Physics 68/118 Computational Physics

Class 3: Connecting model to data

Objectives for today

- Introduce Tracker, which allows us to get the trajectory of our object from the video.
- Get Tracker data into our notebook.
- Introduce some tools to assess the performance of numerical algorithms.

Tracker

Getting things into tracker

Tools to assess numerical algorithms

- **Error** — How the result differs from the “true” solution
- **Order** — How the performance scales with input or error
- **Cost** — The computational work required
- **Stability** — Whether the algorithm succeeds for given input

Error

Fundamentally, arises from the difficulty of representing real numbers on a computer.

Is **not** random in origin, but arises from representation or approximation.

Propagates through an algorithm in ways that can be adverse or beneficial.

Error

Integer arithmetic is **exact**.

$2x3 = 0b0010 \times 0b0011 = 0b0110$

Overflow and underflow can occur

Error

Integer arithmetic is **exact**.

2x3 = 0b0010 x 0b0011 = 0b0110

Overflow and underflow can occur

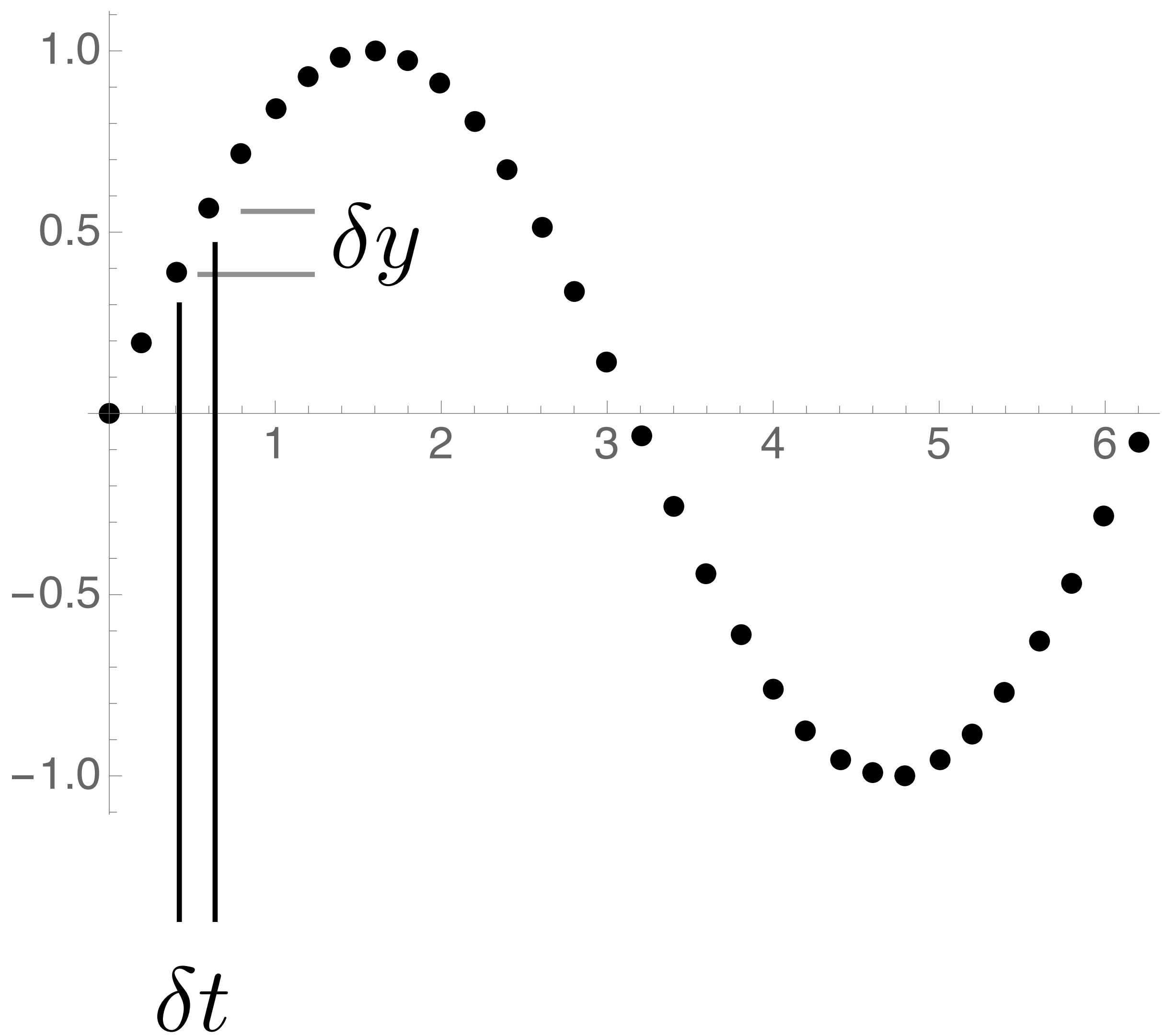
Floating point arithmetic is typically **not** exact.

