

Math 337 - Elementary Differential Equations

Lecture Notes – Numerical Methods for Differential Equations

Joseph M. Mahaffy,
(jmahaffy@sdsu.edu)

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

<http://jmahaffy.sdsu.edu>

Spring 2022

Outline

- 1 Introduction
- 2 Euler's Method
 - Malthusian Growth Example
 - Euler's Method - MatLab
 - Example with $f(t, y)$
 - Euler Error Analysis
- 3 Improved Euler's Method
 - Improved Euler's Method - Algorithm
 - Example
 - Improved Euler's Method Error
 - Order of Error

Introduction

Introduction

- Most differential equations can **not** be solved exactly
- Use the definition of the derivative to create a **difference equation**
- Develop numerical methods to solve differential equations
 - **Euler's Method**
 - **Improved Euler's Method**

Euler's Method

1

Initial Value Problem: Consider

$$\frac{dy}{dt} = f(t, y) \quad \text{with} \quad y(t_0) = y_0$$

- From the definition of the derivative

$$\frac{dy}{dt} = \lim_{h \rightarrow 0} \frac{y(t+h) - y(t)}{h}$$

- Instead of taking the limit, fix h , so

$$\frac{dy}{dt} \approx \frac{y(t+h) - y(t)}{h}$$

- Substitute into the differential equation and with algebra write

$$y(t+h) \approx y(t) + hf(t, y)$$

Euler's Method

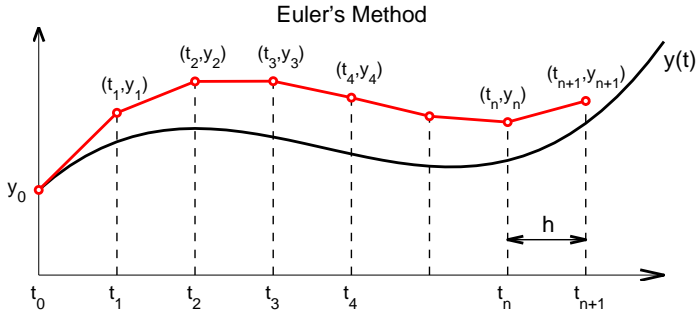
Euler's Method for a fixed h is

$$y(t + h) = y(t) + hf(t, y)$$

- Geometrically, Euler's method looks at the slope of the tangent line
 - The approximate solution follows the tangent line for a time step h
 - Repeat this process at each time step to obtain an approximation to the solution
- The ability of this method to track the solution accurately depends on the length of the time step, h , and the nature of the function $f(t, y)$
- This technique is rarely used as it has very bad convergence properties to the actual solution

Euler's Method

Graph of Euler's Method



Euler's Method

Euler's Method Formula: Euler's method is just a discrete dynamical system for approximating the solution of a continuous model

- Let $t_{n+1} = t_n + h$
- Define $y_n = y(t_n)$
- The initial condition gives $y(t_0) = y_0$
- **Euler's Method** is the discrete dynamical system

$$y_{n+1} = y_n + h f(t_n, y_n)$$

- Euler's Method only needs the initial condition to start and the right hand side of the differential equation (the **slope field**), $f(t, y)$ to obtain the approximate solution

Malthusian Growth Example

1

Malthusian Growth Example: Consider the model

$$\frac{dP}{dt} = 0.2P \quad \text{with} \quad P(0) = 50$$

Find the exact solution and approximate the solution with Euler's Method for $t \in [0, 1]$ with $h = 0.1$

Solution: The exact solution is

$$P(t) = 50e^{0.2t}$$

Malthusian Growth Example

2

Solution (cont): The **Formula for Euler's Method** is

$$P_{n+1} = P_n + h 0.2 P_n$$

The initial condition $P(0) = 50$ implies that $t_0 = 0$ and $P_0 = 50$

Create a table for the Euler iterates

t_n	P_n
$t_0 = 0$	$P_0 = 50$
$t_1 = t_0 + h = 0.1$	$P_1 = P_0 + 0.1(0.2P_0) = 50 + 1 = 51$
$t_2 = t_1 + h = 0.2$	$P_2 = P_1 + 0.1(0.2P_1) = 51 + 1.02 = 52.02$
$t_3 = t_2 + h = 0.3$	$P_3 = P_2 + 0.1(0.2P_2) = 52.02 + 1.0404 = 53.0604$

