{100}{((t-2)^4 + 1)}.$$

$$y' = (-1/25)(t-2)^3 y^2$$

$$\frac{25y'}{y^2} = -(t-2)^3$$

$$\int \frac{25\frac{dy}{dt}}{y^2}dt = \int -(t-2)^3 dt$$

$$\int \frac{25}{y^2}dy = \int -(t-2)^3 dt$$

$$-25y^{-1} + c_1 = -\frac{1}{4}(t-2)^4 + c_2$$

$$\frac{1}{y} + c_1 = \frac{1}{100}(t-2)^4 + c_2$$

Combining c_1 and c_2 into C we get

$$y = \frac{100}{(t-2)^4 + C}$$

Finally, using $t \in [-5, 5]$ we will use our initial condition to show that the solution to the initial-value problem is exactly $y(t) = \frac{100}{((t-2)^4+1)}$

$$y(-5) = \frac{100}{(-5-2)^4 + C} = \frac{100}{(-5-2)^4 + 1}$$

Let $C = 1$ and we have our solution.

1.3 Error Analysis

Since Euler's method is a simple approximation to the actual solution, it is not difficult to show the accuracy and subsequently the error for this method. From Theorem 5.9, we have that the approximations generated by Euler's method are of the form

$$|y(t_i) - w_i| \leq \frac{hM}{2L}[e^{L(t_i-a)} - 1]$$

where

$$|y''(t)| \leq M, \quad \text{for all} \quad t \in [a, b] \quad \text{and} \quad L \quad \text{is our Lipschitz constant}^4$$

To find our Lipschitz constant L we apply the condition:

$$|f(t, y_1) - f(t, y_2) \leq L|y_1 - y2|$$

$$|(-1/25)(t-2)^3y_1^2 - (-1/25)(t-2)^3y_2^2| \leq L|y_1 - y_2|$$
$$|(y_1^2 - y_2^2)[(-1/25)(t-2)^3]| \leq L|y_1 - y_2|$$
$$|(y_1 - y_2)(y_1 + y_2)[(-1/25)(t-2)^3]| \leq L|y_1 - y_2|$$
$$|(y_1 + y_2)[(-1/25)(t-2)^3]| \leq L$$

Notice that the maximum for both y_1 and y_2 should be the same so we can set $(y_1+y_2) = 2y$ Now by using our solution for $y(t)$ we get,

$$|(-1/25)(t-2)^3[\frac{200}{(t-2)^4+1)}]| \leq L$$

Finally,

$$\max_{-5\leq t\leq 5} \frac{-8(t-2)^3}{(t-2)^4+1} \leq 4.6 = L$$

Next we must find M, Using our exact solution we get,

$$y'(t) = (-1/25)(t-2)^3[\frac{100}{(t-2)^4+1}]^2$$

taking the derivative of $y'(t)$ yields,

$$y''(t) = \frac{400(8(t-2)^4 - 3((t-2)^4+1)(t-2)^2}{((t-2)^4+1)^3}$$

and by maximizing $y''(t)$ with $t \in [-5, 5]$ we get,

$$\max_{-5\leq t\leq 5} \frac{400(8(t-2)^4 - 3((t-2)^4+1)(t-2)^2}{((t-2)^4+1)^3} \leq 147 = M$$

Substituting $L = 4.6$ and $M = 147$ into the error formula,

$$|y(t_i) - w_i| \leq \frac{147h}{9.2}[e^{4.6(t_i+5)} - 1]$$

As you can see, our maximum error bound depends on the size of h, the table below shows a few bounds for different values of h.

h	$\lvert y(t_i) - w_i\rvert$
10^{-3}	$\frac{.147}{9.2}[e^{4.6(t_i+5)} - 1]$
10^{-4}	$\frac{.0147}{9.2}[e^{4.6(t_i+5)} - 1]$
10^{-5}	$\frac{.00147}{9.2}[e^{4.6(t_i+5)} - 1]$

1.4 Richardson's Extrapolation

In this section, we derive the Richardson's Extrapolation for step-size h $.01h$ and $.001h$ to show that we can increase the accuracy of Euler's method from $\mathcal{O}(h)$ to $\mathcal{O}(h^3)$. We have assumed that the error between the actual solution and Euler's method is of the form:

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

1.4.1 Extrapolation for h

$$y(t) = y(t,\frac{h}{2}) + m_1\frac{h}{2} + m_2\frac{h^2}{4} + \mathcal{O}(h^3) \qquad (2)$$

By subtracting (1) from two times (2) we get,

$$2y(t) - y(t) = 2[y(t,\frac{h}{2}) + m1\frac{h}{2} + m_2\frac{h^2}{4} + \mathcal{O}(h^3)] - [y(t,h) + m_1 h + m_2 h^2 + \mathcal{O}(h^3)]$$

This can be rewritten as,

$$y(t) = y(t,\frac{h}{2}) + [y(t,\frac{h}{2}) - y^\star(t,h)] - m_2\frac{h^2}{2} + \mathcal{O}(h^3)$$

$$\text{Where } y^\star(t,h) = y(t,\frac{h}{2}) + [y(t,\frac{h}{2}) - y(t,h)]$$

Then we have our first extrapolation,

$$y(t) = y^\star(t,h) - m_2\frac{h^2}{2} + \mathcal{O}(h^3)$$

and we have shown how to improve Euler's method to error $\mathcal{O}(h^2)$

To show Euler's method with $\mathcal{O}(h^3)$, the process is very similar. First we start with our new approximation with $\mathcal{O}(h^2)$.

$$y(t) = y^\star(t,h) - m_2\frac{h^2}{2} + \mathcal{O}(h^3) \quad (3)$$

Since we have the well-posed condition we can apply the same rule as before,

$$y(t) = y^\star(t,\frac{h}{2}) - m_2\frac{h^2}{4} + \mathcal{O}(h^3) \quad (4)$$

By subtracting (3) from two times (4) we get,

$$2y(t) - y(t) = 2[y^\star(t,\frac{h}{2}) - m_2\frac{h^2}{4} + \mathcal{O}(h^3)] - [y^\star(t,h) - m_2\frac{h^2}{2} + \mathcal{O}(h^3)]$$

This can be rewritten as,

$$y(t) = y^{\star}(t, \frac{h}{2}) + [y^{\star}(t, \frac{h}{2}) - y^{\star\star}(t, h)] + \mathcal{O}(h^3)$$

$$\text{Where } y^{\star\star}(t, h) = y^{\star}(t, \frac{h}{2}) + [y^{\star}(t, \frac{h}{2}) - y^{\star}(t, h)]$$

Then we have our second extrapolation,

$$y(t) = y^{\star\star}(t, h) + \mathcal{O}(h^3)$$

and we have shown how to improve Euler's method to error $\mathcal{O}(h^2)$

1.4.2 Extrapolation for $0.01h$

$$y(t) = y(t, \frac{h}{10}) + m_1 \frac{h}{10} + m_2 \frac{h^2}{100} + \mathcal{O}(h^3) \qquad (5)$$

$$y(t) = y(t, \frac{h}{20}) + m1 \frac{h}{20} + m_2 \frac{h^2}{200} + \mathcal{O}(h^3) \qquad (6)$$

By subtracting (5) from two times (6) we get,

$$2y(t) - y(t) = 2[y(t, \frac{h}{20}) + m1 \frac{h}{20} + m_2 \frac{h^2}{200} + \mathcal{O}(h^3)] - [y(t, \frac{h}{10}) + m_1 \frac{h}{10} + m_2 \frac{h^2}{100} + \mathcal{O}(h^3)]$$

This can be rewritten as,

$$y(t) = y(t, \frac{h}{20}) + [y(t, \frac{h}{20}) - y^{\star}(t, \frac{h}{10})] - m_2 \frac{h^2}{200} + \mathcal{O}(h^3)$$

$$\text{Where } y^{\star}(t, \frac{h}{10}) = y(t, \frac{h}{20}) + [y(t, \frac{h}{20}) - y(t, \frac{h}{10})]$$

Then we have our first extrapolation,

$$y(t) = y^{\star}(t, \frac{h}{10}) - m_2 \frac{h^2}{200} + \mathcal{O}(h^3)$$

and we have shown how to improve Euler's method to error $\mathcal{O}(h^2)$

We will apply the same process as before to improve Euler's method to error $\mathcal{O}(h^3)$. We start with our new approximation with $\mathcal{O}(h^2)$.

