

B1 Numerical Algorithms

4 Lectures, MT 2023

Lecture three – Computing Solutions of Ordinary Differential Equations.

Wes Armour

25th October 2023

Errors/Typos/Questions to: wes.armour@eng.ox.ac.uk

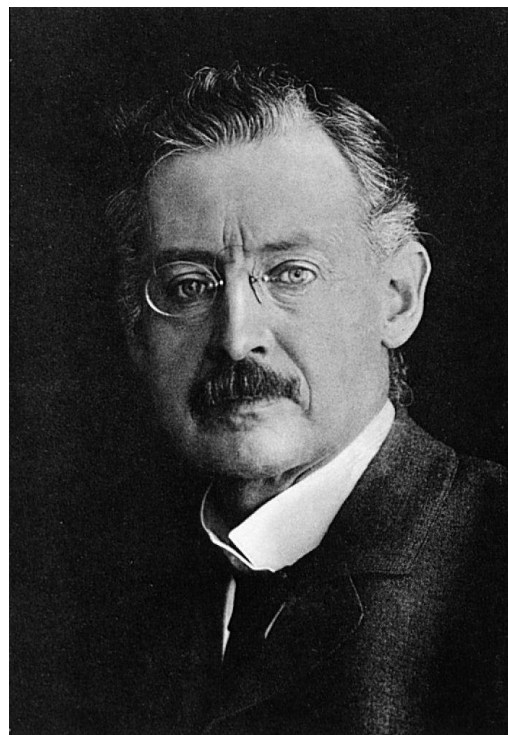
Learning outcomes

➤ Numerical integration of ODEs

- A new toy model
- Euler method,
- Modified Euler method,
- Runge-Kutta methods,
- adaptive step-size control (*Adams-Bashforth / Adams-Moulton*),
- Shooting method,
- Richardson's Method.



Leonhard Euler
1707 - 1783



Karl Runge
1856 - 1927



Martin Wilhelm Kutta
1867 - 1944

Solving Ordinary Differential Equations

Analysis of Engineering problems can often generate lots of differential equations, most of which cannot be easily solved explicitly. So we must learn to solve them numerically on a computer.

There is a close relationship between solving a finite integral using an approximation method and solving initial value problems as we have here.

This motivates the development of the techniques outlined in these slides.

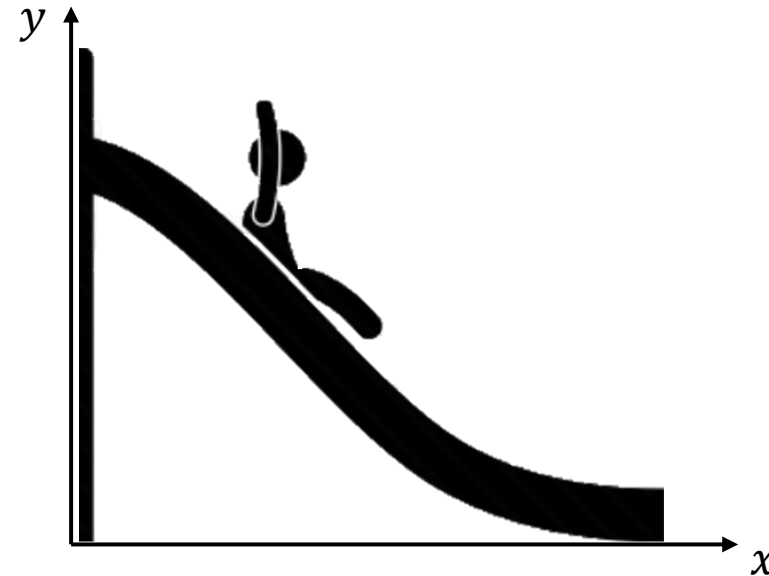
We conclude with the solution of boundary value problems.

A toy model



Harry is bored of the Ferris wheel, he now wants to play on a slide.

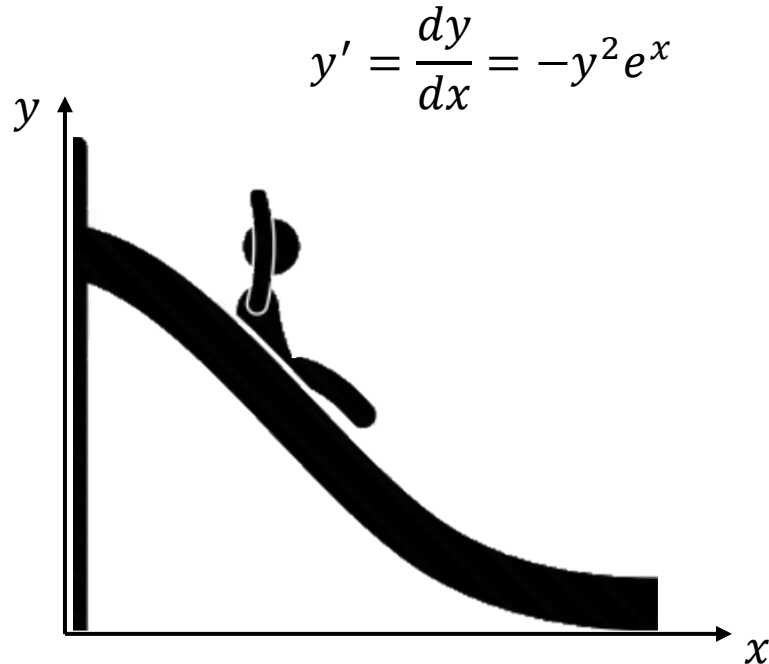
Imagine we want to know Harry's vertical displacement (y) at a position (x) from his starting position (x_0, y_0) on the slide.



** No children were harmed in the making of this slide deck*

First Order Equations

Let's imagine... **the only information** we have to solve our problem is Harry's **initial position** (x_0, y_0) and the following **differential equation** that describes the gradient along the curve of the slide:



We know how to solve this... we rearrange:

$$-\frac{dy(x)}{y^2} = e^x dx$$

And integrate both sides:

$$-\int \frac{dy(x)}{y^2} = \int e^x dx$$

$$\frac{1}{y} = e^x + C \quad \rightarrow$$

$$y(x) = \frac{1}{e^x + C}$$

General First Order Equations

Generally, in the simplest case, this involves only first derivatives of the unknown function y and the RHS is a function only of x

$$y' = \frac{dy}{dx} = f(x)$$

We know how to solve this:

$$dy(x) = f(x)dx$$

$$y(x) = \int f(x) dx$$

Quite simple.

What if we can't solve our ODE analytically?

For more complicated cases where we cannot find exact solution we turn to numerical solutions.

Lets look at three different methods:

- Euler Method
- Modified Euler Method
- Runge-Kutta (4th Order).

Euler Method

The Euler method relies on the forward difference approximation (presented in the first lecture) to calculate an approximate solution to an ordinary differential equation (ODE) with a given initial value.

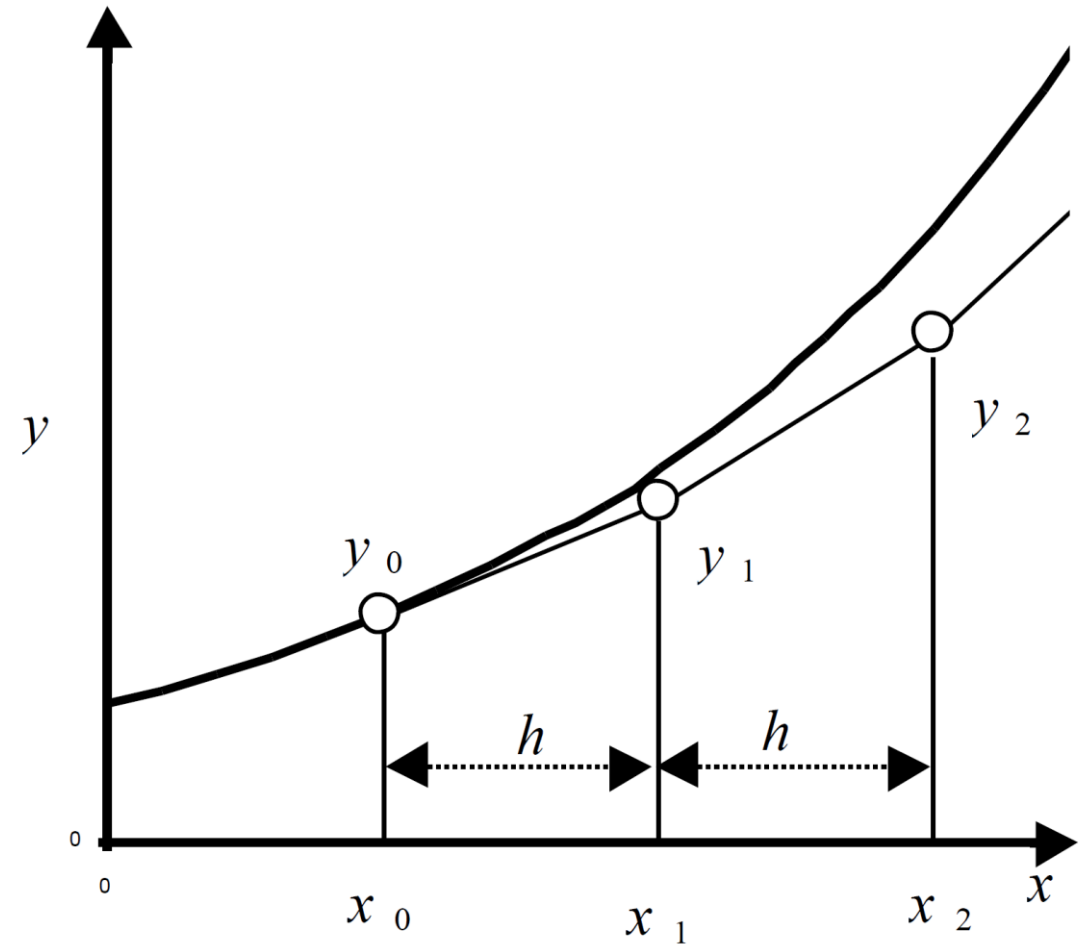
The method is sometimes referred to as the forward Euler method.

It is conceptually simple: Divide the region of interest $[a, b]$ up into discrete values of

$$x_n = nh,$$

$$n = 0, 1, \dots, N \text{ spaced at } h = \frac{b-a}{N}$$

(See image right)



Euler Method

Next we recall our forward difference approximation.

