Numerical Solution of Differential Equations

Statement of the problem 1

Euler's method and why it's bad

The mid-point method

The Runge-Kutta method

Other methods

Higher-order DEs

Two-point Boundary Value Problems

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

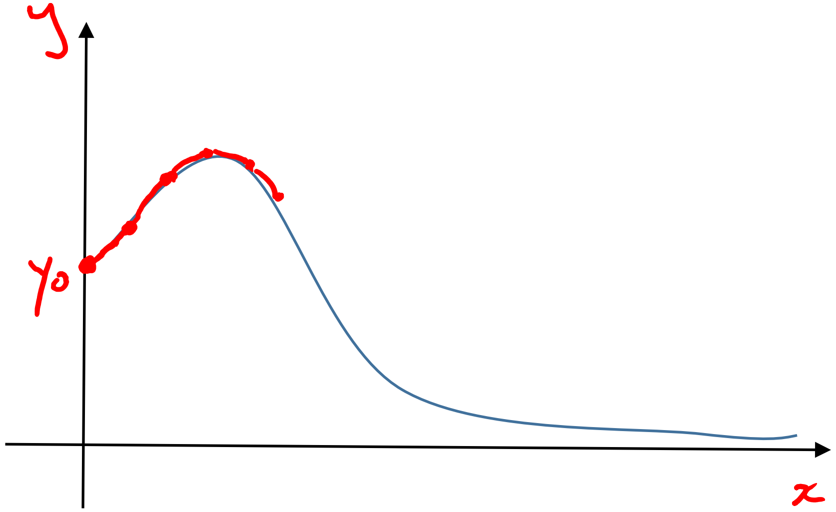
$$= \sin^2 x \cdot y$$

We would like to solve (for the moment) the first-order differential equation

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

with an *initial condition*

$$y(x_0) = y_0 \ .$$



In principle, if we know the derivatives at each point, we should be able to construct the entire function.

<u>Euler's method</u>

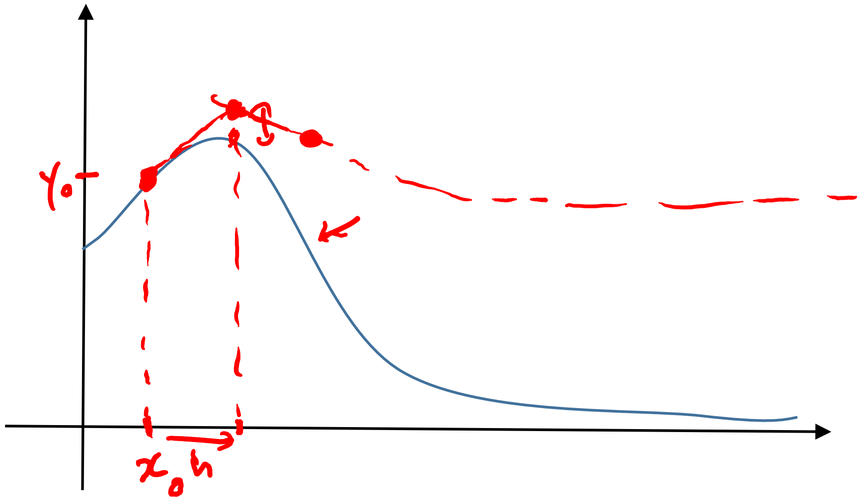
Start with Taylor series

$$y(x+h) = y(x) + h\,y'(x) + \underbrace{\frac{h^2}{2!}\,y''(x) + \dots}_{O(h^2)}$$

~~Re-arrange:~~

Truncate:

$$y(x+h) \approx y(x) + h\,y'(x).$$



Euler's method:

1. Start at $x = x_0, y(x_0) = y_0$

2. Compute

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

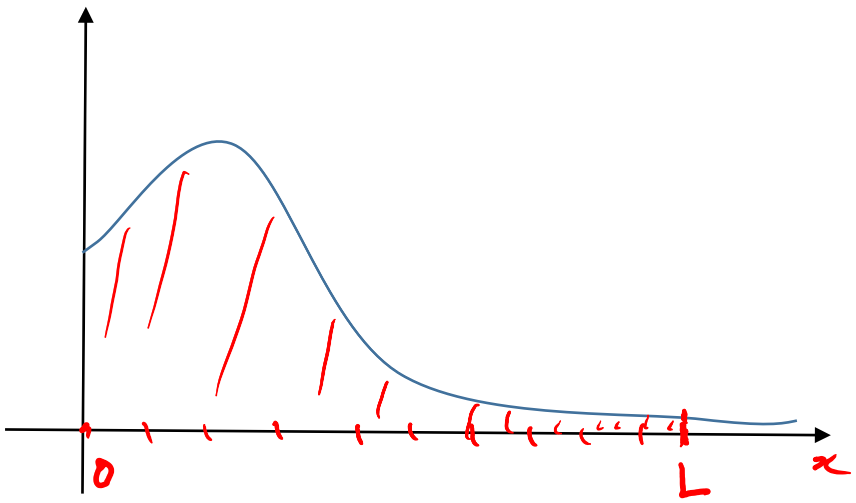
3. Repeat 2

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

Error in Euler's method

Error in each step: $O(h^2)$

Number of steps in an interval of length L: $N = \frac{L}{h}$



Cumulative error at the end: $\sim O\left(\frac{L}{h} h^2\right) = O(hL)$

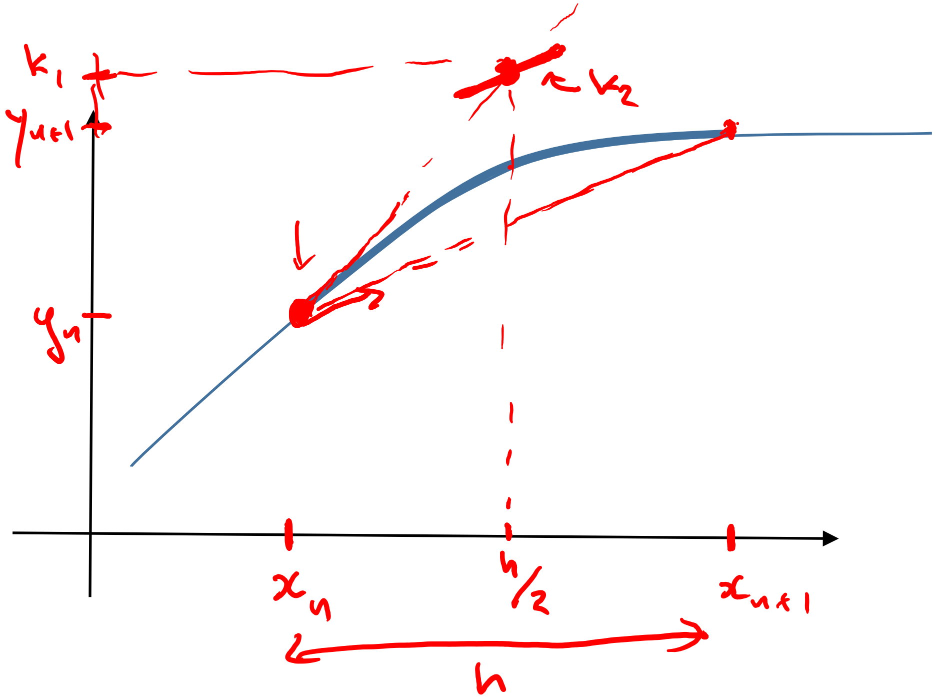
$L_{max} \sim \frac{1}{h}$

1. Very small step-sizes required
2. Really not good if you want to e.g. integrate the function at the end

The mid-point method

(a.k.a 2nd-order Runge-Kutta method)

The idea: instead of taking the slope at the beginning of the interval, take it at the half-step along to the next point.