Stored in the form $A \times 2^B$

		A mantissa	B exponent
	Double precision	52 bits	11 bits
Some are representable	0.5	0xb1000000000000000	-2
Most are not	1/3	0xb1010101010101010…	

Error in representation $\approx 1.5 \times 10^{-17}$

Another source of error is **approximation**



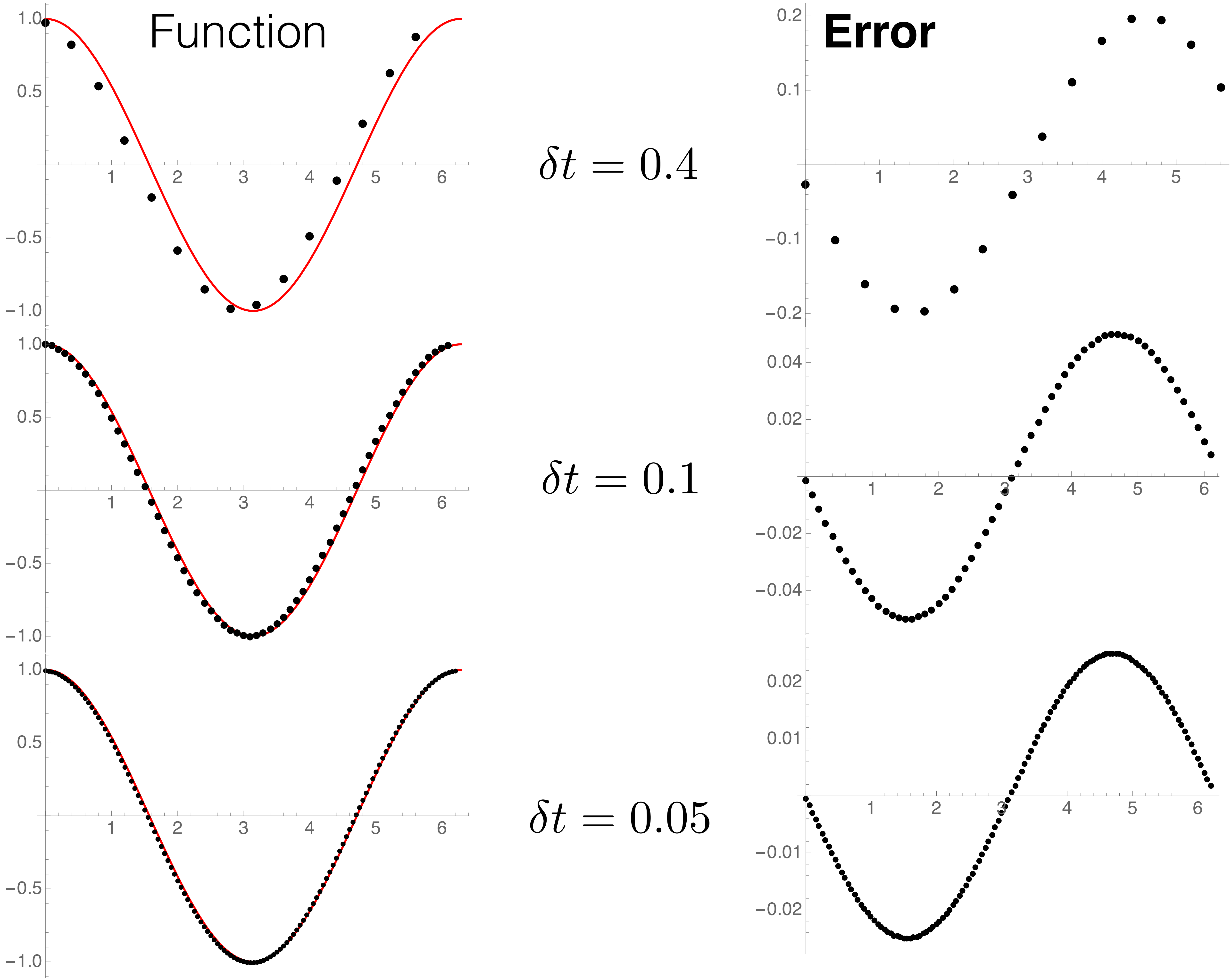
Here, we have equally spaced points

$$y_i \in \{y_1, y_2, \ldots, y_N\}$$

Elementary calculus

$$\lim_{\delta t \to 0} \frac{y(t+\delta t) - y(t)}{\delta t}$$

suggests $\dfrac{dy}{dt} \approx \dfrac{y_{i+1} - y_i}{\delta t}$