Malthusian Growth Example

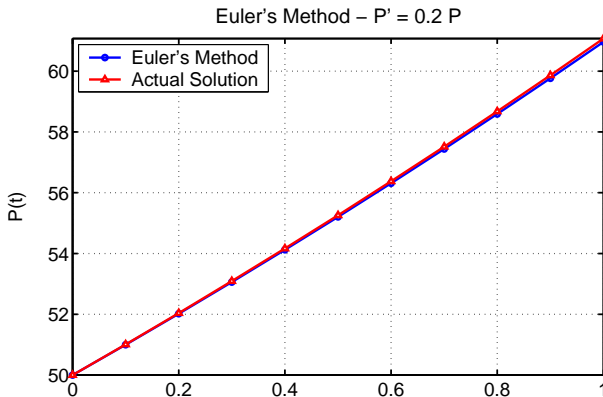
Solution (cont): Iterations are easily continued - Below is table of the actual solution and the Euler's method iterates

t	Euler Solution	Actual Solution
0	50	50
0.1	51	51.01
0.2	52.02	52.041
0.3	53.060	53.092
0.4	54.122	54.164
0.5	55.204	55.259
0.6	56.308	56.375
0.7	57.434	57.514
0.8	58.583	58.676
0.9	59.755	59.861
1.0	60.950	61.070

Malthusian Growth Example

4

Graph of Euler's Method for Malthusian Growth Example



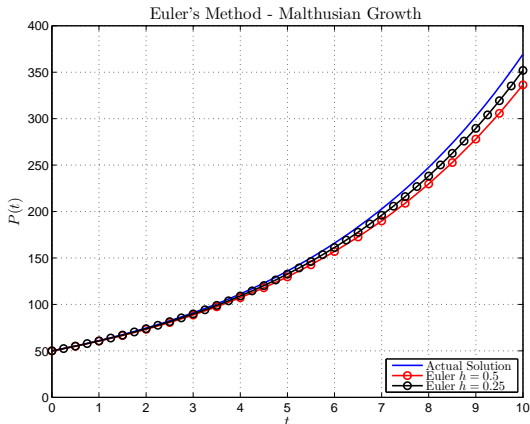
Malthusian Growth Example

Error Analysis and Larger Stepsize

- The table and the graph shows that Euler's method is tracking the solution fairly well over the interval of the simulation
- The error at $t = 1$ is only -0.2%
- However, this is a fairly short period of time and the stepsize is relatively small
- What happens when the stepsize is increased and the interval of time being considered is larger?

Malthusian Growth Example

Graph of Euler's Method with $h = 0.5$ and $h = 0.25$



There is a -9% error in the numerical solution at $t = 10$ for $h = 0.5$, and a -4.7% error when $h = 0.25$

Euler's Method - Algorithm

Algorithm (Euler's Method)

Consider the initial value problem

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

Let h be a fixed stepsize and define $t_n = t_0 + nh$. Also, let $y(t_n) = y_n$. **Euler's Method** for approximating the solution to the IVP satisfies the **difference equation**

$$y_{n+1} = y_n + hf(t_n, y_n).$$

Euler's Method - MatLab

Define a MatLab function for Euler's method for any function (func) with stepsize h , $t \in [t_0, t_f]$, and $y(t_0) = y_0$

```
1 function [t,y] = euler(func,h,t0,tf,y0)
2 % Euler's Method - Step size h, time from t0 to tf, initial
   y is y0
3
4 % Create time interval and initialize y
5 t = [t0:h:tf];
6 y(1) = y0;
7
8 % Loop for Euler's method
9 for i = 1:length(t)-1
10     y(i+1) = y(i) + h*(feval(func,t(i),y(i)));
11 end
12
13 % Create column vectors t and y
14 t = t';
15 y = y';
16
17 end
```

Euler's Method - Population

Our initial example was $\frac{dP}{dt} = 0.2P$ with $P(0) = 50$

```
1 function z = pop(t, y)
2 % Malthusian growth
3 z = 0.2*y;
4 end
```