1. Estimate the mid-point using an Euler Step

$$\frac{dy}{dx} = f$$

$$\rightarrow \quad k_1 = y_{n+\frac{1}{2}} = y_n + \frac{h}{2} f(x_n, y_n)$$

2. Compute the slope at this estimated mid-point

$$\rightarrow \quad k_2 = f\left(x_n + \frac{h}{2}, k_1\right)$$

3. Do a time step using this slope.

$$y_{n+1} = y_n + hk_2 .$$

Mid-point method:

1. Start at $x = x_0, y(x_0) = y_0$
2. Compute

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1 h) \\ y_{n+1} &= y_n + hk_2 \end{aligned}$$

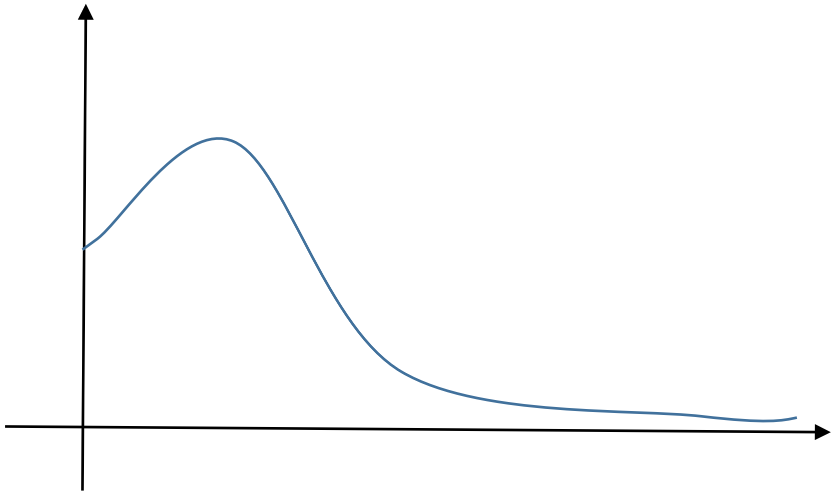
3. Repeat 2

Error in the mid-point method

Error in each step: $\sim O(h^3)$

Number of steps in an interval of length L: $O\left(\frac{L}{h}\right)$

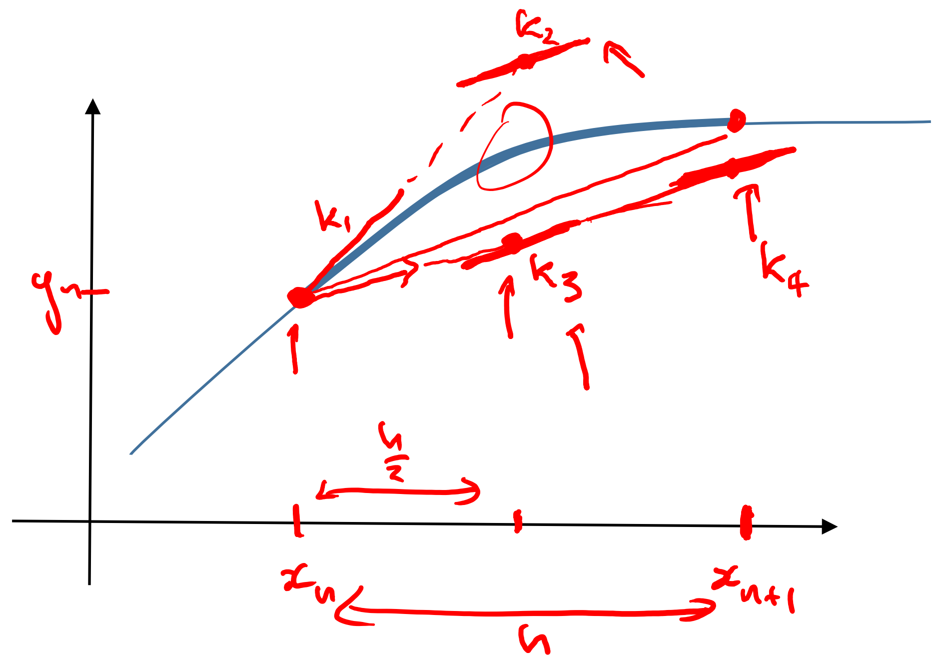
Cumulative error at the end: $O(Lh^2)$.