$$y(t) = y^{\star}(t, \frac{h}{10}) - m_2 \frac{h^2}{200} + \mathcal{O}(h^3) \qquad (7)$$

$$y(t) = y^{\star}(t, \frac{h}{20}) - m_2 \frac{h^2}{400} + \mathcal{O}(h^3) \qquad (8)$$

By subtracting (7) from two times (8) we get,

$$2y(t) - y(t) = 2[y^{\star}(t, \frac{h}{20}) - m_2\frac{h^2}{400} + \mathcal{O}(h^3)] - [y^{\star}(t, \frac{h}{10}) - m_2\frac{h^2}{200} + \mathcal{O}(h^3)]$$

This can be rewritten as,

$$y(t) = y^{\star}(t, \frac{h}{20}) + [y^{\star}(t, \frac{h}{20}) - y^{\star\star}(t, \frac{h}{10})] + \mathcal{O}(h^3)$$

$$\text{Where } y^{\star\star}(t, \frac{h}{10}) = y^{\star}(t, \frac{h}{20}) + [y^{\star}(t, \frac{h}{20}) - y^{\star}(t, \frac{h}{10})]$$

Then we have our second extrapolation,

$$y(t) = y^{\star\star}(t, \frac{h}{10}) + \mathcal{O}(h^3)$$

1.4.3 Extrapolation for $0.001h$

$$y(t) = y(t, \frac{h}{100}) + m_1\frac{h}{100} + m_2\frac{h^2}{1000} + \mathcal{O}(h^3) \qquad (9)$$

$$y(t) = y(t, \frac{h}{200}) + m1\frac{h}{200} + m_2\frac{h^2}{2000} + \mathcal{O}(h^3) \qquad (10)$$

By subtracting (9) from two times (1) we get,

$$2(t) - y(t) = 2[y(t, \frac{h}{200}) + m1\frac{h}{200} + m_2\frac{h^2}{2000} + \mathcal{O}(h^3)] - [y(t, \frac{h}{100}) + m_1\frac{h}{100} + m_2\frac{h^2}{1000} + \mathcal{O}(h^3)]$$

This can be rewritten as,

$$y(t) = y(t, \frac{h}{200}) + [y(t, \frac{h}{200}) - y^{\star}(t, \frac{h}{100})] - m_2\frac{h^2}{2000} + \mathcal{O}(h^3)$$

$$\text{Where } y^{\star}(t, \frac{h}{100}) = y(t, \frac{h}{200}) + [y(t, \frac{h}{200}) - y(t, \frac{h}{100})]$$

Then we have our first extrapolation,

$$y(t) = y^{\star}(t, \frac{h}{100}) - m_2\frac{h^2}{2000} + \mathcal{O}(h^3)$$

and we have shown how to improve Euler's method to error $\mathcal{O}(h^2)$

We will apply the same process as before to improve Euler's method to error $\mathcal{O}(h^3)$. We start with our new approximation with $\mathcal{O}(h^2)$.

$$y(t) = y^{\star}(t, \frac{h}{100}) - m_2\frac{h^2}{2000} + \mathcal{O}(h^3) \qquad (11)$$

$$y(t) = y^{\star}(t, \frac{h}{200}) - m_2 \frac{h^2}{4000} + \mathcal{O}(h^3) \qquad (12)$$

By subtracting (11) from two times (12) we get,

$$y(t) = 2[y^{\star}(t, \frac{h}{200}) - m_2 \frac{h^2}{4000} + \mathcal{O}(h^3)] - [y^{\star}(t, \frac{h}{100}) - m_2 \frac{h^2}{2000} + \mathcal{O}(h^3)]$$

This can be rewritten as,

$$y(t) = y^{\star}(t, \frac{h}{200}) + [y^{\star}(t, \frac{h}{200}) - y^{\star\star}(t, \frac{h}{100})] + \mathcal{O}(h^3)$$

$$\text{Where } y^{\star\star}(t, \frac{h}{100}) = y^{\star}(t, \frac{h}{200}) + [y^{\star}(t, \frac{h}{200}) - y^{\star}(t, \frac{h}{100})]$$

Then we have our second extrapolation,

$$y(t) = y^{\star\star}(t, \frac{h}{100}) + \mathcal{O}(h^3)$$

2 Code

This section contains the code used to solve the initial-value problem:

$$y' = (1/25)(t-2)^2 y^3; \quad y(-a) = \frac{1}{((-a-2)^4+1)}; \quad t \in [-a, a]$$

Note: This code is written in such a manner so that it can solve any first order initial value problem.

