CS URA Program

Department of Computer Science Undergraduate Research Assistantships (URA)

What's a URA?

A URA is an opportunity for an exceptional student to help a professor with a research project. The student is paid $600 per term in exchange for six hours of work per week over a period of ten weeks.

What's in it for the student?

The student gets some real research experience and earns some pocket money. The experience should also be very useful for helping the student decide whether to pursue graduate studies in Computer Science.

Who's eligible?

Any student who has completed their second year in the Faculty of Mathematics with an average of at least 80% is eligible. Preference is given to students enrolled in Computer Science major plans. A student can only do one URA per term.

What does the professor have to do?

The professor should email a three-line description of a project to cs-ug-research@cs.uwaterloo.ca, so it can be posted on the website ***at least one week before the beginning of lectures.*** The professor commits to supporting the student with $600.00 of his/her research funding.

What does the student have to do?

Find a project they would be interested in working on by visiting www.cs.uwaterloo.ca/current/enrichment/ura.shtml. The student should send a resume and recent grade report to the professor and make an appointment with him/her during the first two weeks of the term to discuss the opportunity.

The agreement

The student and the professor will reach a joint agreement concerning the details of the project. After a verbal agreement is reached, the student or professor should return completed URA forms to MC 4036 so they can be processed.

Questions?

Email cs-ug-research@cs.uwaterloo.ca or visit MC 4036.

Interpolation – Piecewise Polynomials and Cubic Splines

CS370 – May 11, 2016

Lagrange polynomials: Line example

$$p(x) = \sum_{k=1}^{n} y_k L_k(x) \text{, where } L_k(x) = \frac{(x - x_1)(\dots)(x - x_{k-1})(x - x_{k+1})(\dots)(x - x_n)}{(x_k - x_1)(\dots)(x_k - x_{k-1})(x_k - x_{k+1})(\dots)(x_k - x_n)}$$

Consider the two points we fit a line to earlier:

$$(x_1, y_1) = (1,2), (x_2, y_2) = (-1,4)$$

What are the corresponding L_k, and polynomial $p(x)$?

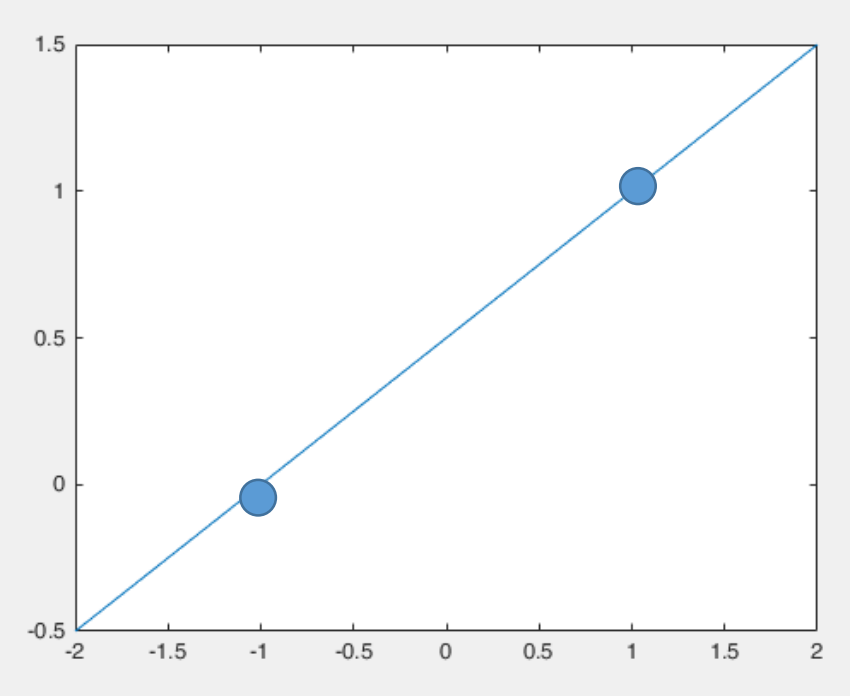
$$L_1(x) = \frac{x--1}{1--1} = \frac{x+1}{2}, \qquad L_2(x) = \frac{x-1}{-1-1} = \frac{x-1}{-2}$$