$L_{max} = \frac{1}{h^2}$.

The 4th-order Runge-Kutta method

This is the "workhorse" of most numerical methods.



The idea: get a better estimate of the "total" slope by using a Weighted Average of the slopes across the interval:

Specifically:

> k_1 is the slope at the beginning of the interval
> k_2 is the slope at the midpoint, using y and k_1
> k_3 is the slope at the midpoint, using y and k_2
> k_4 is the slope at the endpoint, using y and k_3

We then "step" using an average of these slopes:

weighted average slope $= \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

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

4th-order Runge-Kutta method

1. Start at $x = x_0, y(x_0) = y_0$
2. Compute

$$
\begin{aligned}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + \frac{1}{2}h, y_n + \frac{h}{2}k_1) \\
k_3 &= f(x_n + \frac{1}{2}h, y_n + \frac{h}{2}k_2) \\
k_4 &= f(x_n + h, y_n + h k_3) \\
y_{n+1} &= \underline{y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)}
\end{aligned}
$$

3. Repeat 2

Error in each step: $O(h^5)$

Number of steps in an interval of length L: $O\left(\frac{L}{h}\right)$

Cumulative error at the end:

$\sim O(h^4 L)$

$L_{max} \sim \frac{1}{h^4}$

$\sim \left(\frac{1}{0.1}\right)^4 \sim 10^4.$

Other important methods:

- **Adaptive step Runge-Kutta method**

  Uses two interleaved methods of different order (say 5th and 4th order), and uses the difference between these two to estimate the error. If the error exceeds a given threshold, the step size is changed.

- **Predictor-Corrector methods**

  These extrapolate the existing curve to a new point (predict), and then use this new point to *correct* the estimation.

- **Bulirsch-Stoer method**

  Uses rational function interpolation to extrapolate to the next point, then match this to the power series of the function. Complicated but very useful for Solving “stiff” ODEs.

Solving higher-order equations

Higher order ODEs can be converted to *systems of first order ODEs.*

E.g.

$$\frac{d^3y}{dx^3} - \frac{d^2y}{dx^2} + 2\frac{dy}{dx} + 5y = 0 \implies \frac{dy_2}{dx} - y_2 + 2y_1 + 5y_0 = 0$$

$$\implies \frac{dy_2}{dx} = y_2 - 2y_1 - 5y_0$$

$$\frac{dy_1}{dx} = y_2$$

$$\frac{dy_0}{dx} = y_1$$

Let

$$y_0 = y$$

$$y_1 = \frac{dy_0}{dx} = \frac{dy}{dx}$$

$$y_2 = \frac{dy_1}{dx} = \frac{d^2y}{dx^2}$$

$$y_3 = \frac{dy_2}{dx} = \frac{d^3y}{dx^3}$$

$$\frac{d}{dx}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ -5y_0 - 2y_1 + y_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -5 & -2 & 1 \end{bmatrix}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

This new system can then be solved using (say) Runge-Kutta.

Two-point boundary value problems

For higher-order DEs, we are often give a *two-point boundary value problem* instead of an initial condition.