2.1 Euler's Method

This code has been pulled from the Euler.m file which has been reformatted for convenience, comments are written in blue and black text is actual code.

```
Written by: Cameron Weigel
TITLE:
Euler's Method for solving ODEs
GOAL:
Solve f(t,y) = y'(t) 1st Order ODEs

function [TS WS] = Euler(f,a,b,h,alpha)
```

```
INPUTS:
f is a function handle that takes a 1st Order ODE as input
[a, b] is the interval chosen for the 1st Order ODE
h is the stepsize used to approximate the points of the equation y(t)
alpha is the initial condition in the ODE
OUTPUTS:
TS is a vector containing the interval [a,b] partitioned by h
WS is a vector containing approximations to the solution at the TS partitions

format long;   Allows more precision

w = alpha;   Initial condition

i = 1;   This will be our iterator variable

t = a;   Beginning of our interval

while (t < b)   While we are still in our interval divided by the stepsize

    w = w + h*(f(t,w));   Euler Iteration
    t = a+i*h;   Increase the partition we are iterating over

    ts(i+1) = t;   Save our partitions in a vector
    ws(i+1) = w;   Save our approximations in a vector

    i = i+1;   Increase our iterator by 1

end

TS = ts;   Output our partitioned interval
WS = ws;   Output our approximations

Below is the subroutine that plots the approximation of the solution to the ODE

figure;
plot(TS,WS);
title(['Euler Approximation of the solution with ' num2str(h) ' step size'])
print('Eulerplot','-dpng');
end
```

2.2 Modified Euler's Method

This code has been pulled from the ModEuler.m file which has been reformatted for convenience, comments are written in blue and black text is actual code.

```
Written by: Cameron Weigel
TITLE:
Modified Euler's Method for solving ODEs
GOAL:
Solve f(t,y) = y'(t) 1st Order ODEs

function [TS WS] = ModEuler(f,a,b,h,alpha)

INPUTS:
f is a function handle that takes a 1st Order ODE as input
[a, b] is the interval chosen for the 1st Order ODE
h is the stepsize used to approximate the points of the equation y(t)
alpha is the initial condition in the ODE
OUTPUTS:
TS is a vector containing the interval [a,b] partitioned by h
WS is a vector containing approximations to the solution at the TS partitions

format long;   Allows more precision

w = alpha;   Initial condition

i = 1;   This will be our iterator variable

t = a;   Beginning of our interval

while (t < b)   While we are still in our interval divided by the stepsize

    w = w + (h/2)*(f(t,w)+f((t+h),(w+h*(f(t,w))));   Modified Euler's Iteration
    t = a+i*h;   Increase the partition we are iterating over

    ts(i+1) = t;   Save our partitions in a vector
    ws(i+1) = w;   Save our approximations in a vector

    i = i+1;   Increase our iterator by 1

end
```

```
TS = ts;    Output our partitioned interval
WS = ws;    Output our approximations

Below is the subroutine that plots the approximation of the solution to the ODE

figure;
plot(TS,WS);
title(['Modified Euler Approximation of the solution with ' num2str(h) ' step size'])
print('ModifiedEulerplot','-dpng');
end
```

2.3 Richardson's Extrapolation

```
Applies Richardson's first extrapolation to two vectors


INPUTS:

Two vectors WS1 with step size h, and WS2 with step size h/2


OUTPUT:

Vector RS with the first Richardson Extrapolation of WS1 and WS2



    function [RS] = Richardsons(WS1,WS2)

format long;
n = length(WS1);
k = length(WS2);

rs = zeros(n,1);
rs(1)=WS1(1);


for i = 2:n
```

```
if (2*i ¡ k)
rs(i)=WS2(2*i)+(WS2(2*i)-WS1(i));

end
end

RS = rs;
```