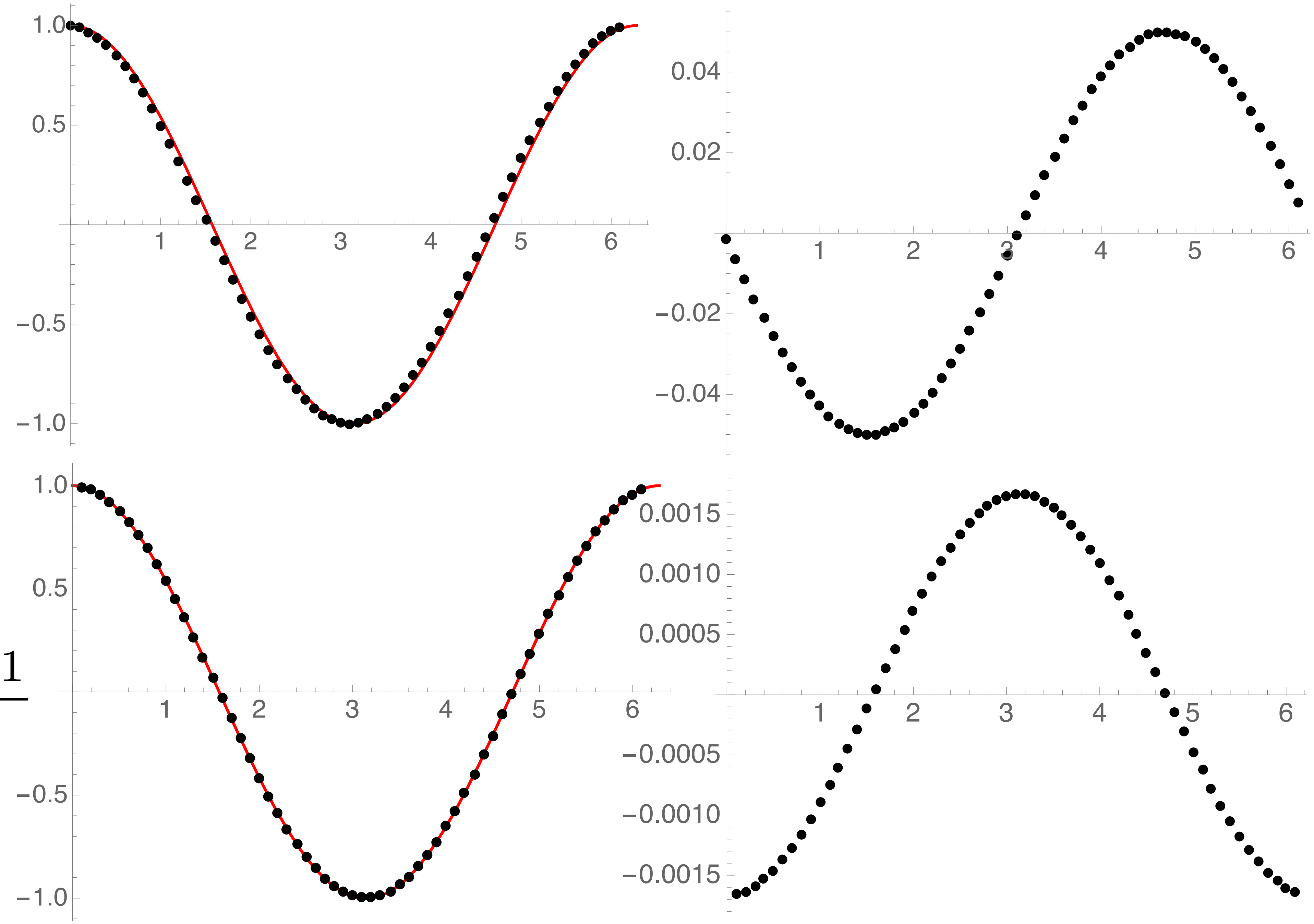


Function
Error
$\delta t = 0.4$
$\delta t = 0.1$
$\delta t = 0.05$

A simple change dramatically improves the error

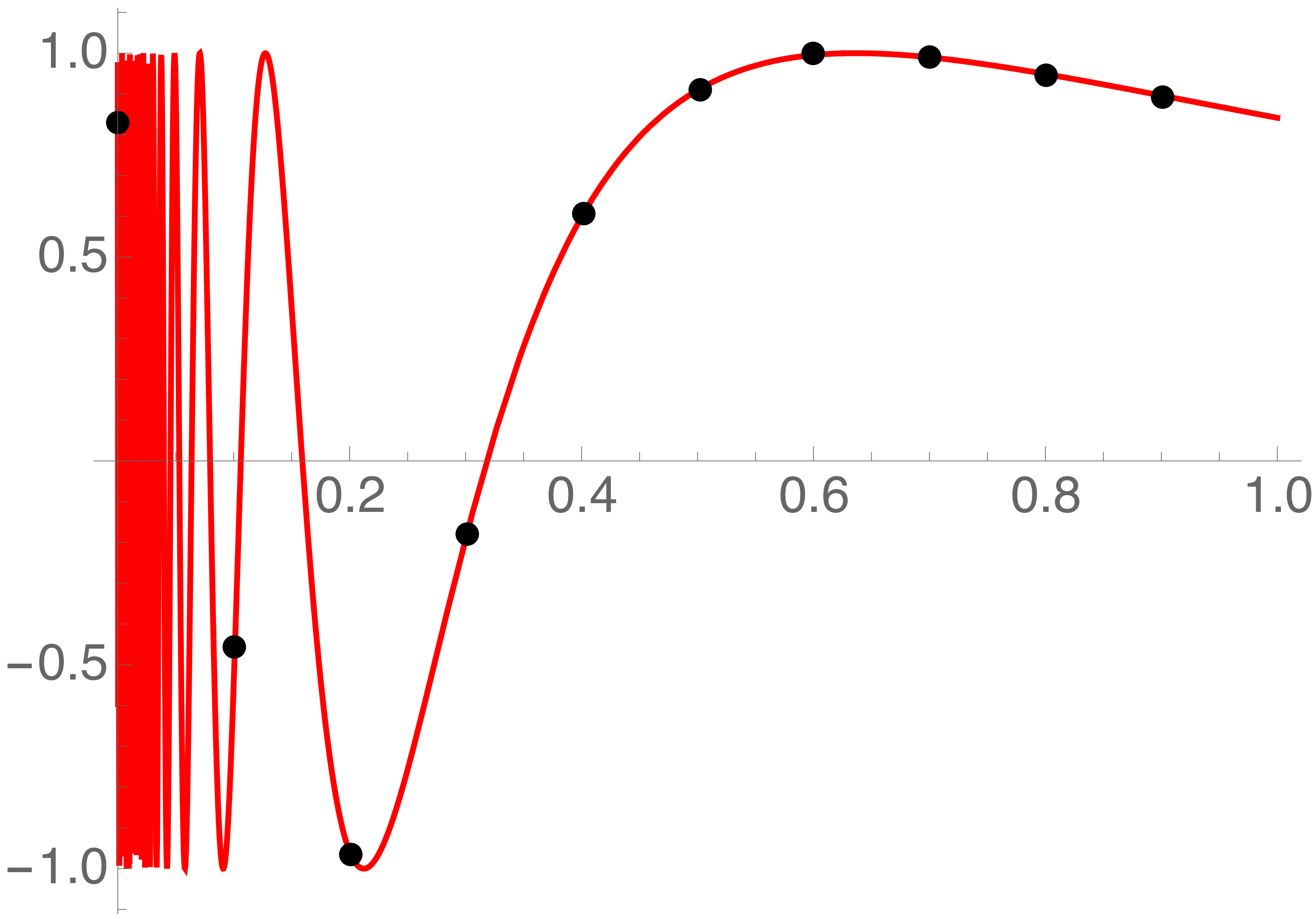
$$\frac{dy}{dt} \approx \frac{y_{i+1} - y_i}{\delta t}$$