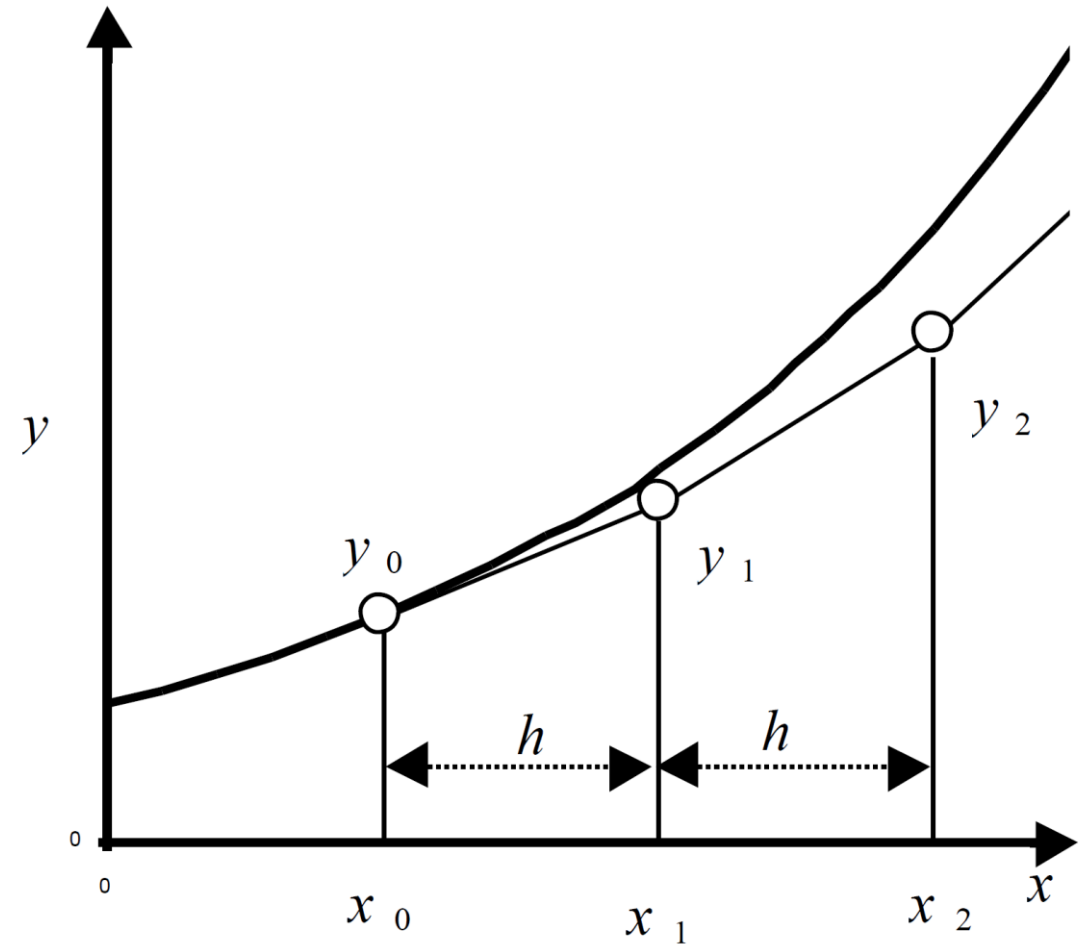
$$\frac{dy}{dx} = y'_n \approx \frac{y_{n+1} - y_n}{h} = \frac{y(nh + h) - y(nh)}{h}$$

And rearrange for y_{n+1}

$$y_{n+1} \approx y_n + h \frac{dy}{dx}$$

Given $y(nh) = y_n$ we use the above equation to step along the region of interest $x = nh$ ($n = 0, 1 \dots, N$).

Let's return to our slide example...



Euler Method

Harrys vertical displacement (y) at a position (x) is given by:

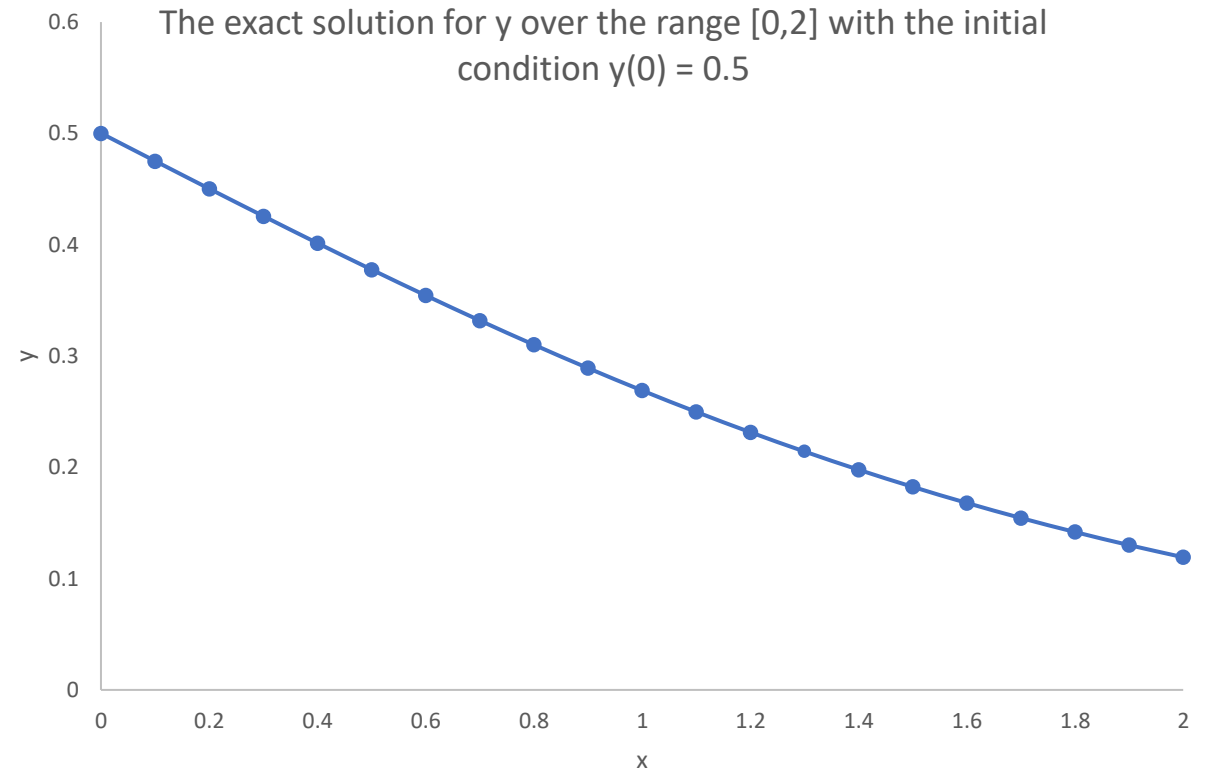
$$y' = \frac{dy}{dx} = -y^2 e^x$$

Which has analytic solution:

$$y(x) = \frac{1}{e^x + C}$$

Let's pretend we don't know an analytic solution for this. Let's choose our initial condition to be:

$$y(0) = y_0 = 0.5 \quad (\text{hence } C = 1).$$



We will use the exact solution (plotted above) to study the accuracy of the Euler Method.

Euler Method

Now using the Euler method with $h = 0.1$:

$$y_1 \approx y_0 + h \frac{dy_0}{dx_0} = y_0 - hy_0^2 e^{x_0}$$

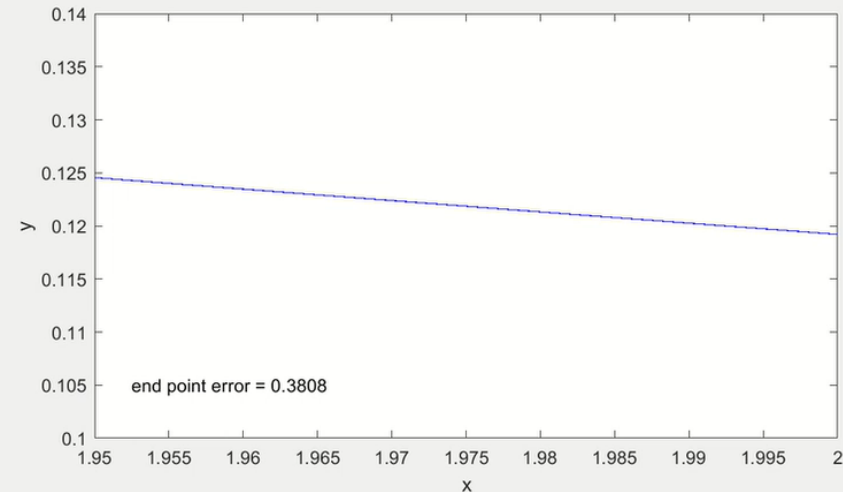
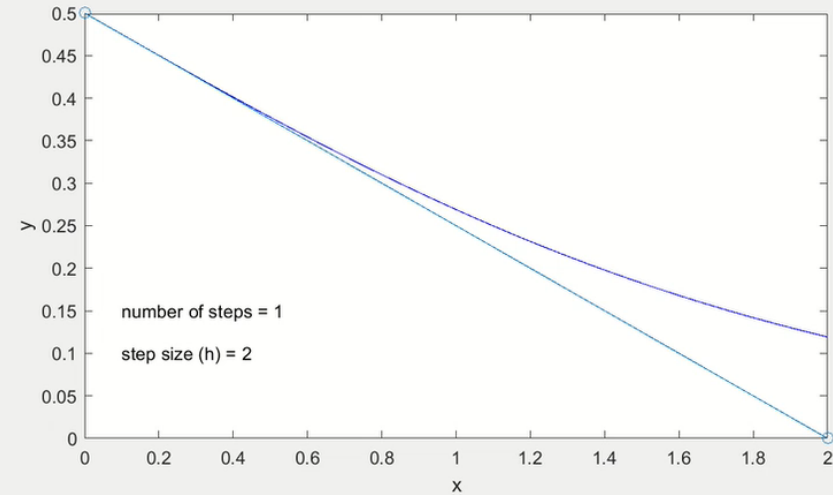
We have chosen $y_0 = 0.5$

$$y_1 = 0.5 - 0.1 * 0.5 * 0.5 * e^0 = 0.475$$

$$y_2 \approx y_1 + h \frac{dy_1}{dx_1} = y_1 - hy_1^2 e^{x_1}$$

And so on...

See results right.



Accuracy of the Euler Method

To quantify the accuracy of the Euler method we return to the Taylor series.

Expand in Taylor series:

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{2}y''_n + \dots = \overbrace{y_n + h \frac{dy_n}{dx}}^{y_{n+1}} + \underbrace{\frac{h^2}{2} \frac{d^2y_n}{dx^2} + \dots}_{\text{Error!}}$$

Thus the error per step (**Local Error**) is $O(h^2)$

However there are $\frac{L}{h} = \frac{b-a}{h}$ steps in the interval $x = x_0, \dots, x_N$ which is proportional to $\left(\frac{1}{h}\right)$ so the total error would be proportional to...

number of steps \times error at each step

hence...

The global (truncation) error in the Euler method is $O(h)$.

Part B – The modified Euler method

The Modified Euler Method

The modified Euler method, also known as the ***predictor-corrector method*** uses a better approximation of the slope from (x_n, y_n) to (x_{n+1}, y_{n+1}) .

Recall our previous Euler equation:

$$y_{n+1} \approx y_n + hf(x_n, y_n)$$

A better estimation of the slope from (x_n, y_n) to (x_{n+1}, y_{n+1}) would be:

$$y_{n+1} \approx y_n + \frac{h}{2} (f(x_n, y_n) + f(x_{n+1}, y_{n+1}))$$

For the keen eyed – you'll spot the problem... ***We don't know y_{n+1} !!***

The Modified Euler Method

However, we can evade this problem by using our original Euler method to estimate y_{n+1} . This gives a two stage ***predictor-corrector*** algorithm, the modified Euler method.

Lets see how this works...

- Step one gives an initial guess (estimate) of y_{n+1} .
- This guess of y_{n+1}^* is used to get an estimate of the average y' over the interval h .
- This is then used to correct the estimate of y_{n+1} in Step two.