3 Results

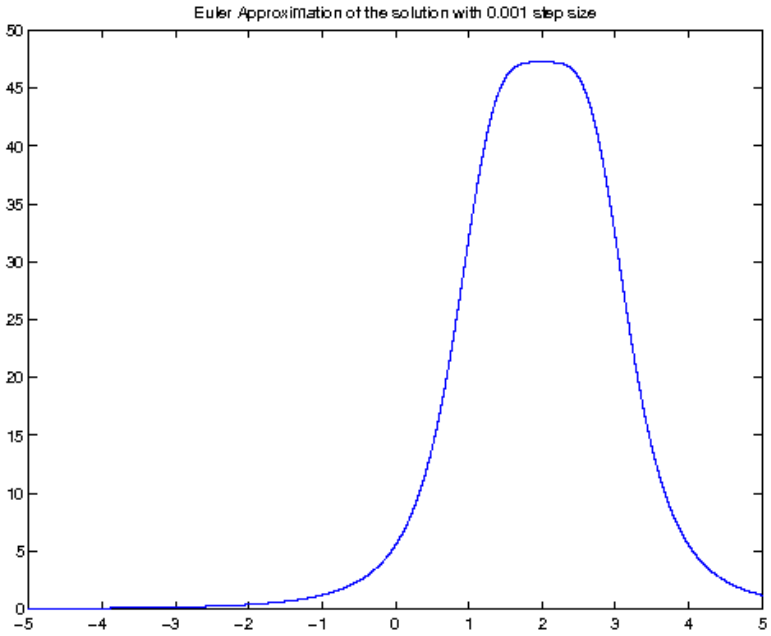
Overall, the results were as expected in most cases. Euler's method had increased error as we approximate further down the interval, with a highest error at $t = 2$ which is an interesting point in our actual solution. Euler's modified method did much better in approximating our solution at all points and was less prone to inaccuracy along the interval. Finally, Richardson's first extrapolation providing some surprising results, it's error was much higher than originally anticipated.

3.1 Euler's Method

Below are the tables and graphs for Euler's Method with step sizes $h = 10^{-3}, 10^{-4}$, and 10^{-5}

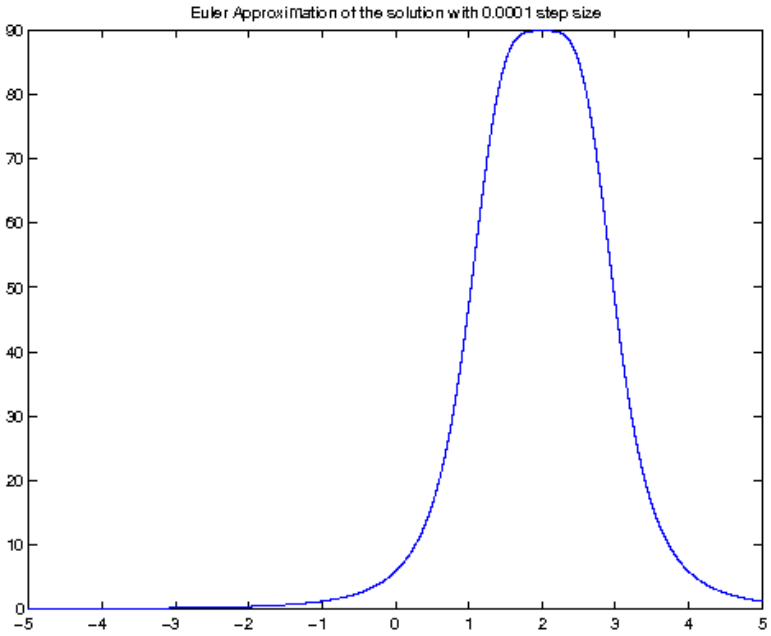
$h = 10^{-3}$

t=	w=
-5	0.0416319...
-4.999	0.0416557...
-4.998	0.0416795...
-4.997	0.0417033...
-4.996	0.0417271...
⋮	⋮
5	1.2021105...