Create graph shown above

```
1 tt = linspace(0,10,200);
2 yy = 50*exp(0.2*tt); % Actual solution
3 [t,y]=euler(@pop,0.5,0,10,50); % Implement Euler's method, 0.5
4 [t1,y1]=euler(@pop,0.25,0,10,50); % Implement Euler's method, 0.25
5 plot(tt,yy,'b-', 'LineWidth',1.5); % Actual solution
6 hold on % Plots Multiple graphs
7 plot(t,y,'r-o', 'LineWidth',1.5, 'MarkerSize',7); % Euler h = 0.5
8 plot(t1,y1,'k-o', 'LineWidth',1.5, 'MarkerSize',7); % Euler h = 0.25
9 grid % Adds Gridlines
10 h = legend('Actual Solution', 'Euler $h = 0.5$', 'Euler $h = 0.25$',4)
;
11 set(h, 'Interpreter', 'latex') % Allow LaTeX in legend
12 axis([0 10 0 400]); % Defines limits of graph
```


Euler's Method with $f(t, y)$

1

Euler's Method with $f(t, y)$: Consider the model

$$\frac{dy}{dt} = y + t \quad \text{with} \quad y(0) = 3$$

Find the solution to this initial value problem

Rewrite this linear DE and find the integrating factor:

$$\frac{dy}{dt} - y = t \quad \text{with} \quad \mu(t) = e^{-t}$$

Solving

$$\frac{d}{dt} (e^{-t}y) = te^{-t} \quad \text{or} \quad e^{-t}y(t) = \int te^{-t} dt = -(t+1)e^{-t} + C$$

With the initial condition the solution is

$$y(t) = 4e^t - t - 1$$

Euler's Method with $f(t, y)$

2

Solution (cont): Euler's formula with $h = 0.25$ is

$$y_{n+1} = y_n + 0.25(y_n + t_n)$$

t_n	Euler solution y_n
$t_0 = 0$	$y_0 = 3$
$t_1 = 0.25$	$y_1 = y_0 + h(y_0 + t_0) = 3 + 0.25(3 + 0) = 3.75$
$t_2 = 0.5$	$y_2 = y_1 + h(y_1 + t_1) = 3.75 + 0.25(3.75 + 0.25) = 4.75$
$t_3 = 0.75$	$y_3 = y_2 + h(y_2 + t_2) = 4.75 + 0.25(4.75 + 0.5) = 6.0624$
$t_4 = 1$	$y_4 = y_3 + h(y_3 + t_3) = 6.0624 + 0.25(6.0624 + 0.75) = 7.7656$

Actual solution is $y(1) = 8.8731$, so the Euler solution has a -12.5% error

If $h = 0.1$, after 10 steps $y(1) \approx y_{10} = 8.3750$ with -5.6% error

Euler's Method with $f(t, y)$

3

Solution (cont): Euler's formula with different h is

$$y_{n+1} = y_n + h(y_n + t_n)$$

t_n	$h = 0.2$	$h = 0.1$	$h = 0.05$	$h = 0.025$	Actual
0.2	3.6	3.64	3.662	3.6736	3.6856
0.4	4.36	4.4564	4.5098	4.538	4.5673
0.6	5.312	5.4862	5.5834	5.6349	5.6885
0.8	6.4944	6.7744	6.9315	7.015	7.1022
1	7.9533	8.375	8.6132	8.7403	8.8731
2	21.7669	23.91	25.16	25.8383	26.5562
% Err	-18.0	-9.96	-5.26	-2.70	

We see the percent error at $t = 2$ (compared to the actual solution) declining by about $\frac{1}{2}$ as h is halved

Euler Error Analysis

- Consider the solution of the IVP $y' = f(t, y)$, $y(t_0) = y_0$ denoted $\phi(t)$
 - **Euler's formula**, $y_{n+1} = y_n + hf(t_n, y_n)$, approximates $y_n \approx \phi(t_n)$
 - Expect the **error** to decrease as h decreases
 - How small does h have to be to reach a certain tolerance?
- Errors
 - **Local truncation error**, e_n , is the amount of error at each step
 - **Global truncation error**, E_n , is the amount of error between the algorithm and $\phi(t)$
 - **Round-off error**, R_n , is the error due to the fact that computers hold finite digits

Local Truncation Error

1

Assume that $\phi(t)$ solves the IVP, so

$$\phi'(t) = f(t, \phi(t))$$

Use Taylor's theorem with a remainder, then

$$\phi(t_n + h) = \phi(t_n) + \phi'(t_n)h + \frac{1}{2}\phi''(\bar{t}_n)h^2,$$

where $\bar{t}_n \in (t_n, t_n + h)$

From ϕ being a solution of the IVP

$$\phi(t_{n+1}) = \phi(t_n) + hf(t_n, \phi(t_n)) + \frac{1}{2}\phi''(\bar{t}_n)h^2,$$