Step one: The Predictor

$$y_{n+1}^* = y_n + hf(x_n, y_n)$$

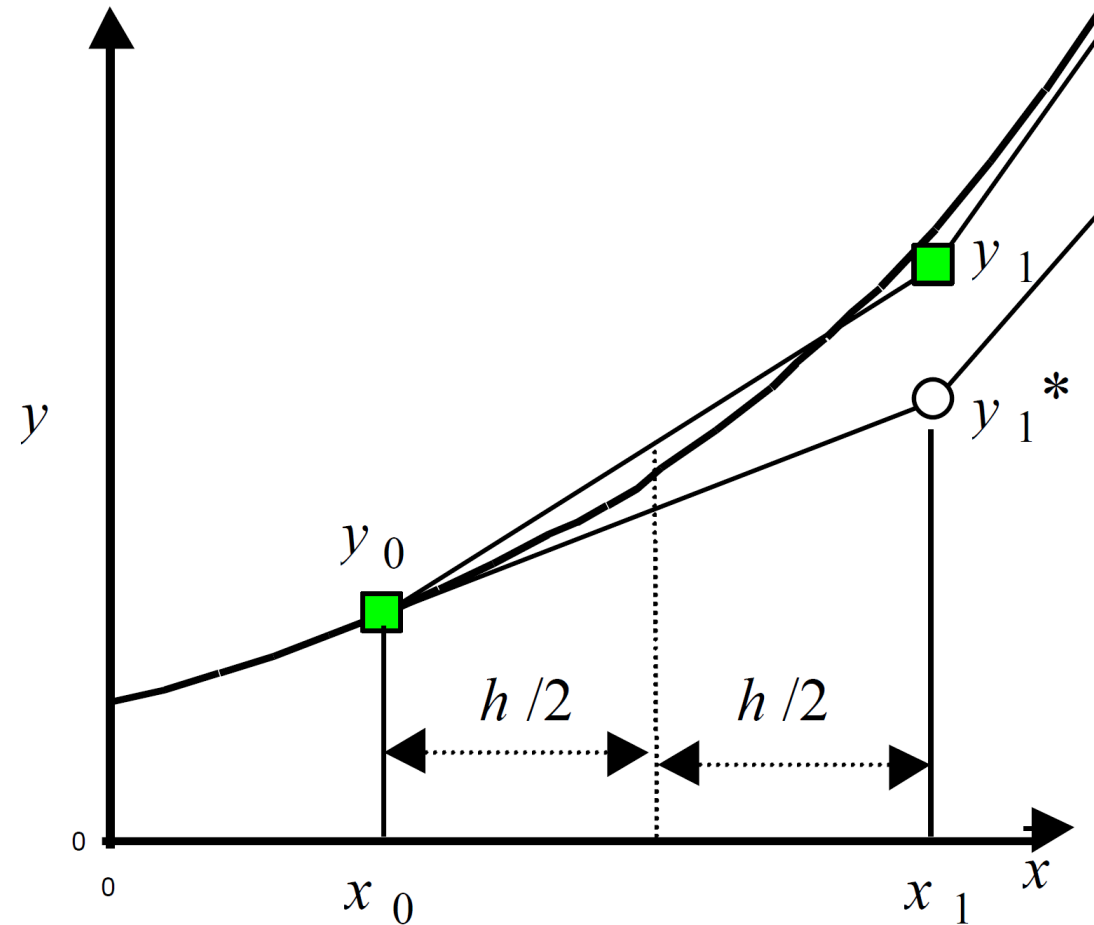
Step two: The Corrector (trapezoid rule!)

$$y_{n+1} = y_n + \frac{h}{2} (f(x_n, y_n) + f(x_{n+1}, y_{n+1}^*))$$

The Modified Euler Method

The image on the right will help you remember how the scheme works.

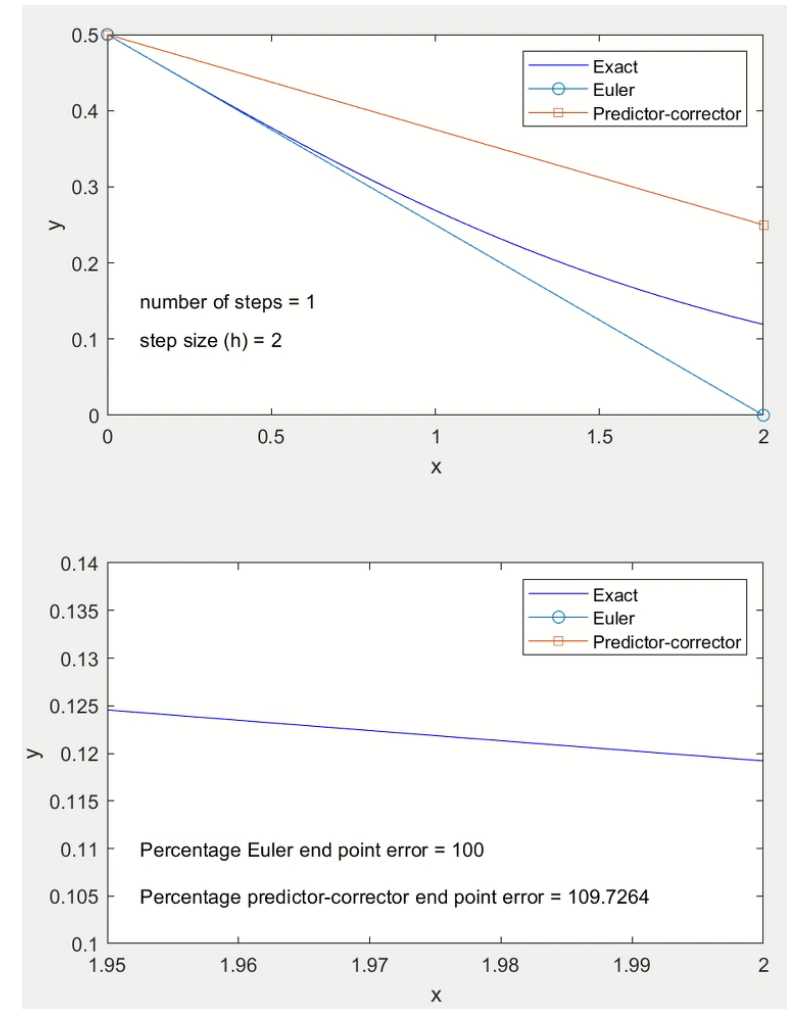
This is also sometimes called a Second-Order Runge-Kutta method. We will discover why later...



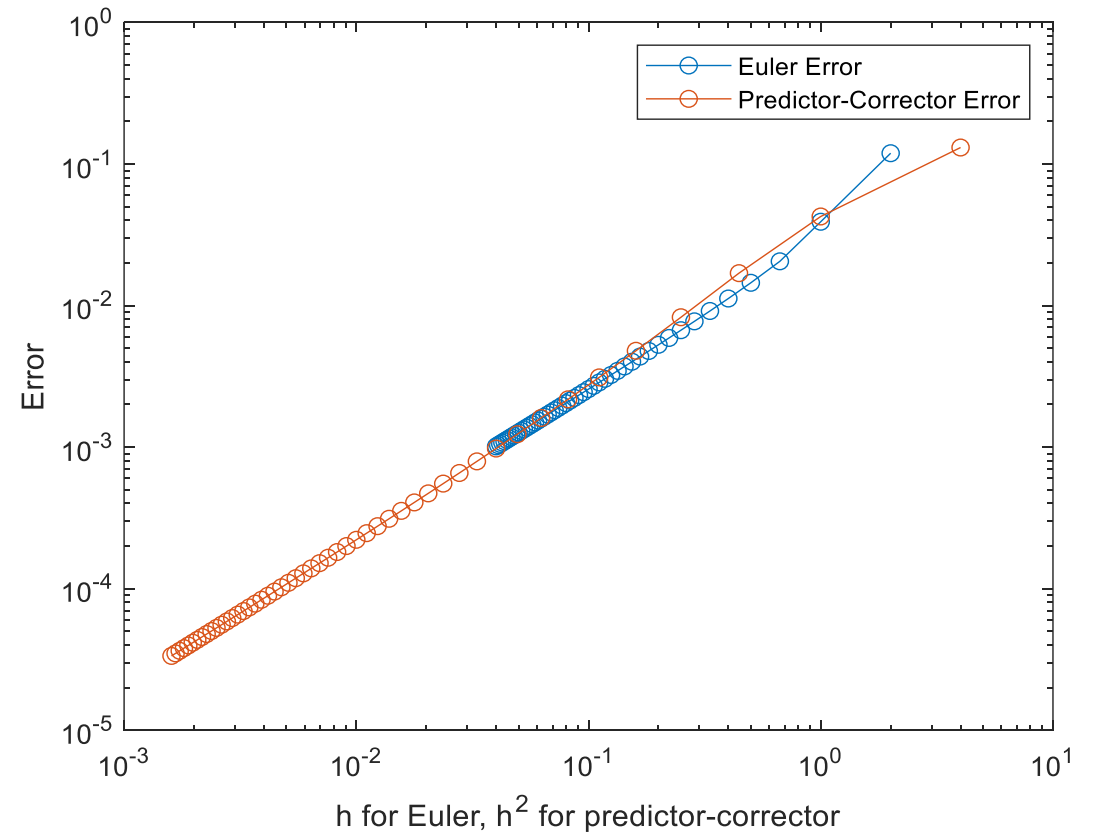
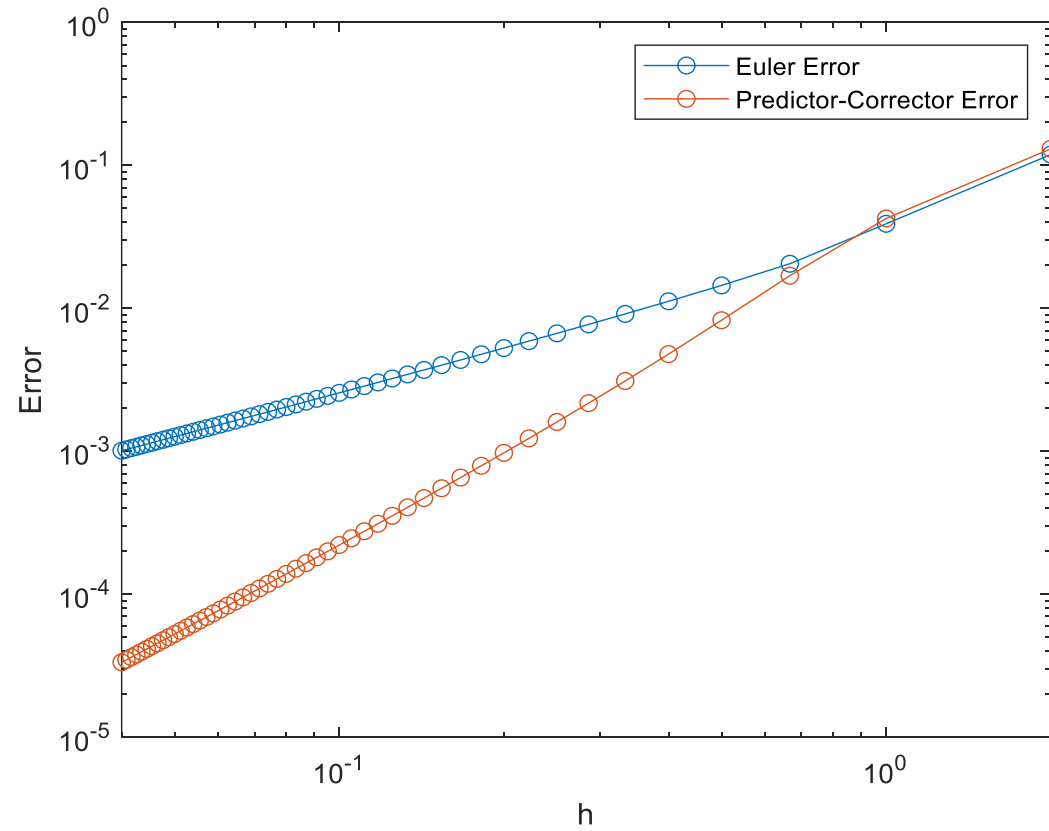
The Modified Euler Method

Let's now look at the accuracy of the predictor-corrector for our ODE example compared to the Euler method.

$$y' = \frac{dy}{dx} = -y^2 e^x$$



Errors of the Modified Euler Method



Accuracy of the Modified Euler Method

Panopto bonus video

B1 Numerical Algorithms

4 Lectures, MT 2022

Lecture three.