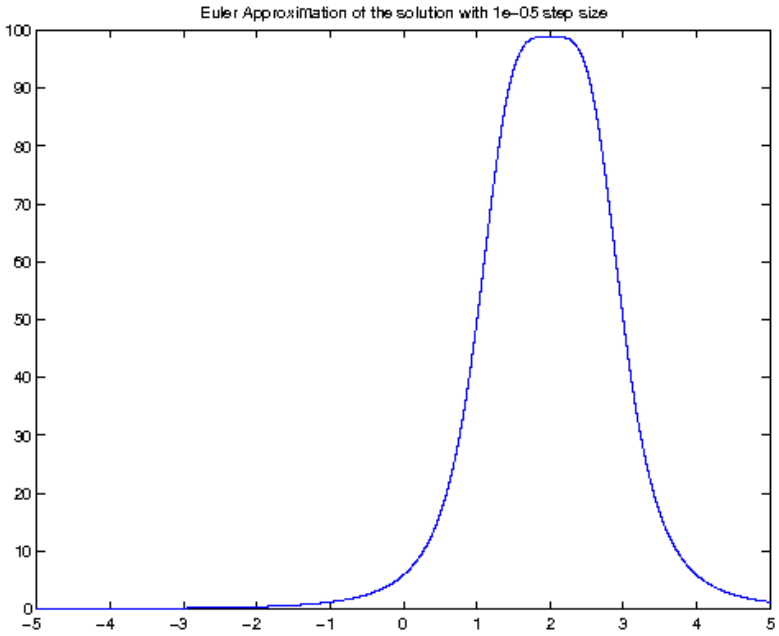
$h = 10^{-4}$

t=	w=
-5	0.0416319...
-4.9999	0.0416343...
-4.9998	0.0416367...
-4.9997	0.0416391...
-4.9996	0.0416414...
⋮	⋮
5	1.2177252...



$h = 10^{-5}$

t=	w=
-5	0.0416319...
-4.99999	0.0416322...
-4.99998	0.0416324...
-4.99997	0.0416326...
-4.99996	0.0416329...
⋮	⋮
5	1.2193329...

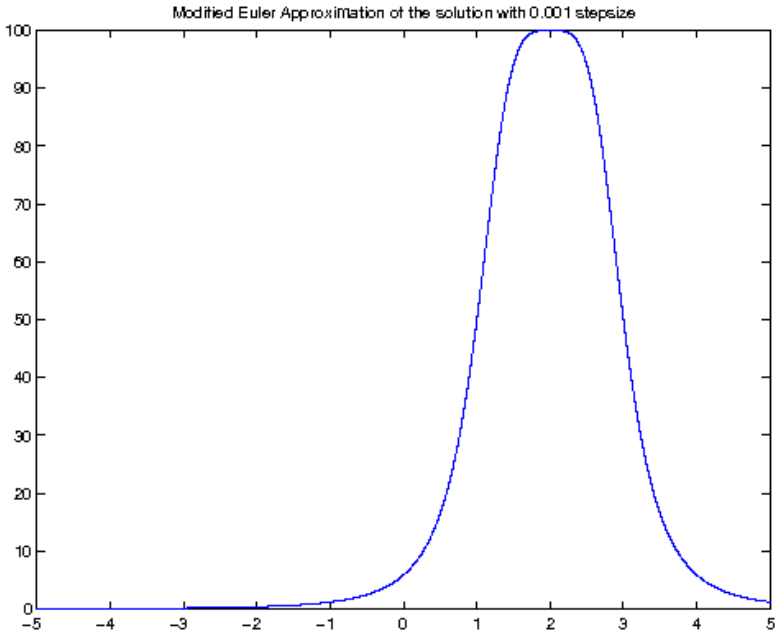


3.2 Euler's Modified Method

Below are the tables and graphs for Euler's Modified Method with step sizes $h = 10^{-3}, 10^{-4}$, and 10^{-5}