If $y_n = \phi(t_n)$ is the correct solution, then the **Euler approximate solution** at t_{n+1} is

$$y_{n+1}^* = \phi(t_n) + hf(t_n, \phi(t_n)),$$

so the **local truncation error** satisfies

$$e_{n+1} = \phi(t_{n+1}) - y_{n+1}^* = \frac{1}{2}\phi''(\bar{t}_n)h^2$$

Local Truncation Error

Since the **local truncation error** satisfies

$$e_{n+1} = \frac{1}{2}\phi''(\bar{t}_n)h^2,$$

then if there is a **uniform bound** $M = \max_{t \in [a, b]} |\phi''(t)|$, the local error is bounded with

$$|e_n| \leq \frac{Mh^2}{2}$$

Thus, **Euler's Method** is said to have a **local truncation error of order** h^2 often denoted $\mathcal{O}(h^2)$

This result allows the choice of a stepsize to keep the numerical solution within a certain tolerance, say ε , or

$$\frac{Mh^2}{2} \leq \varepsilon \quad \text{or} \quad h \leq \sqrt{2\varepsilon/M}$$

Often difficult to estimate either $|\phi''(t)|$ or M

Global Truncation

Other Errors

- The **local truncation error** satisfies $|e_n| \leq Mh^2/2$
 - This error is most significant for **adaptive numerical routines** where code is created to maintain a certain tolerance
- **Global Truncation Error**
 - The more important error for the numerical routines is this error over the entire simulation
 - **Euler's method** can be shown to have a **global truncation error**,

$$|E_n| \leq Kh$$

- Note error is one order less than **local error**, which scales proportionally with the stepsize or $|E_n| \leq \mathcal{O}(h)$
- HW problem using Taylor's series and Math induction to prove this result

Global Truncation and Round-Off Error

Other Errors - continued

- **Round-Off Error**, R_n

- This error results from the finite digits in the computer
- All numbers in a computer are truncated
- This is beyond the scope of this course

- **Total Computed Error**

- The total error combines the machine error and the error of the algorithm employed
- It follows that

$$|\phi(t_n) - Y_n| \leq |E_n| + |R_n|$$

- The machine error cannot be controlled, but choosing a **higher order method** allows improving the **global truncation error**

Numerical solutions of DEs

Numerical solutions of differential equations

- Euler's Method is simple and intuitive, but lacks accuracy
- Numerical methods are available through standard software
 - MatLab's `ode23`
 - Maple's `dsolve` with *numeric* option
- Many types of numerical methods - different accuracies and stability
 - Easiest are **single stepsize Runge-Kutta methods**
 - Software above uses **adaptive stepsize Runge-Kutta methods**
 - Many other techniques shown in Math 542
- **Improved Euler's method** (or **Heun formula**) is a simple extension of Euler's method - However, significantly better

Improved Euler's Method - Algorithm

Algorithm (Improved Euler's Method (or Heun Formula))

Consider the initial value problem

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

Let h be a fixed stepsize. Define $t_n = t_0 + nh$ and the approximate solution $y(t_n) = y_n$.

- 1 Approximate y by **Euler's Method**

$$ye_n = y_n + hf(t_n, y_n)$$

- 2 **Improved Euler's Method** is the difference formula

$$y_{n+1} = y_n + \frac{h}{2} (f(t_n, y_n) + f(t_n + h, ye_n))$$

Improved Euler's Method

Improved Euler's Method Formula: This technique is an easy extension of Euler's Method

- The Improved Euler's method uses an average of the Euler's method and an Euler's method approximation to the function
- This technique requires two function evaluations, instead of one
- Simple two step algorithm for implementation
- Can show this converges as $\mathcal{O}(h^2)$, which is significantly better than Euler's method