Bonus video 1 – The Accuracy of the Modified Euler Method

Wes Armour

Errors/typos/questions to: wes.armour@eng.ox.ac.uk



★ 00:20

Accuracy of the Modified Euler Method

Use the normal Taylor's Series method:
Define local error ϵ

$$y_{n+1} = y_n + \frac{h}{2} (f_n + f'_n) + \epsilon$$

Now, expanding f'_n as a two-dimensional Taylor Series in terms of $x_n + h$, $y_n + hf_n$

$$f'_n = f(x_n + h, y_n + hf_n) = f(x_n, y_n) + h \frac{\partial f}{\partial x} + hf_n \frac{\partial f}{\partial y} + O(h^2)$$

$$= f_n + h \frac{\partial f}{\partial x} + hf_n \frac{\partial f}{\partial y} + O(h^2)$$

Thus

$$y_{n+1} = y_n + \frac{h}{2} \left(f_n + f_n + h \frac{\partial f}{\partial x} + hf_n \frac{\partial f}{\partial y} + O(h^2) \right) + \epsilon$$

$$= y_n + hf_n + \frac{h^2}{2} \frac{\partial f}{\partial x} + \frac{h^2}{2} f_n \frac{\partial f}{\partial y} + O(h^3) + \epsilon$$

Homework...

★ 01:39

Accuracy of the Modified Euler Method.

Thus

$$y_{n+1} = y_n + \frac{h}{2} \left(f_n + f_n + h \frac{\partial f}{\partial x} + hf_n \frac{\partial f}{\partial y} + O(h^2) \right) + \epsilon$$

$$= y_n + hf_n + \frac{h^2}{2} \frac{\partial f}{\partial x} + \frac{h^2}{2} f_n \frac{\partial f}{\partial y} + O(h^3) + \epsilon$$

$$= y_n + hf_n + \frac{h^2}{2} \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} f_n \right) + O(h^3)$$

- It can be seen from this that the local, per step, error $\epsilon = O(h^3)$.
- Hence the global error for the modified Euler method is $O(h^2)$.

Homework...

★ 00:15

The end...



★ 00:13

Part C – Runge-Kutta and
other higher order methods

Runge-Kutta

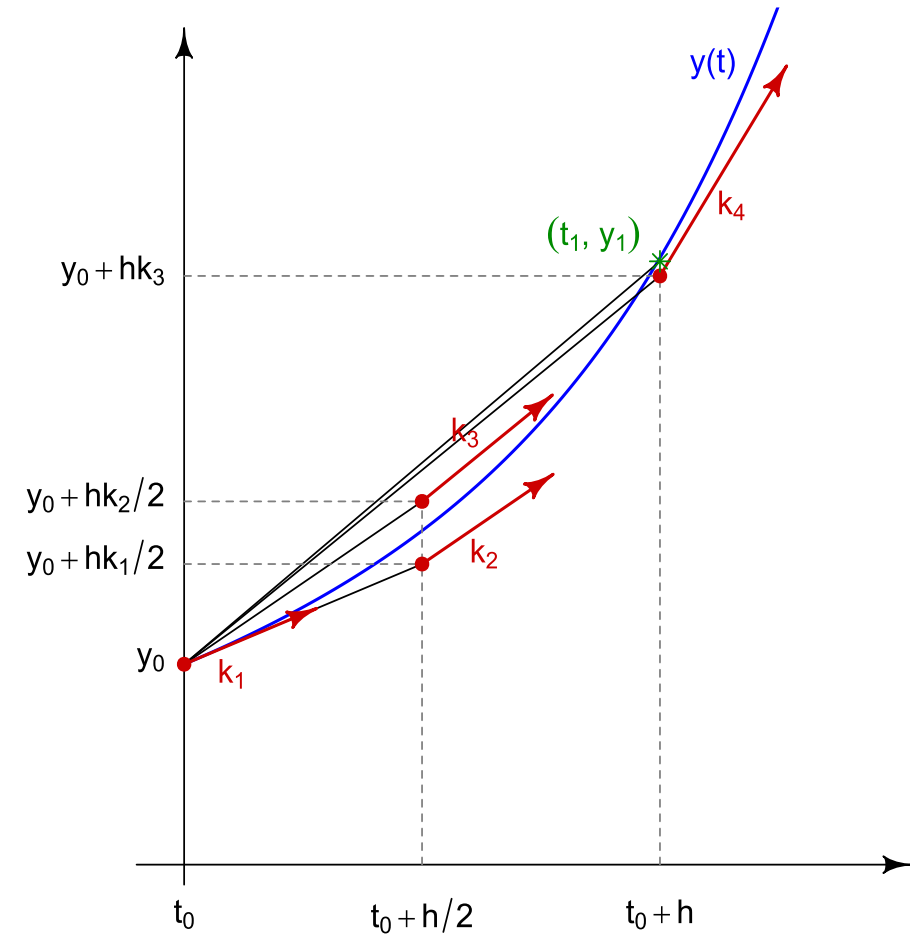
Runge-Kutta builds on the idea of the predictor-corrector method.

The classic method (sometimes called RK4) uses y_n and four weighted increments (see right) to predict y_{n+1}

It can be proved (although we won't do this) that the local error is $O(h^5)$ and so using the same arguments as for the Euler method the global truncation error is $O(h^4)$.

See previous slide: Accuracy of the Euler Method

See Kreyzig (pp. 1040, 7th Ed), HLT (pp. 29), Kiusalaas (pp. 247, 3rd Ed) or Wikipedia link below for more information.



Runge-Kutta

$$k_1 = hf(x_n, y_n)$$

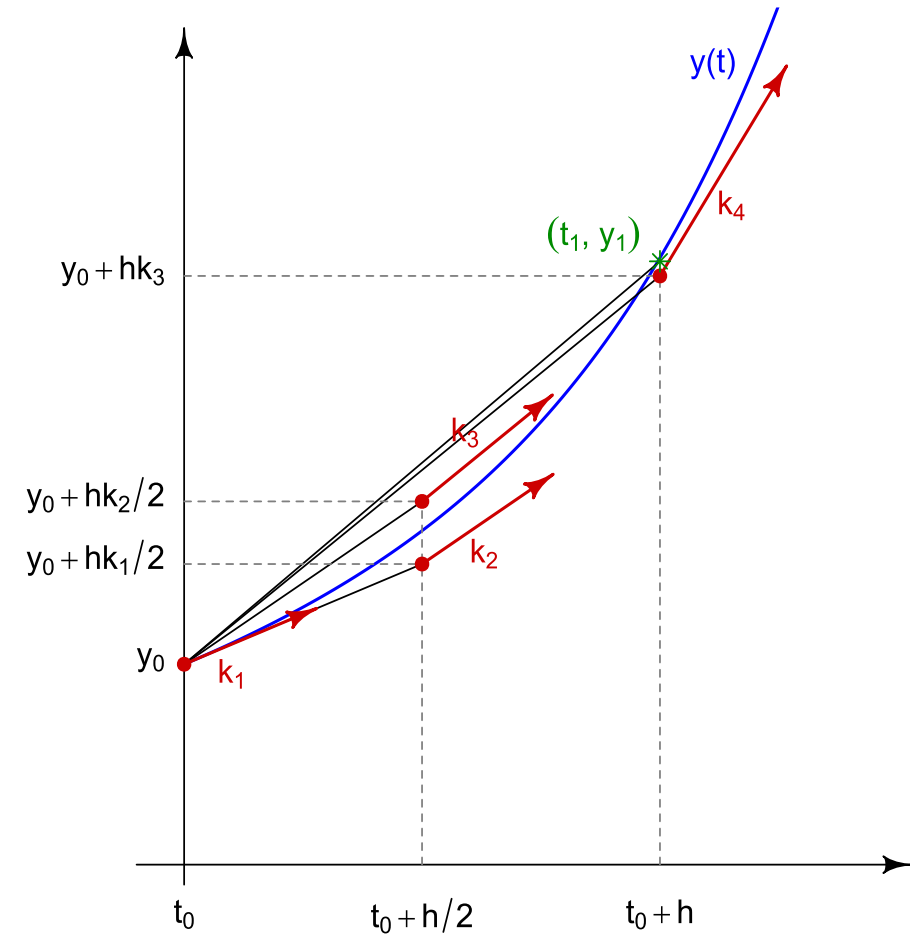
$$k_2 = hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right)$$

$$k_3 = hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right)$$

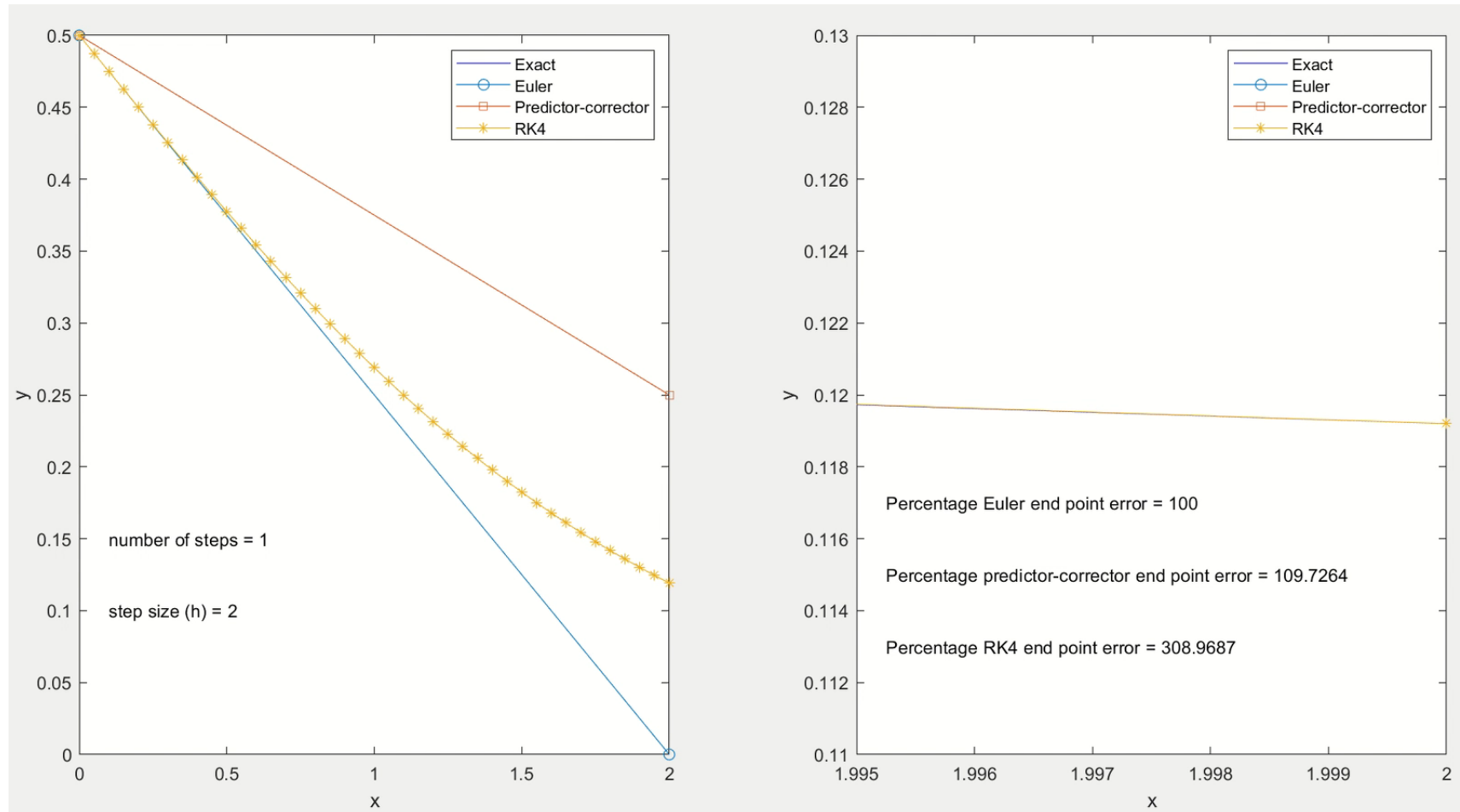
$$k_4 = hf(x_n + h, y_n + k_3)$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