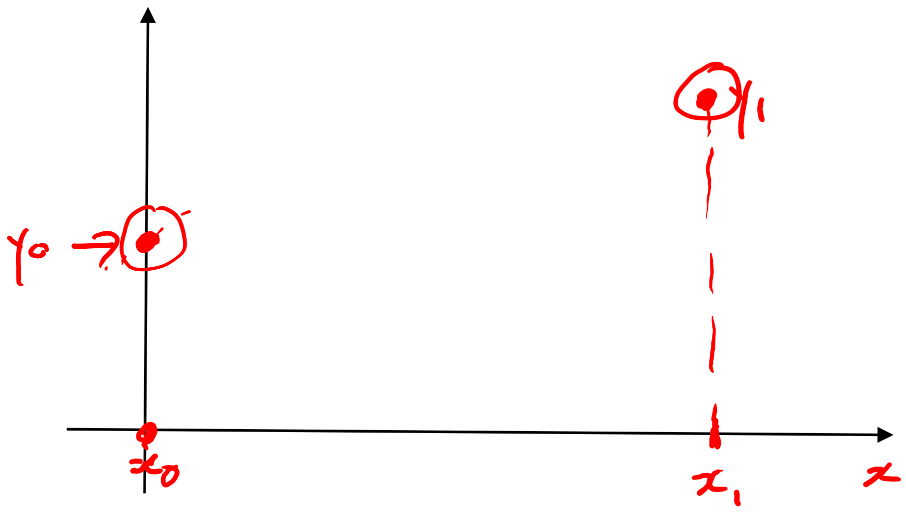
E.g. the 2nd-order differential equation

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

With *boundary conditions*

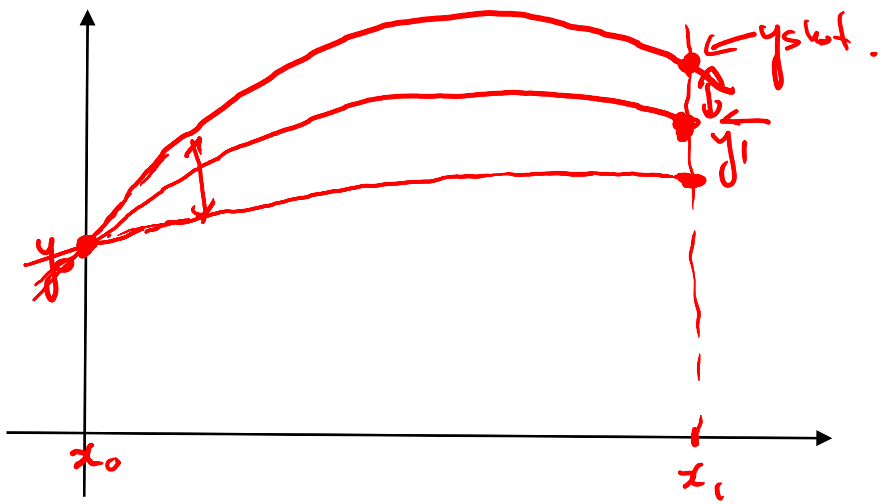
$$y(x_0) = y_0 \quad , \quad y(x_1) = y_1$$

Note that the 1st-order derivatives are not specified.



The shooting method

The idea: Start at one side, pick a 1st derivative, and “shoot” towards the other.



By changing the value of the first derivative, you can minimise the distance between the “shot” and the “target”.

Shooting method pseudo-code

```
Function yshot(dydx0,f,x0,x1,y0)
        x, y = odesolve(f,x0,x1,y0,dydx0)        # solve the ode y’ = f(x,y), starting at x0, ending at x1
                                                  # and with initial conditions y0, dydx0
        yshot = y(end)        # pick the final value


dydx = minimise(abs(yshot-y1))   # find the value of dydx0 that minimises
                                 # the distance to y1

x,y = odesolve(f,x0,x1,y0,dydx)   # solve the ode for the particular value of dydx
```

Other main methods for solving DEs:

The *relaxation method*

Start with an estimated solution and change each point to minimize the average error

Finite Differences

Discretize the solution on a regular grid - we will cover this next week

The *Finite element method*

The "gold standard" for solving all differential equations (but could be its own entire subject).
Convert the DE into integral form, approximate the solution using polynomial interpolation. The Differential Equation is then reduced to solving a big sparse matrix.