Improved Euler's Method - MatLab

Define a MatLab function for the Improved Euler's method for any function (func) with stepsize h , $t \in [t_0, t_f]$, and $y(t_0) = y_0$

```
1 function [t,y] = im_euler(func,h,t0,tf,y0)
2 % Improved Euler's Method - Stepsize h, time from t0 to tf,
   % initial y is y0
3 % Create time interval and initialize y
4 t = [t0:h:tf];
5 y(1) = y0;
6 % Loop for Improved Euler's method
7 for i = 1:length(t)-1
8     ye = y(i) + h*(feval(func,t(i),y(i))); % Euler's step
9     y(i+1) = y(i) + (h/2)*(feval(func,t(i),y(i)) + feval(
        func,t(i+1),ye));
10 end
11 % Create column vectors t and y
12 t = t';
13 y = y';
14 end
```

Example: Improved Euler's Method

1

Example: Improved Euler's Method: Consider the initial value problem:

$$\frac{dy}{dt} = y + t \quad \text{with} \quad y(0) = 3$$

- The solution to this differential equation is

$$y(t) = 4e^t - t - 1$$

- Numerically solve this using Euler's Method and Improved Euler's Method using $h = 0.1$
- Compare these numerical solutions

Example: Improved Euler's Method

2

Solution: Let $y_0 = 3$, the Euler's formula is

$$y_{n+1} = y_n + h(y_n + t_n) = y_n + 0.1(y_n + t_n)$$

The Improved Euler's formula is

$$ye_n = y_n + h(y_n + t_n) = y_n + 0.1(y_n + t_n)$$

with

$$y_{n+1} = y_n + \frac{h}{2} ((y_n + t_n) + (ye_n + t_n + h))$$

$$y_{n+1} = y_n + 0.05 (y_n + ye_n + 2t_n + 0.1)$$

Example: Improved Euler's Method

3

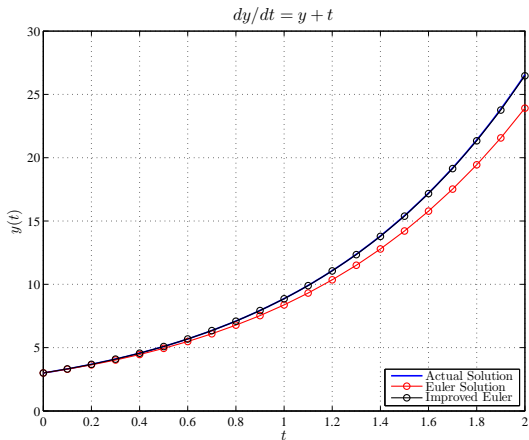
Solution: Below is a table of the numerical computations

t	Euler's Method	Improved Euler	Actual
0	$y_0 = 3$	$y_0 = 3$	$y(0) = 3$
0.1	$y_1 = 3.3$	$y_1 = 3.32$	$y(0.1) = 3.3207$
0.2	$y_2 = 3.64$	$y_2 = 3.6841$	$y(0.2) = 3.6856$
0.3	$y_3 = 4.024$	$y_3 = 4.0969$	$y(0.3) = 4.0994$
0.4	$y_4 = 4.4564$	$y_4 = 4.5636$	$y(0.4) = 4.5673$
0.5	$y_5 = 4.9420$	$y_5 = 5.0898$	$y(0.5) = 5.0949$
0.6	$y_6 = 5.4862$	$y_6 = 5.6817$	$y(0.6) = 5.6885$
0.7	$y_7 = 6.0949$	$y_7 = 6.3463$	$y(0.7) = 6.3550$
0.8	$y_8 = 6.7744$	$y_8 = 7.0912$	$y(0.8) = 7.1022$
0.9	$y_9 = 7.5318$	$y_9 = 7.9247$	$y(0.9) = 7.9384$
1	$y_{10} = 8.3750$	$y_{10} = 8.8563$	$y(1) = 8.8731$

Example: Improved Euler's Method

4

Graph of Solution: Actual, Euler's and Improved Euler's



The Improved Euler's solution is very close to the actual solution



Example: Improved Euler's Method

Solution: Comparison of the numerical simulations

- It is very clear that the Improved Euler's method does a substantially better job of tracking the actual solution
- The Improved Euler's method requires only one additional function, $f(t, y)$, evaluation for this improved accuracy
- At $t = 1$, the Euler's method has a -5.6% error from the actual solution
- At $t = 1$, the Improved Euler's method has a -0.19% error from the actual solution