A combination of predictive steps within the interval are used as a weighted average to make the final prediction for y_{n+1}

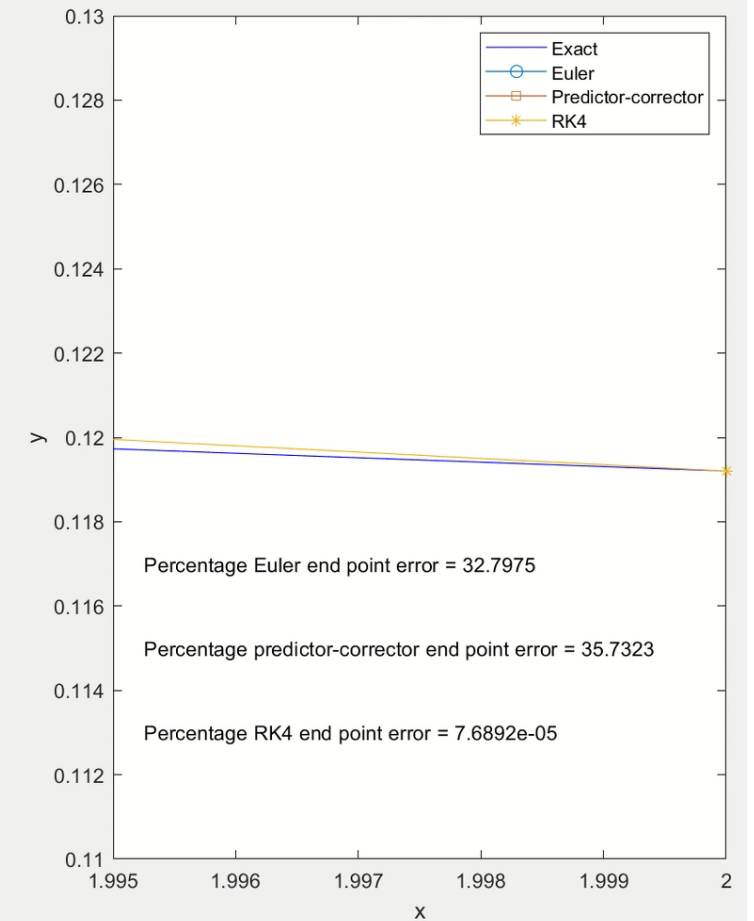
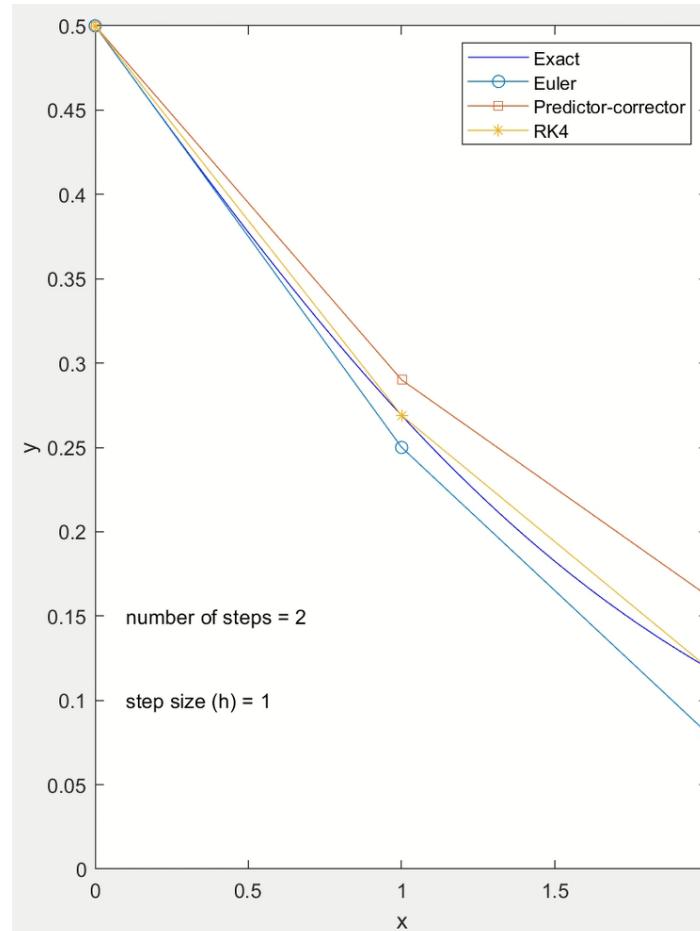


Comparison of the three methods



Comparison in more detail

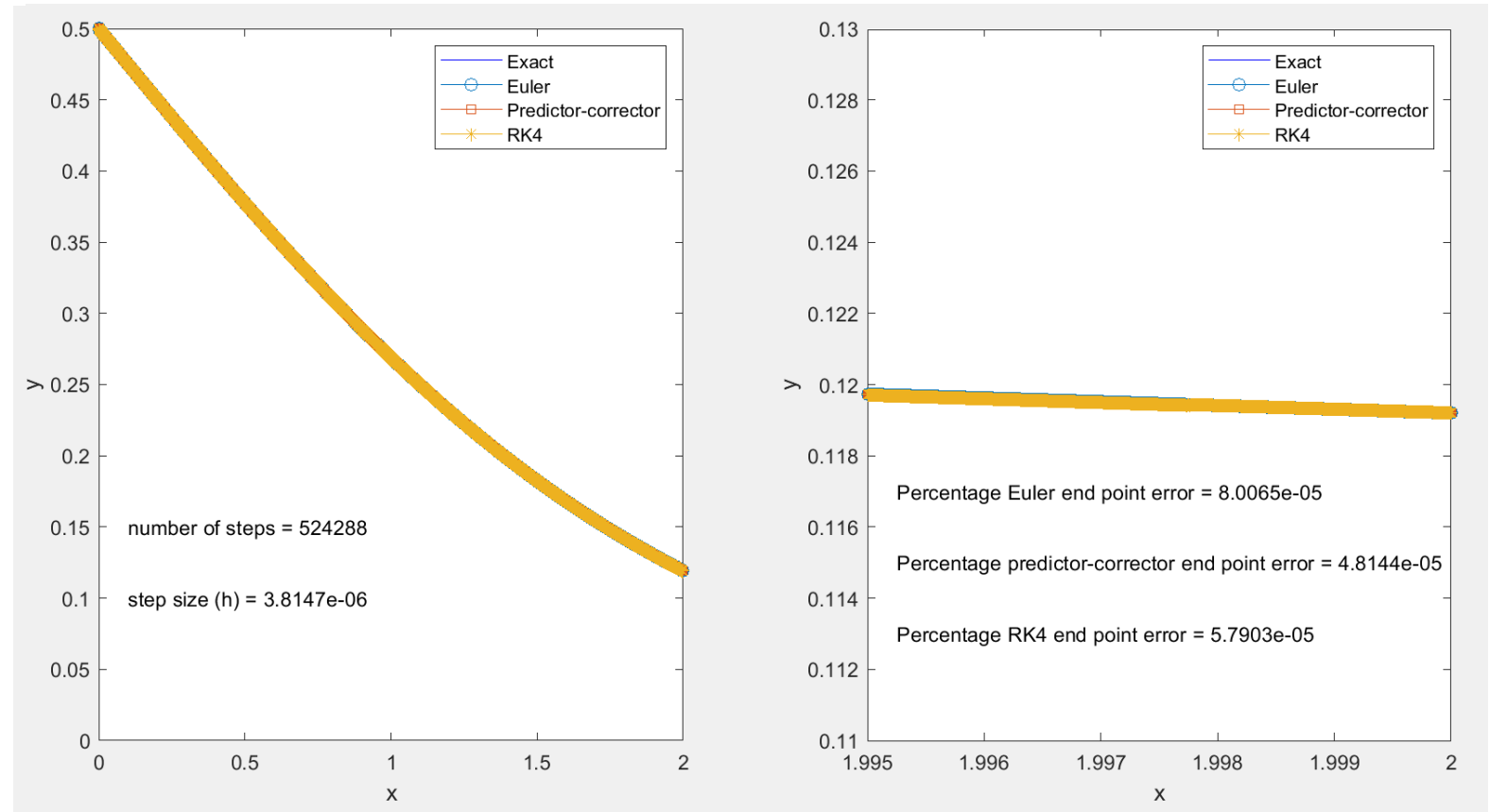
After about 8,000 steps the predictor corrector catches up with RK4. But even after reaching over 100,000 evaluation points and a step size of $\sim 10^{-5}$, Euler still has an error \sim **an order of magnitude** greater than RK4!



Comparison in more detail

After **several hours** matlab reaches $\frac{1}{2}$ million points and Euler begins to catch up with RK4...

Take home message is that a sensible choice of algorithm and implementation in code can save an enormous amount of computing time (and energy, CO2...).



Multi-Step Methods

There are other multistep methods, where the computed solutions depend on more than one past step of y_n

Popular methods include the *Adams-Bashforth* and the *Adams-Moulton* algorithms. See the text books for further details (eg Burden and Faires Ch 5.6)

Adams-Bashforth

$$y_p^{n+1} = y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$$

$$y_c^{n+1} = y_n + \frac{h}{24}(9f_p^{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$$

Adams-Bashforth and fourth order Runge-Kutta are of similar accuracy, however Adams-Bashforth is more computationally efficient since values can be stored from previous time steps and hence only two computations of $f(x, y)$ are required per step.

Many computer algorithms start with fourth order Runge-Kutta to get 4 points and then change to Adams-Bashforth for speed.

A note on MATLAB routines...

ode45 and ode23 are based on an explicit Runge-Kutta formula.

ode113 is a variable order Adams-Bashforth-Moulton solver.

Look at manuals to see descriptions of these and others.

Part D – Second order equations

Second order equations

An example of a second order equation might be a damped harmonic oscillator.