$$\frac{dy}{dt} \approx \frac{y_{i+1} - y_{i-1}}{2\delta t}$$



Detect scaling by plotting on log scales

$$\sin(1/x)$$



$\sin(1/x)$

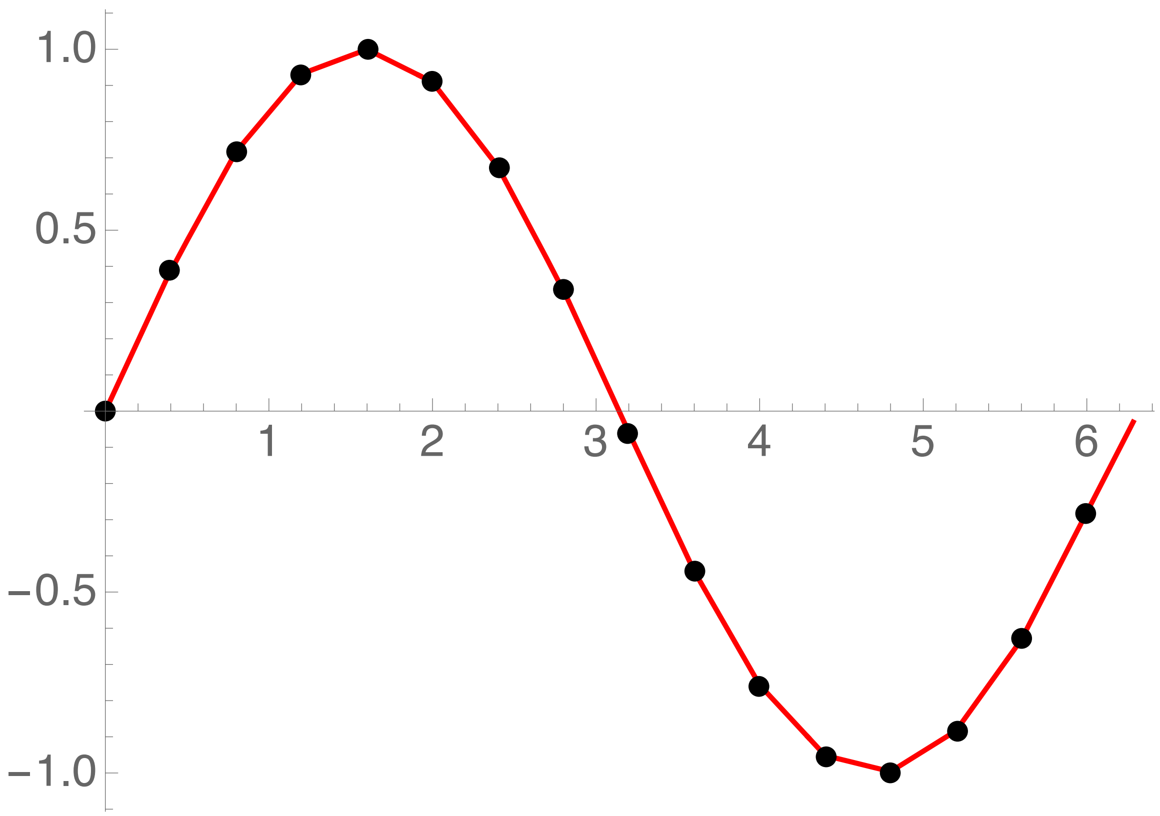
Error

Order

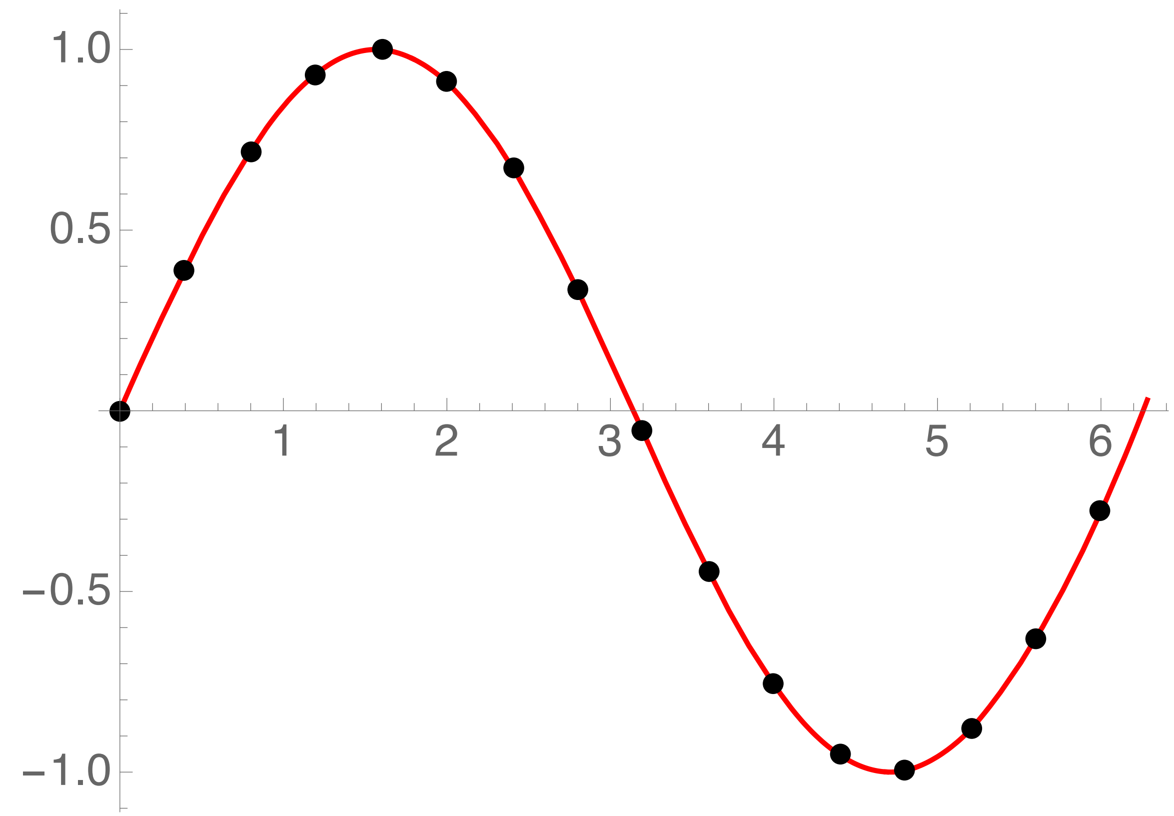
Cost

	$\frac{dy}{dt} \approx \frac{y_{i+1} - y_i}{\delta t}$	$\frac{dy}{dt} \approx \frac{y_{i+1} - y_{i-1}}{2\delta t}$
Error	δt	δt^2
Order	1	2
Cost	2 evaluations of y	2 evaluations of y

An alternative approach is to locally represent the function by a polynomial