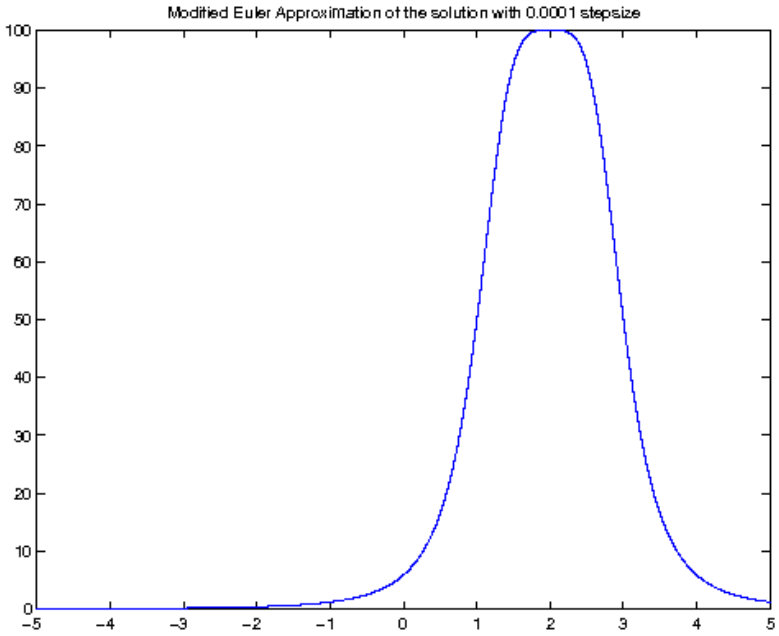
$h = 10^{-3}$

t=	w=
-5	0.0416319...
-4.999	0.0416557...
-4.998	0.0416795...
-4.997	0.0417033...
-4.996	0.0417272...
⋮	⋮
5	1.2195033...



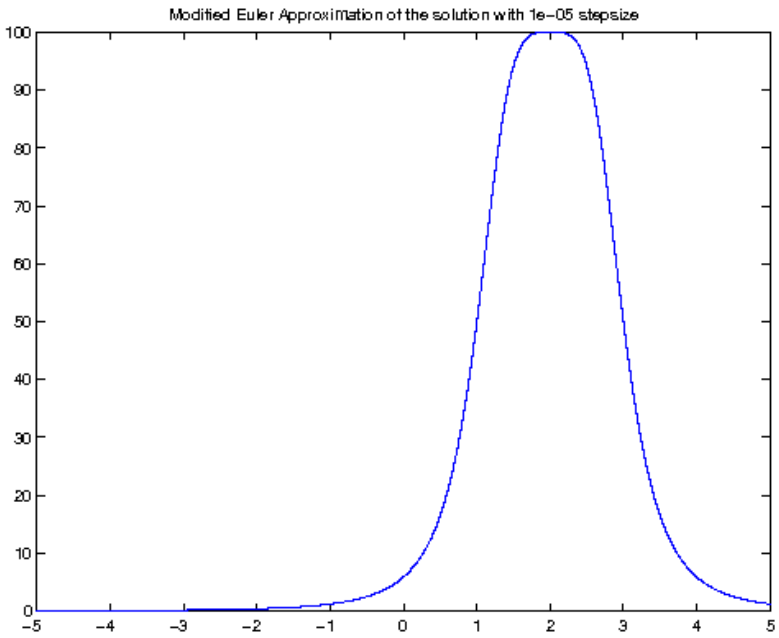
$h = 10^{-4}$

t=	w=
-5	0.0416319...
-4.9999	0.0416343...
-4.9998	0.0416367...
-4.9997	0.0416391...
-4.9996	0.0416414...
⋮	⋮
5	1.2195121...



$h = 10^{-5}$

t=	w=
-5	0.0416319...
-4.99999	0.0416322...
-4.99998	0.0416324...
-4.99997	0.0416326...
-4.99996	0.0416329...
⋮	⋮
5	1.2195121...



The graphs indicate how accurate Modified Euler's approximations are to the actual solution. Even with a step size of $h = 10^{-3}$, Modified Euler's method is more than twice as accurate as Euler's method at $t = 2$ with the same step size.