Equations of motion for this system are:

$$F_{mass} = ma = m\ddot{x}$$

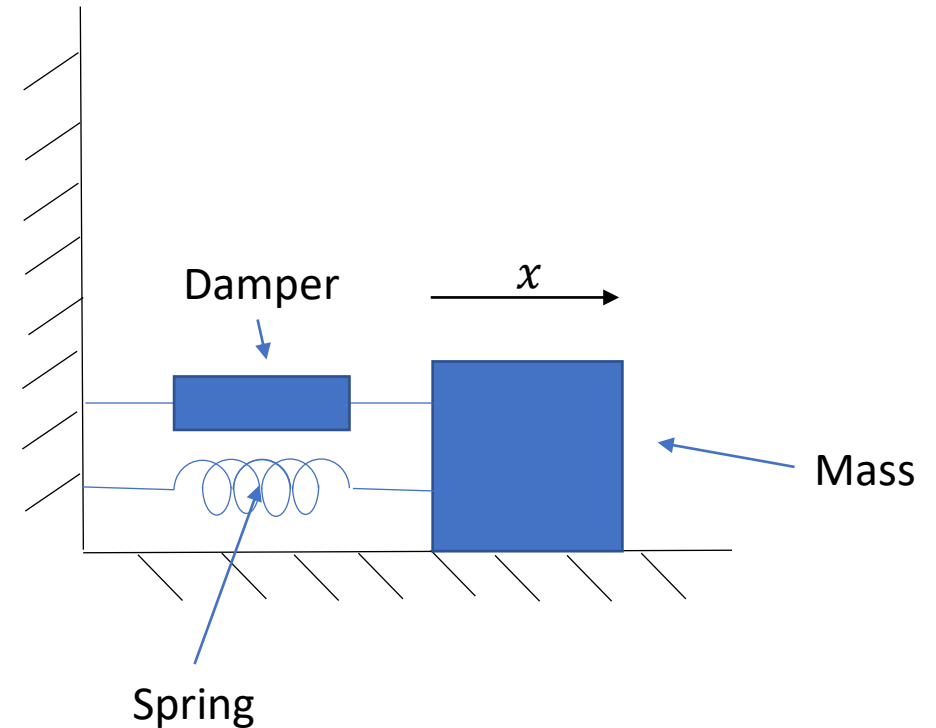
$$F_{spring} = -kx$$

$$F_{damper} = -c\dot{x}$$

Equating forces:

$$m\ddot{x} = -kx - c\dot{x}$$

$$\ddot{x} + \frac{c}{m}\dot{x} + \frac{k}{m}x = 0$$

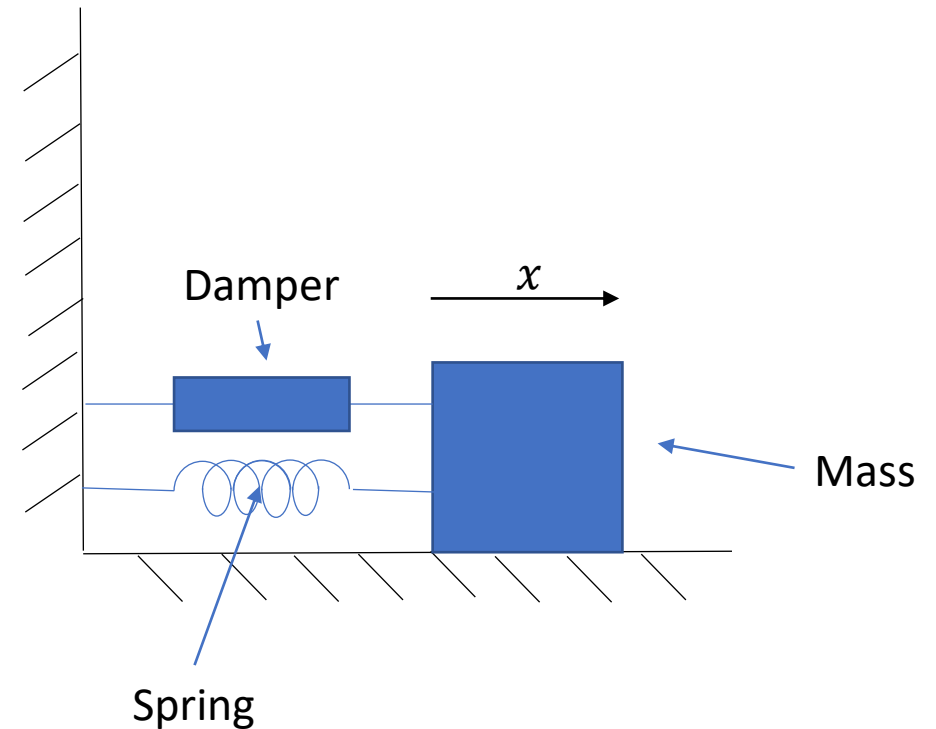


Second order equations

This is a second order equation.
However all of our methods that we
have considered so far work for first
order equations.

To solve this problem convert the
second order equation to a pair of first
order equations by treating y' as an
independent variable.

Let's see...



Second order general method

Consider a general second order ODE of the form

$$y'' = f(x, y, y')$$

Treat y' as an independent variable:

$$\begin{aligned}\frac{dy'}{dx} &= f(x, y, y') \\ \frac{dy}{dx} &= y'\end{aligned}$$

Writing this in matrix form:

$$\frac{d}{dx} \begin{bmatrix} y \\ y' \end{bmatrix} = \begin{bmatrix} y' \\ f(x, y, y') \end{bmatrix}$$

OR...

$$\frac{d\mathbf{Y}}{dx} = \mathbf{F}(x, \mathbf{Y})$$

Hence we can write the Euler method as:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + h\mathbf{F}_n$$

Second order equations

To solve this problem define:

$$\omega_0 = \sqrt{\frac{k}{m}} = \text{natural frequency}$$

$$\zeta = \frac{c}{2\sqrt{km}} = \text{damping ratio}$$

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2x = 0$$

Next define:

$$\dot{x} = v$$

$$\dot{v} = -2\zeta\omega_0v - \omega_0^2x$$

Using this we can write the matrix equation:

$$\frac{d}{dt} \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\zeta\omega_0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}$$

Or...

$$\dot{\mathbf{Y}} = \mathbf{A}\mathbf{Y}$$

The Euler equation becomes:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + h\mathbf{A}\mathbf{Y}_n = (\mathbf{I} + h\mathbf{A})\mathbf{Y}_n$$

Second order equations

Lets consider a simple example to understand this technique. A simple harmonic oscillator:

$$y'' = -y \quad \text{with initial conditions} \quad y(0) = 1, \quad y'(0) = 0.$$

In matrix form, this equation can be written as

$$\frac{d}{dx} \begin{bmatrix} y \\ y' \end{bmatrix} = \begin{bmatrix} y' \\ -y \end{bmatrix} \quad \text{with initial conditions} \quad \begin{bmatrix} y_0' \\ y_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Write a MATLAB function

Second order equations

```
function shm(h)
% Euler algorithm to evaluate  $y'' = -y$  ,  $y(0) = 1$  ,  $y'(0) = 0$  .
% h is the step size.

x = [0:h:7]'; % x range
fn = cos(x); % exact solution

y = x; % get matrix sizes for y and y'
yd = x;

y(1) = 1; % initial conditions
yd = 0;

for i=1:(length(x)-1)
    y(i+1) = y(i) + h*yd(i); %Euler algorithm
    yd(i+1)=yd(i) - h*y(i);
end

plot(x,fn,x,y, '*')
```

Second order equations (Euler and Modified)

20 Steps

- Euler all over the place
- Predictor-corrector begins to approximate the function well.

40 steps

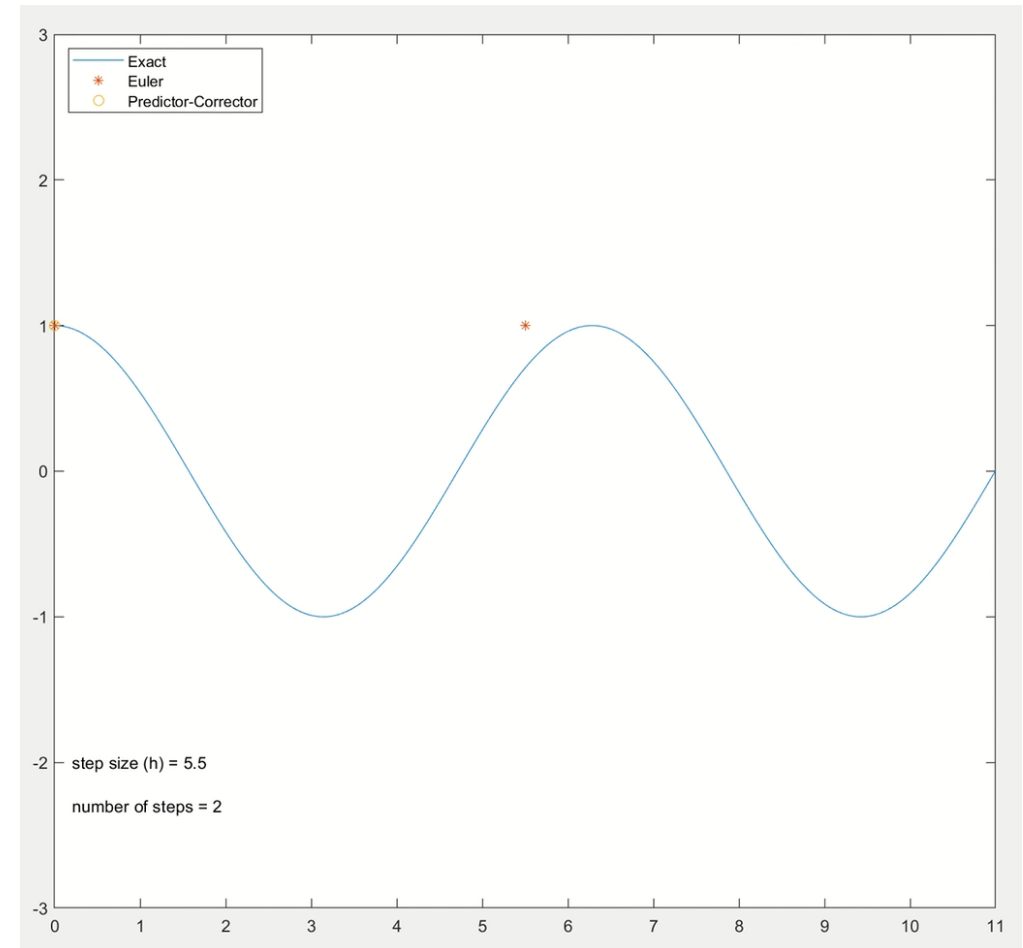
- Euler oscillating wildly
- Predictor-corrector getting close to real function.

60 steps

- Euler still far from the actual function
- Predictor-corrector very close to real function.

80 steps

- Game over – predictor-corrector is almost exact!



Part E – Boundary value problems

Initial value problems

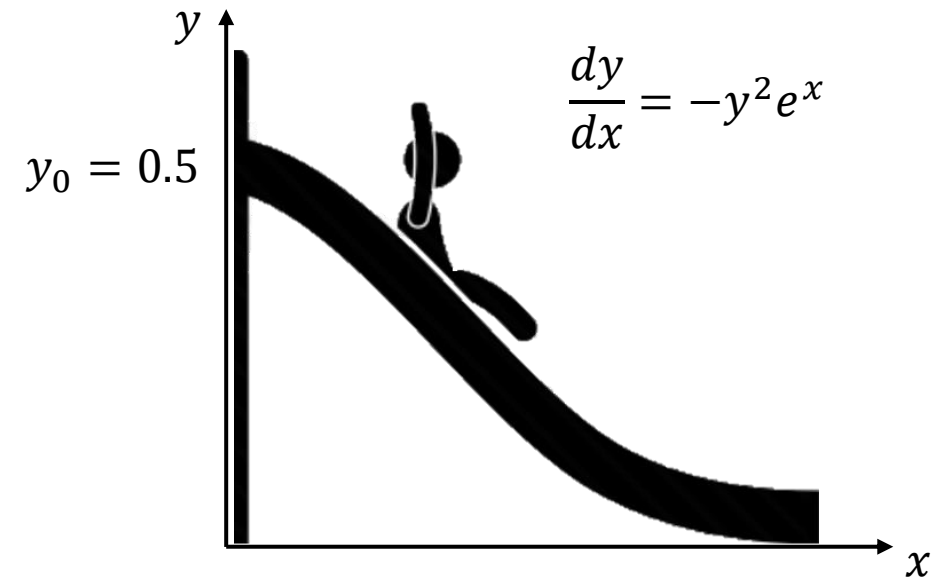
So far we have considered Initial value problems:

We have been given an **ODE** along with a specified value (the **initial condition**), of an unknown function at a given point in the solution domain.