Function is represented within each **element** by a linear function.



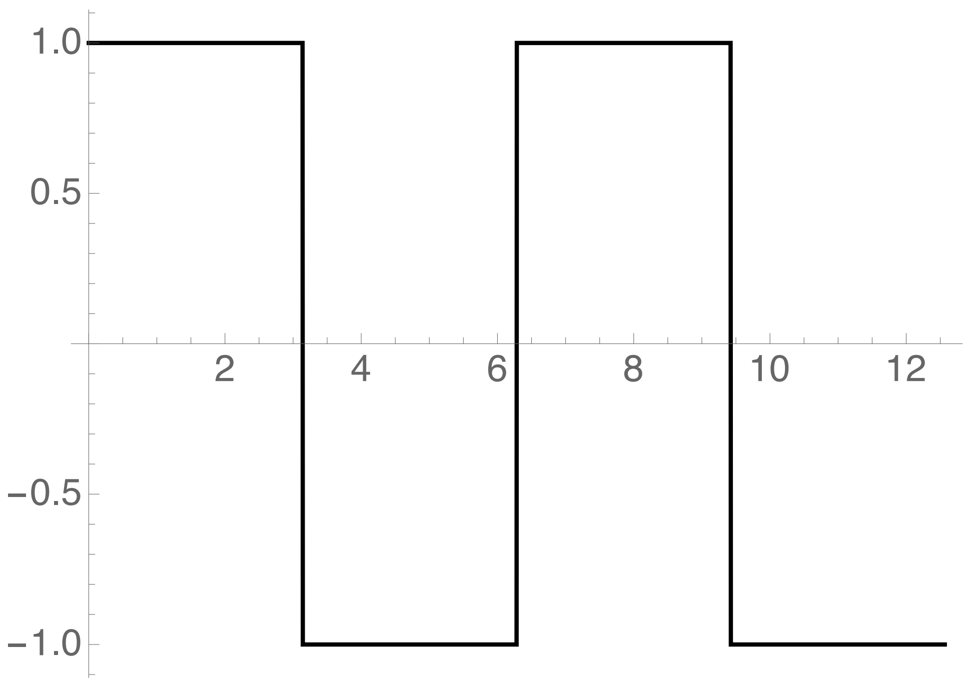
Quadratic

We’ll return to this idea later…

It’s useful for **interpolation** of functions from tabulated data.

It’s one of the key ideas behind **Finite element** solvers for PDEs.

A third approach is to represent the function through a set of canonical functions



Consider a **periodic** function. It makes sense to represent this using other periodic functions; a natural choice are trigonometric functions.