3.3 Approximation Error

This section contains just an echoing of the Error between the different methods and the actual solution. Euler's method followed the pattern expected where as we traverse down our interval, the approximations become less accurate due to the growing error from each previous approximation. Modified Euler's method had exceptional accuracy which is also expected, by adding more terms to the Taylor series approximation at each step, we are able to achieve a much higher accuracy. Finally, I did not expect Richardson's first extrapolation method to have relatively high errors compared to Euler's method. While Richardson's method did approximate $t = 2$ better than Euler's, its error terms in the beginning of our interval were higher than Euler's error terms. I believe this is probably an issue with my code that needs to be resolved. I wrote my Richardson's Extrapolation in a separate file to allow some modularity between my programs but maybe there was a lot of data truncated by moving between the two functions.

3.3.1 Euler's Method

$$h = 10^{-3}$$

t	$\lvert y(t) - w(t) \rvert$
-5	0
-4.999	$8.4873677... \times 10^{-11}$
-4.998	$1.6979435... \times 10^{-10}$
-4.997	$2.5467197... \times 10^{-10}$
-4.996	$3.3959655... \times 10^{-10}$
⋮	⋮
2	52.7619554...
⋮	⋮
5	$1.7869238... \times 10^{-3}$

3.3.2 Modified Euler's Method

$$h = 10^{-3}$$

t	$\lvert y(t) - w(t) \rvert$
-5	0
-4.999	$3.6380135... \times 10^{-12}$
-4.998	$7.2837916... \times 10^{-12}$
-4.997	$1.0937410... \times 10^{-11}$
-4.996	$1.4598836... \times 10^{-11}$
⋮	⋮
2	0.0696702...
⋮	⋮
5	$8.8661656... \times 10^{-6}$

3.3.3 Richardson's First Extrapolation of Euler's Method

$$h = 10^{-3}$$

t	$\lvert y(t) - w(t)\rvert$
-5	0
-4.999	$2.3796768...\times10^{-5}$
-4.998	$2.3813756...\times10^{-5}$
-4.997	$2.3830759...\times10^{-5}$
-4.996	$2.3847776...\times10^{-5}$
⋮	⋮
2	19.11779401...
⋮	⋮
5	$2.2906410...\times10^{-4}$

3.4 Application of Richardson's Extrapolation

This section contains the data when applying Richardson's extrapolation to different values of h. (**Figure 1**) Examines the differences in approximations between Richardson's first extrapolation of Euler's method and Modified Euler's method. (**Figure 2**) Shows the differences between Richardson's first extrapolation of Euler's method and the actual solution. As you can see from (**Figure 2**), Richardson's Extrapolation improves the approximation to the actual solution significantly. For $h = 10^{-3}$ we see our approximation increase from ≈ 47.24 to ≈ 80.88 with an exact solution of 100. Thus, we have increased our accuracy by 72% with only one iteration of the extrapolation process.

h	$y(2,h)$	$y(2,\frac{h}{2})$	$y^\star(2,h)$	$w(2) - y^\star(2,h)$
10^{-3}	47.238044551925682	64.060125268616460	80.882205985307237	19.048123733700848
10^{-4}	89.874538668810544	94.664565960877496	99.454593252944449	0.544708873170734
10^{-5}	98.884852889637685	99.439269399770126	99.993685909902567	0.006307112317188

Figure 1: This table contains the different step sizes h, the Euler approximations $y(2,h)$, the Euler approximations with the intervals halved $y(2,\frac{h}{2})$, the respective first Richardson Extrapolation $y^\star(2,h)$, and the difference between the Modified Euler approximations with step size h and the first Richardson Extrapolation.

h	$y(2) - y^\star(2,h)$
10^{-3}	19.117794014692763
10^{-4}	0.545406747055551
10^{-5}	0.006314090097433

Figure 2: This table contains the differences between the actual solution at $t = 2$ and the first Richardson Extrapolation.

4 Citations

(1) Page 261,"Numerical Analysis" Richard L. Burden, Douglas J. Faires, Annette M. Burden, 10^{th} edition
(2) Page 261,"Numerical Analysis" Richard L. Burden, Douglas J. Faires, Annette M. Burden, 10^{th} edition
(3) Page 263,"Numerical Analysis" Richard L. Burden, Douglas J. Faires, Annette M. Burden, 10^{th} edition
(4) Page 270,"Numerical Analysis" Richard L. Burden, Douglas J. Faires, Annette M. Burden, 10^{th} edition