Returning to our first problem: Harry's change in height as a function of position (x) is given by the **ODE**:

$$y' = \frac{dy}{dx} = -y^2 e^x$$

And we choose his height at $x = 0$ (our **initial condition**) to be $y(0) = y_0 = 0.5$

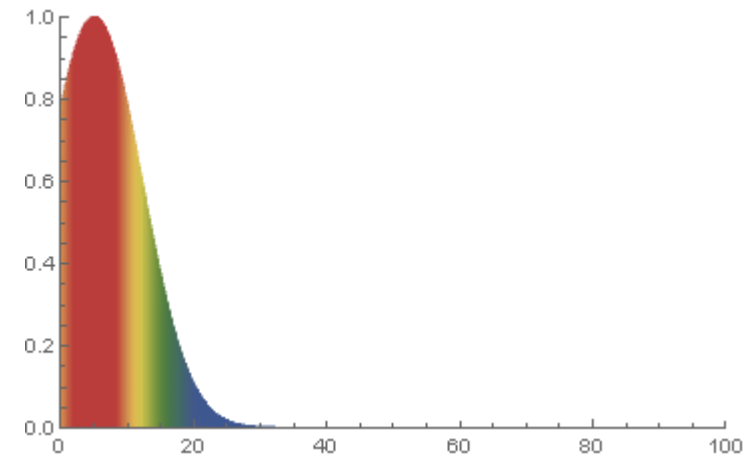
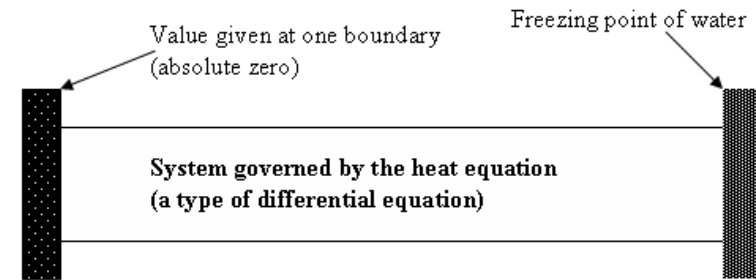


Boundary value problems

If we are given a differential equation of order greater than 1, then the **boundary values** may be given instead of the initial conditions.

An example might be the heat flow along a rod whose ends are kept at a fixed temperature (see right).

We will touch on this in Lecture 4 and you will see much more when studying Finite difference methods for example.



Boundary value problems

One method to solve boundary value problems is the shooting method (we will cover this in the computational class). It works as follows:

The shooting method

- Guess an initial value for y' .
- Integrate the equation (Euler?).
- Compare the final y with the given boundary.
- Adjust the initial guess of y' and repeat.
- Once you have two values either side of the given boundary interpolate.

Let's quickly run through this for our SHM equation:

$$y'' = -y$$

But instead of **initial values**:

$$y(0) = 1, \quad y'(0) = 0$$

We are given **boundary conditions** of:

$$y(0) = 1, \quad y(11) = 0$$

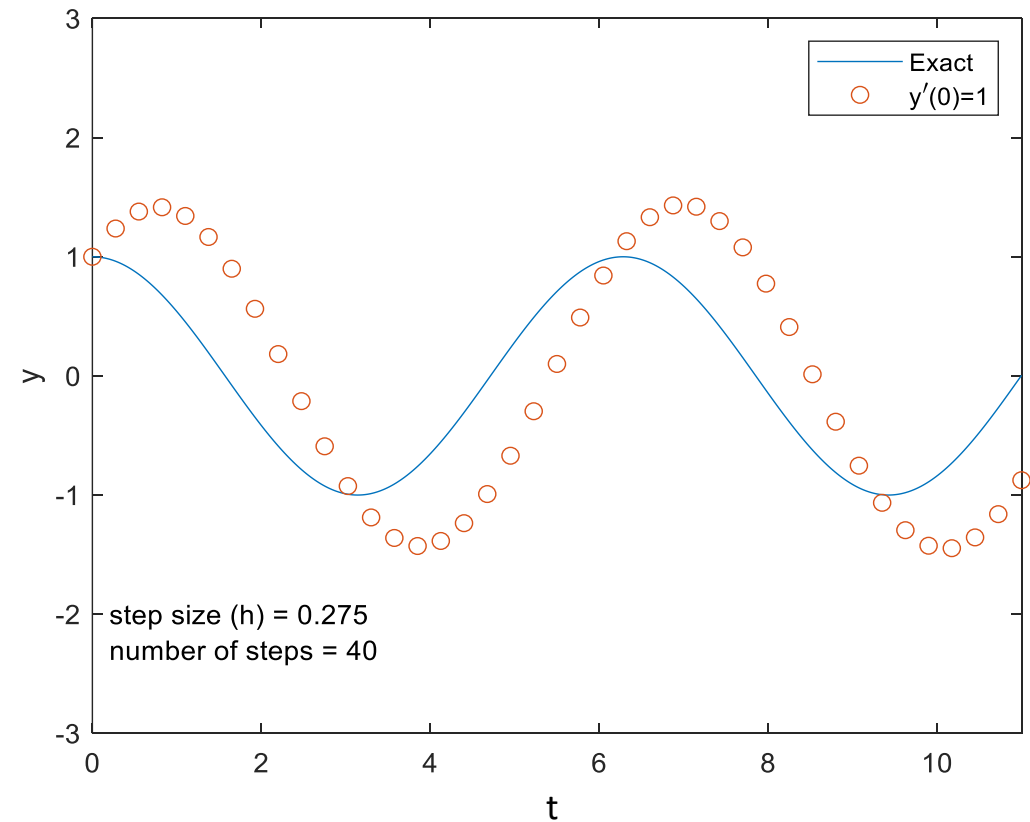
Boundary value problems

Guess a initial value for $y'(0)$

I guess: $y'(0) = 1$

I see: $y(11) = -0.8755$

Too low!!



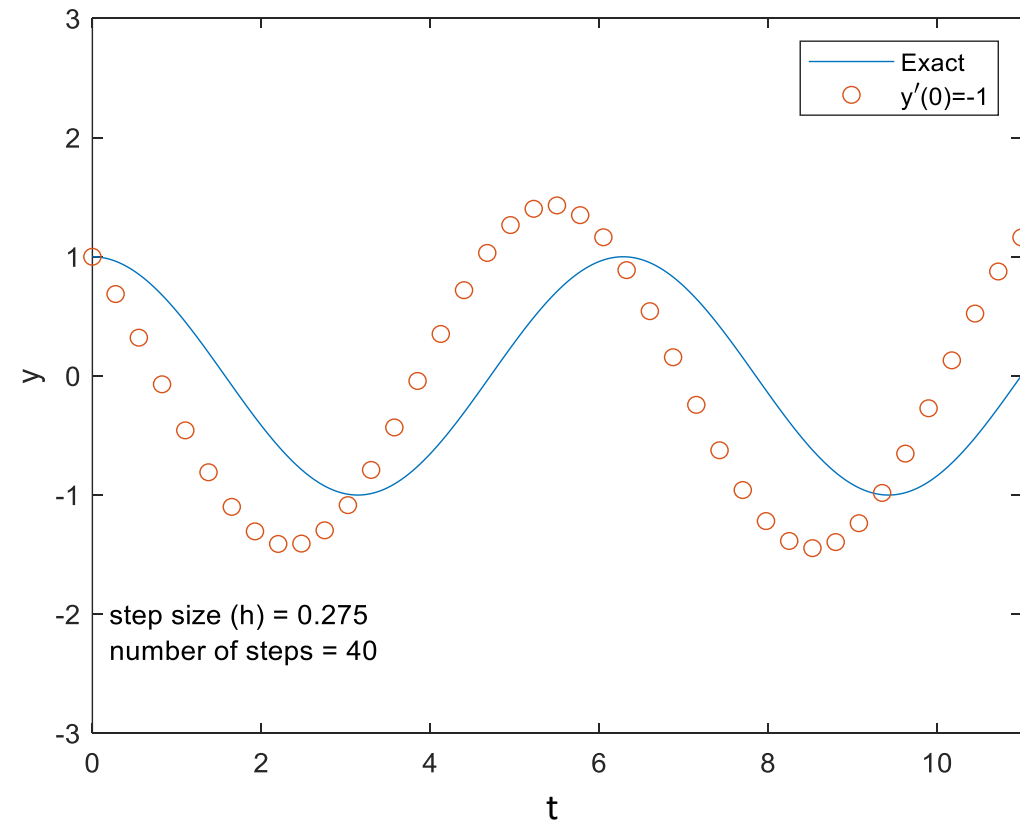
Boundary value problems

Guess a new initial value for $y'(0)$

I guess: $y'(0) = -1$

I see: $y(11) = 1.1624$

Too high!!



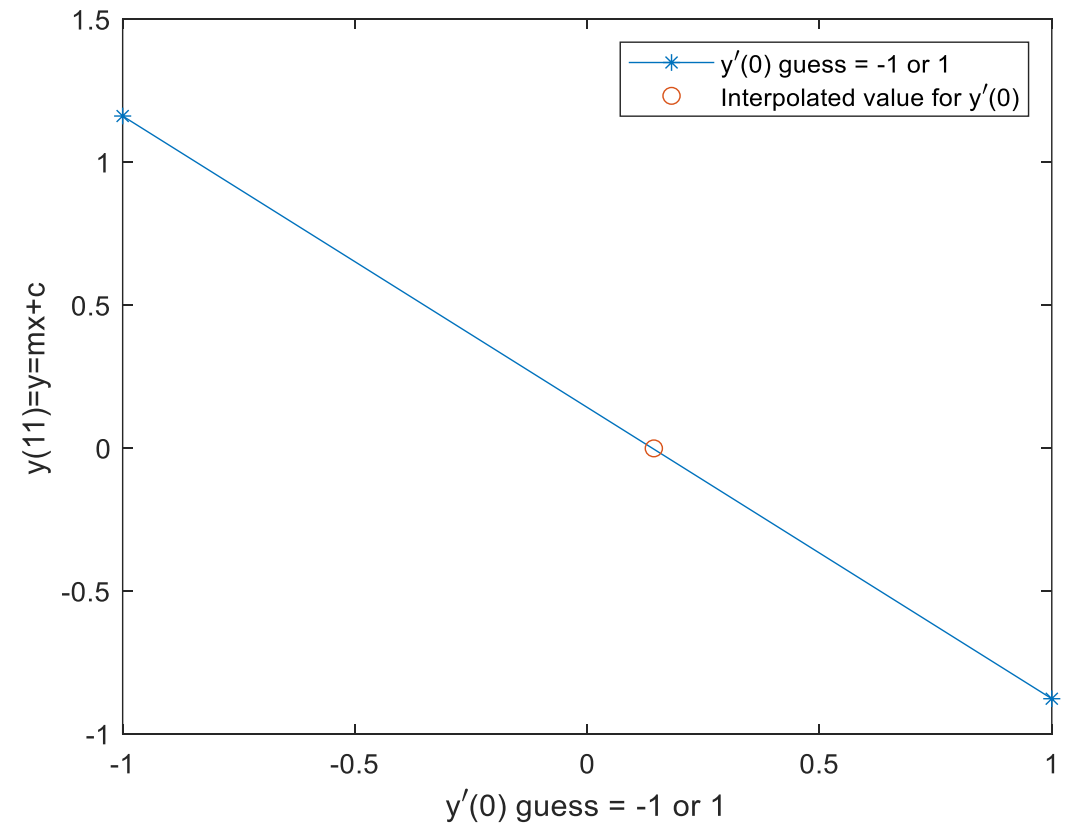
Boundary value problems

Use my two guesses to interpolate for $y'(0)$ at :
 $y(11) = 0$

I see an interpolated value:

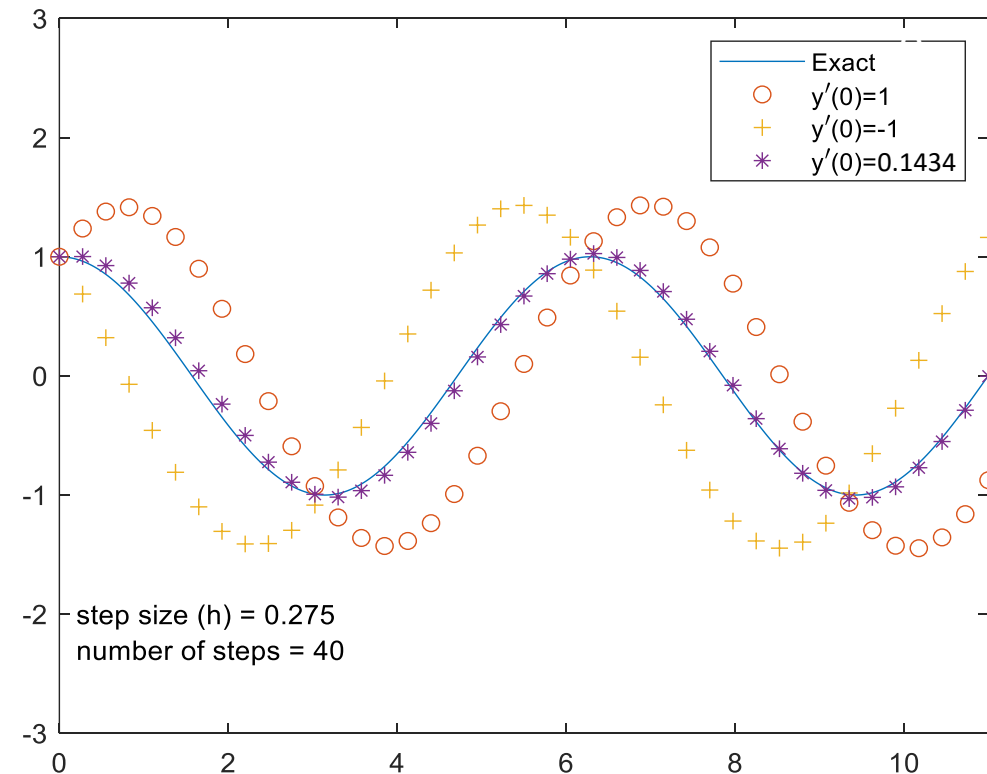
$$y'(0) = 0.1434$$

Let's try it...



Boundary value problems

Very close to the correct value!!




The Shooting Method

Panopto bonus video

B1 Numerical Algorithms

4 Lectures, MT 2022
Lecture three.
Bonus video 2 – The shooting method
Wes Armour



Errors/Typos/Questions to: wes.armour@eng.ox.ac.uk

1

★ 00:11

Boundary value problems

One method to solve boundary value problems is the shooting method (we will cover this in the computational class). It works as follows:

The shooting method

- Guess an initial y' .
- Integrate the equation (Euler?).
- Compare the final y with the given boundary.
- Adjust the initial guess of y' and repeat.
- Once you have two values either side of the given boundary interpolate.

Let's quickly run through this for our SHM equation:

$$y'' = -y$$

But instead of initial values:

$$y(0) = 1, \quad y'(0) = 0$$

We are given boundary conditions of:

$$y(0) = 1, \quad y(11) = 0$$

2

★ 01:21

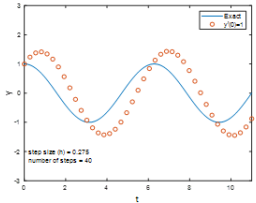
Boundary value problems

Guess a initial value for $y'(0)$

I guess: $y'(0) = 1$

I see: $y(11) = -0.8755$

Too low!!



3

★ 00:49

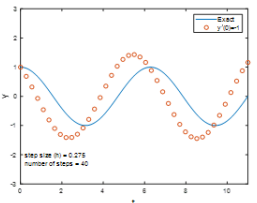
Boundary value problems

Guess a new initial value for $y'(0)$

I guess: $y'(0) = -1$

I see: $y(11) = 1.1624$

Too high!!



4

★ 00:38

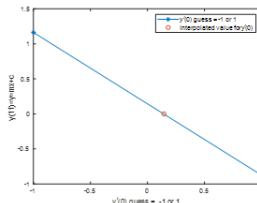
Boundary value problems

Use my two guesses to interpolate for $y'(0)$ at : $y(11) = 0$

I see an interpolated value:

$$y'(0) = 0.1434$$

Let's try it...

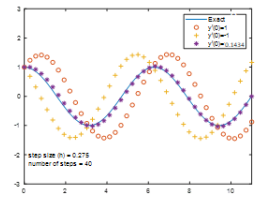


5

★ 02:26

Boundary value problems

Very close to the correct value!!



6

★ 01:44

Richardson Extrapolation

Introduced by Lewis Fry Richardson, this technique is extremely useful for increasing accuracy in results from several different numerical techniques. An overview (for our study of solutions to ODEs) is as follows:

1. Compute the approximate solution to our ODE for a decreasing sequence of step sizes. (We cannot go to a zero step size because of numerical problems and noise.)
2. Assume that the approximate solutions are functions of the step size and fit a polynomial as a function of 'h', the step size.
3. Plug $h=0$ into the polynomial to get an estimate of the limit.

See the printed notes for more details.

Richardson Extrapolation

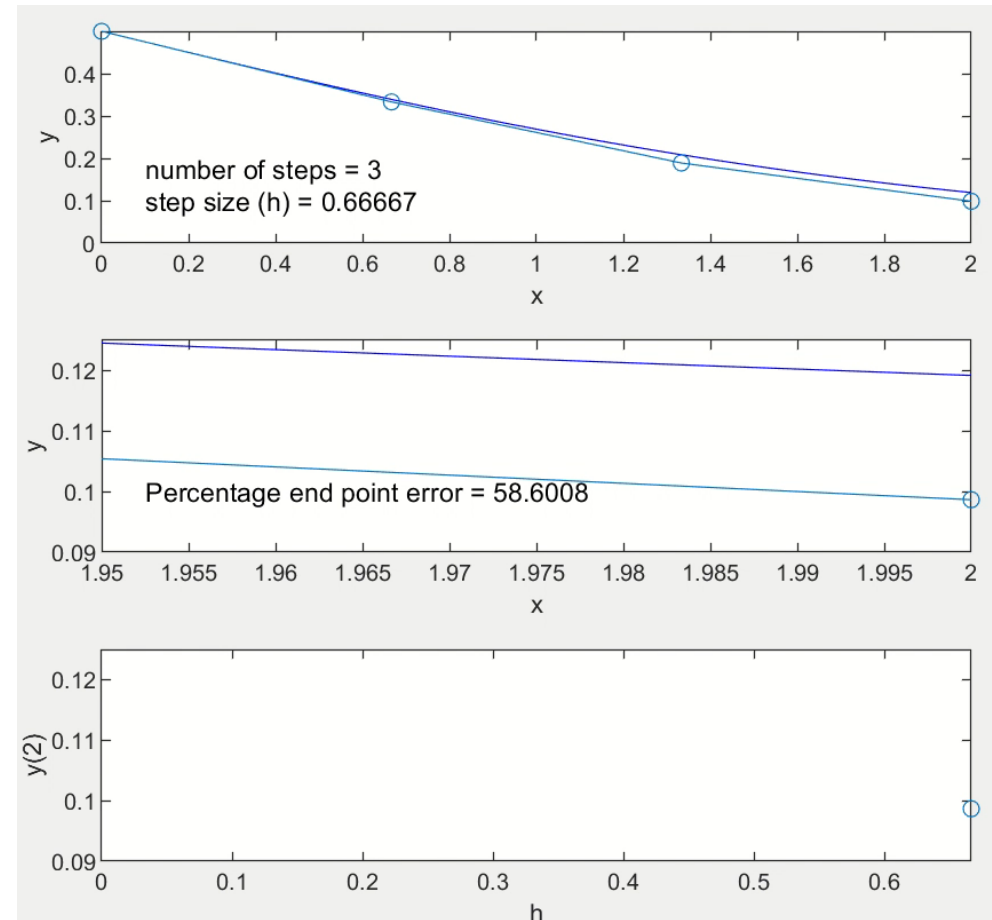
We perform the (terrible) forward Euler (as shown at the start of our lecture) for 3, 5, 7 and 9 steps (I've chosen these randomly, you could choose whatever values you like).

I want to know the value of y at $x = 2$. So we plot the value of $y(2)$ against h .

We then fit a polynomial - I have fitted a quadratic. Note: I have 3 unknowns and 4 data points to constrain the fit!

The constant in the quadratic fit will represent the value of $y(2)$ when $h = 0$.

Using Richardson's method we see an error that is nearly two orders of magnitude smaller!!



Conclusions – what have we learnt?

- How to solve ODEs using numerical techniques.
- Different order methods for numerical solution to ODEs and errors associated with them.
- Computationally efficient methods of solution (adaptive step size methods).
- Different types of problem, Initial value and boundary conditions.



Additional material: The Shooting Method

For linear equations the method is easy

Choose two initial velocities and integrate the equation to get two final values for y .

$$(y(0), dy/dt(0)_1) \rightarrow y(t_{\text{final}})_1$$

$$(y(0), dy/dt(0)_2) \rightarrow y(t_{\text{final}})_2$$

Equations are linear, thus

$$\lambda_1(y(0), dy/dt(0)_1) \rightarrow \lambda_1 y(t_{\text{final}})_1$$

$$\lambda_2(y(0), dy/dt(0)_2) \rightarrow \lambda_2 y(t_{\text{final}})_2$$

Additional material: The Shooting Method

Two simultaneous equations

$$(\lambda_1 + \lambda_2) y(0) = y(0)$$

$$\lambda_1 y(t_{\text{final}})_1 + \lambda_2 y(t_{\text{final}})_2 = y(t_{\text{final}})_{\text{required}}$$

Solve for λ_1 and λ_2

Additional material: The Shooting Method

Example (Gerald)

Solve

$$\frac{d^2y}{dx^2} - \left(1 - \frac{x}{5}\right)y = x$$

Such that

$$y(1) = 2, y(3) = -1$$

Guess initial conditions and integrate numerically

$$y'(1) = -1.5 \rightarrow y(3) = 4.7876$$

$$y'(1) = -3.0 \rightarrow y(3) = 0.4360$$

Substitute into the equations

$$\lambda_1 + \lambda_2 = 1$$

$$4.7867 \lambda_1 + 0.4360 \lambda_2 = -1$$

Thus

$$\lambda_1 = -0.33$$

$$\lambda_2 = 1.33$$