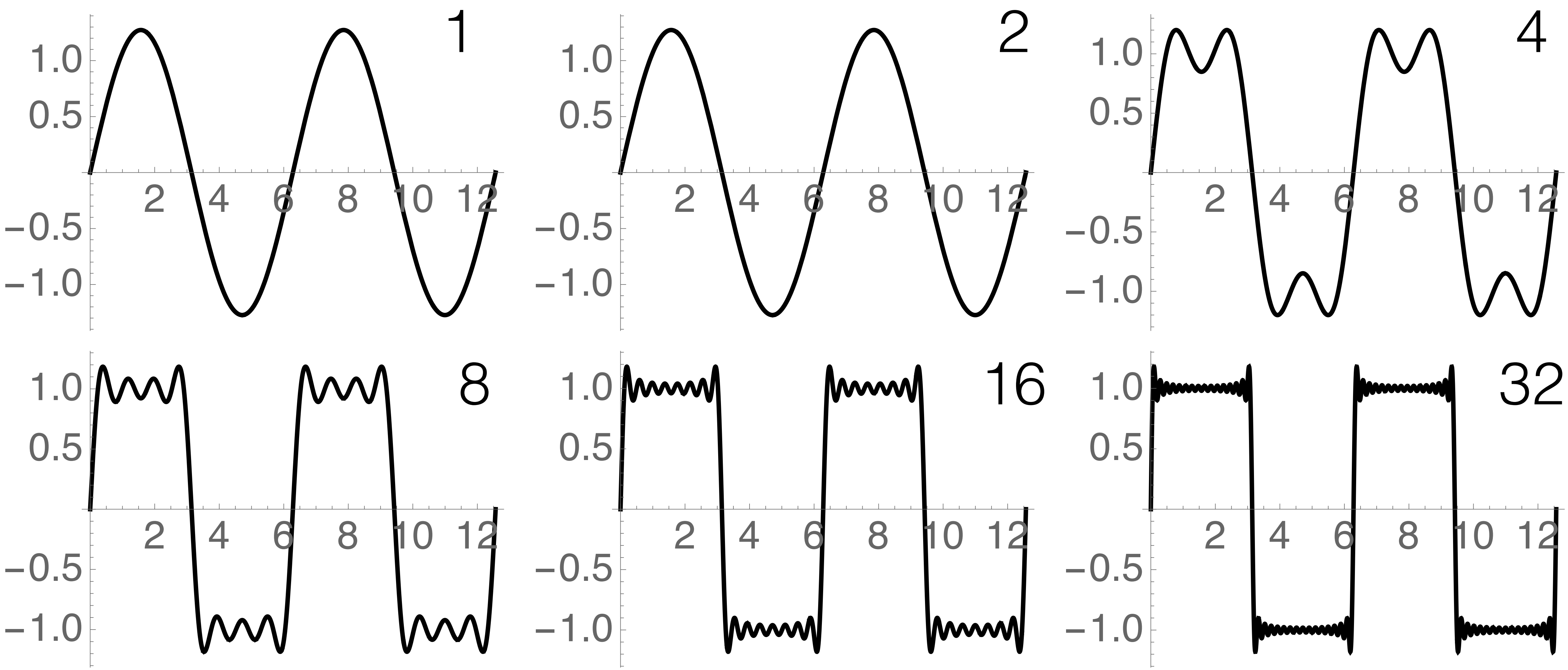
$$f(x) \approx \sum_{n=1}^{\infty} c_n \sin(nx)$$

This is a **Fourier series**.

$$c_n = \frac{2}{\pi} \int_0^{2\pi} f(x) \sin(nx) \mathrm{d}x$$

$$c_n = \langle f, \psi_n \rangle \longleftarrow **$$

Increasing number of terms improves the representation, but reveals pathology.