Improved Euler's Method Error

Improved Euler's Method Error

- Showed earlier that **Euler's method** had a **local truncation error** of $\mathcal{O}(h^2)$ with **global error** being $\mathcal{O}(h)$
- Similar **Taylor expansions** (in two variables) give the **local truncation error** for the **Improved Euler's method** as $\mathcal{O}(h^3)$
- For **Improved Euler's method**, the **global truncation error** is $\mathcal{O}(h^2)$
- From a practical perspective, these results imply:
 - With **Euler's method**, the reduction of the stepsize by a factor of 0.1 gains one digit of accuracy
 - With **Improved Euler's method**, the reduction of the stepsize by a factor of 0.1 gains two digits of accuracy
 - This is a **significant improvement** at only the cost of one additional function evaluation per step

Numerical Example

1

Numerical Example: Consider the IVP

$$\frac{dy}{dt} = 2e^{-0.1t} - \sin(y), \quad y(0) = 3,$$

which has no exact solution, so must solve numerically

- Solve this problem with **Euler's method** and **Improved Euler's method**
- Show differences with different stepsizes for $t \in [0, 5]$
- Show the order of convergence by halving the stepsize twice
- Graph the solution and compare to solution from *ode23* in MatLab, closely approximating the exact solution

Numerical Example

2

Numerical Solution for $\frac{dy}{dt} = 2e^{-0.1t} - \sin(y)$, $y(0) = 3$

Used MatLab's *ode45* to obtain an accurate numerical solution to compare **Euler's method** and **Improved Euler's method** with stepsizes $h = 0.2$, $h = 0.1$, and $h = 0.05$

	"Actual"	Euler	Im Eul	Euler	Im Eul	Euler	Im Eul
t_n		$h = 0.2$	$h = 0.2$	$h = 0.1$	$h = 0.1$	$h = 0.05$	$h = 0.05$
0	3	3	3	3	3	3	3
1	5.5415	5.4455	5.5206	5.4981	5.5361	5.5209	5.5401
2	7.1032	7.1718	7.0881	7.1368	7.0995	7.1199	7.1023
3	7.753	7.836	7.743	7.7939	7.7505	7.7734	7.7524
4	8.1774	8.2818	8.167	8.2288	8.1748	8.2029	8.1768
5	8.5941	8.7558	8.5774	8.6737	8.5899	8.6336	8.5931
		1.88%	-0.194%	0.926%	-0.0489%	0.460%	-0.0116%

Last row shows percent error between the different approximations and the accurate solution

Numerical Example

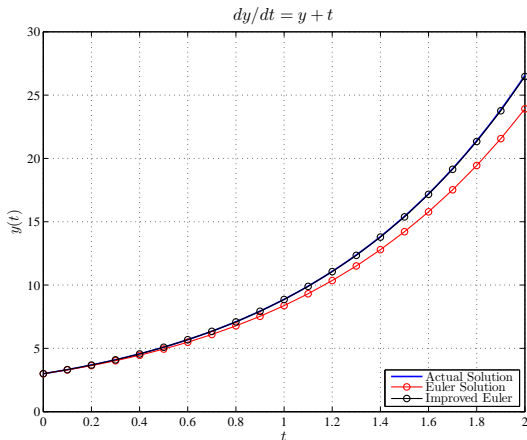
3

Error of Numerical Solutions

- Observe that the **Improved Euler's method** with stepsize $h = 0.2$ is more accurate at $t = 5$ than **Euler's method** with stepsize $h = 0.05$
- With **Euler's method** the error cuts in half with halving of the stepsize
- With the **Improved Euler's method** the errors cuts in quarter with halving of the stepsize

Numerical Example

Graph of Solution: Actual, Euler's and Improved Euler's methods with $h = 0.2$



The Improved Euler's solution is very close to the actual solution

Order of Error

Error of Numerical Solutions

- **Order of Error** without good “Actual solution”
 - Simulate system with stepsizes h , $h/2$, and $h/4$ and define these simulates as y_n^1 , y_n^2 , and y_n^3 , respectively
 - Compute the ratio (from Cauchy sequence)

$$R = \frac{|y_n^3 - y_n^2|}{|y_n^2 - y_n^1|}$$

- If the numerical method is **order** m , then this ratio is approximately $\frac{1}{2^m}$
- Above example at $t = 5$ has $R = 0.488$ for **Euler's method** and $R = 0.256$ for **Improved Euler's method**
- Allows user to determine how much error numerical routine is generating