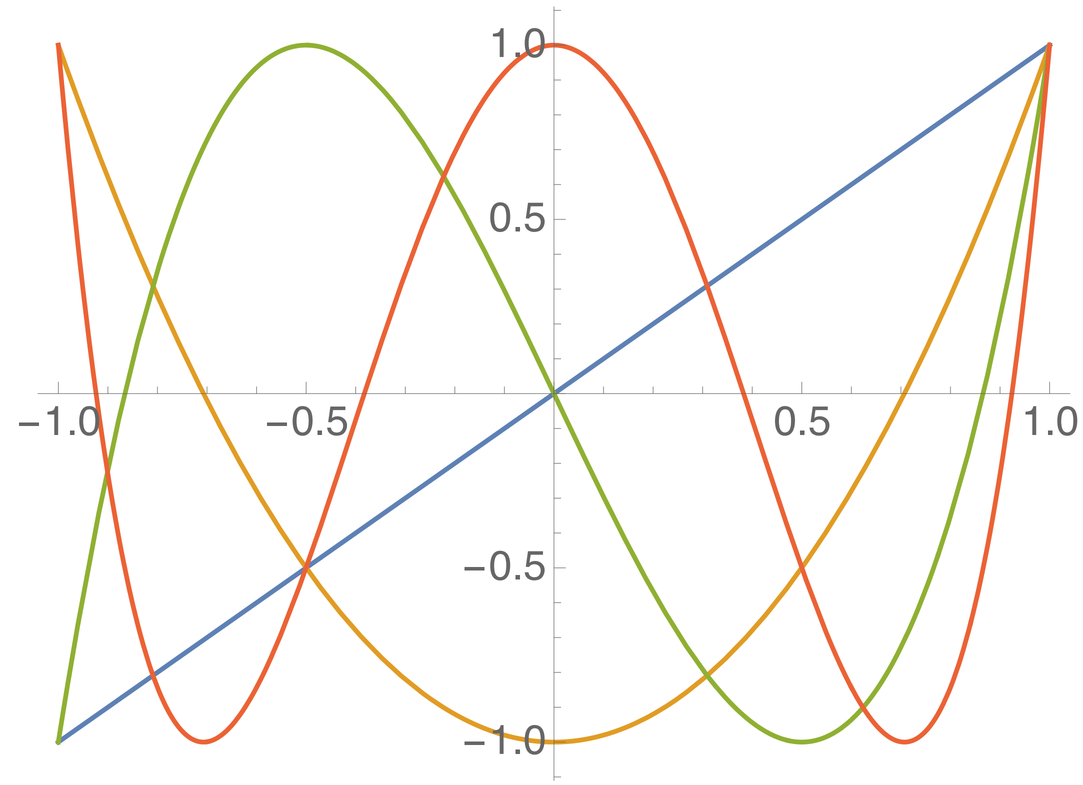


***

A particularly attractive set of canonical functions are the **Chebyshev polynomials**.

These are the first few…

$$x, 2x^2 - 1, 4x^3 - 3x, 8x^4 - 8x^2 + 1$$



They are defined by…

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

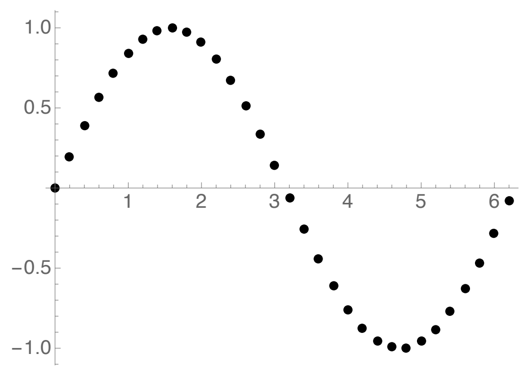
Fascinatingly,

$$T_n(x) = \cos(n \arccos x)$$

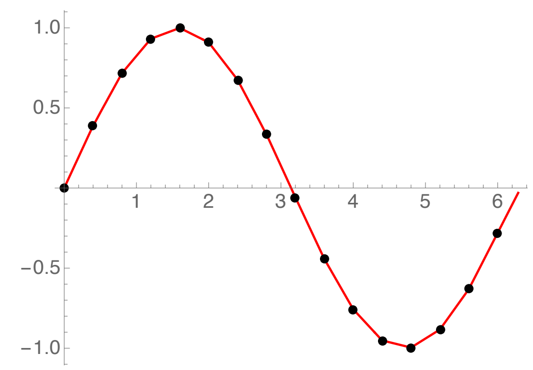
They are **orthogonal** and **normalizable**.

Summary:

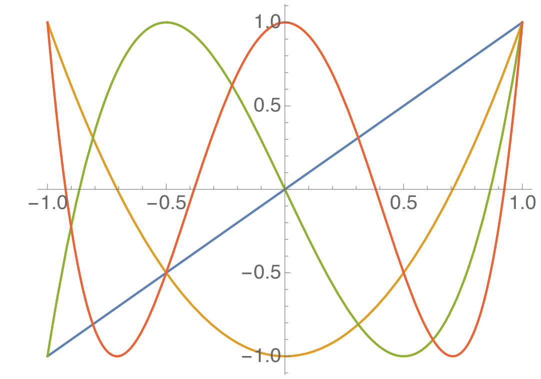
We can represent a function by:—



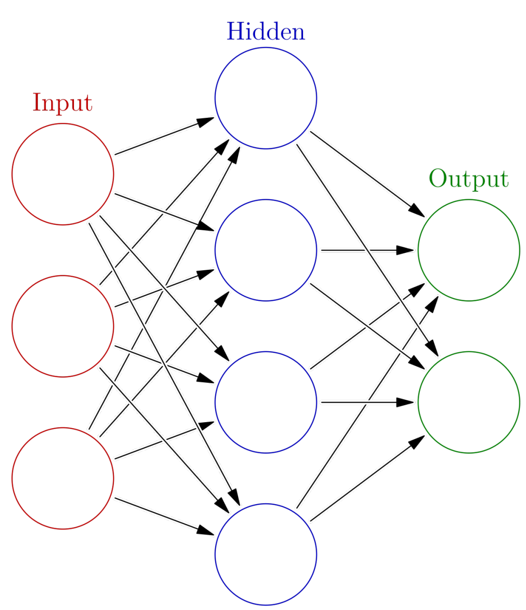
Tabulated points



Local polynomial approximants



Canonical functions



(A neural network)

Objectives for today

- Introduce Tracker, which allows us to get the trajectory of our object from the video.
- Get Tracker data into our notebook.
- Introduce some tools to assess the performance of numerical algorithms.