$$p(x) = 2L_1(x) + 4L_2(x) = (x + 1) - 2(x - 1) = -x + 3$$

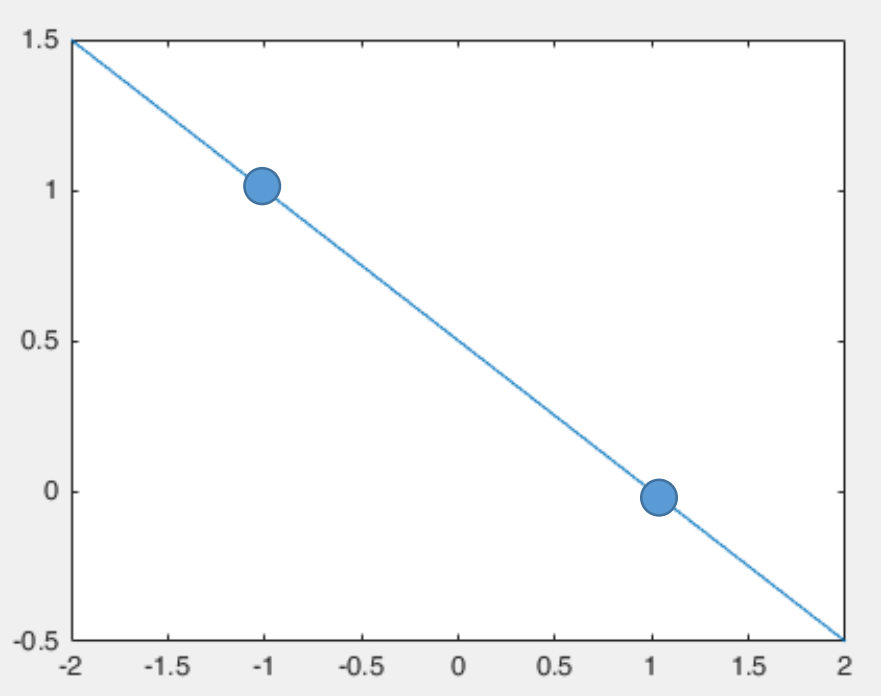
Same as before, just derived differently.

Lagrange polynomials visualized

Each $L_i(x)$ evaluates to 1 at x_i, 0 at the rest.



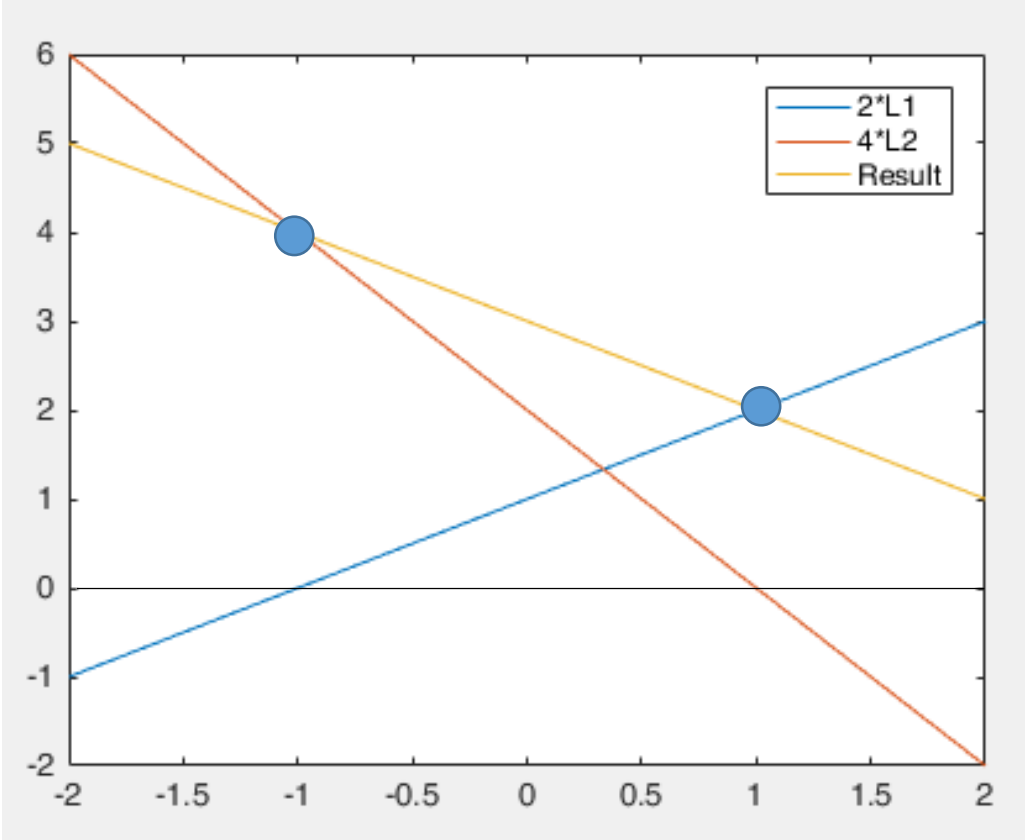
$L_1(x)$



$L_2(x)$

Lagrange polynomials visualized

Scaling each by its y-coordinate and summing gives the desired interpolant.



Another Lagrange polynomial example

$$p(x) = \sum_{k=1}^{n} y_k L_k(x)\,, \text{where } L_k(x) = \frac{(x - x_1)(\dots)(x - x_{k-1})(x - x_{k+1})(\dots)(x - x_n)}{(x_k - x_1)(\dots)(x_k - x_{k-1})(x_k - x_{k+1})(\dots)(x_k - x_n)}$$

What are the first two Lagrange basis functions we would use to fit the 4 points:

(-2,1), (1,3), (3,2), (4,-1)

Give $p(x)$ in terms of the basis functions, $L_1(x), L_2(x), L_3(x), L_4(x)$.

Lagrange polynomials – why?

Q: If it's the same polynomial in the end, why might we prefer the Lagrange basis to the monomial basis?

A: We can directly write down the polynomial from the Lagrange basis functions, L_k, and the data points, (x_i, y_i).

No need to solve a linear system!

Summary – Polynomial Interpolation

- Interpolation allows us to approximate *unknown* functions given discrete data.

- Fitting a (low-degree) polynomial function to a small set of points is a fundamental operation, with many applications.

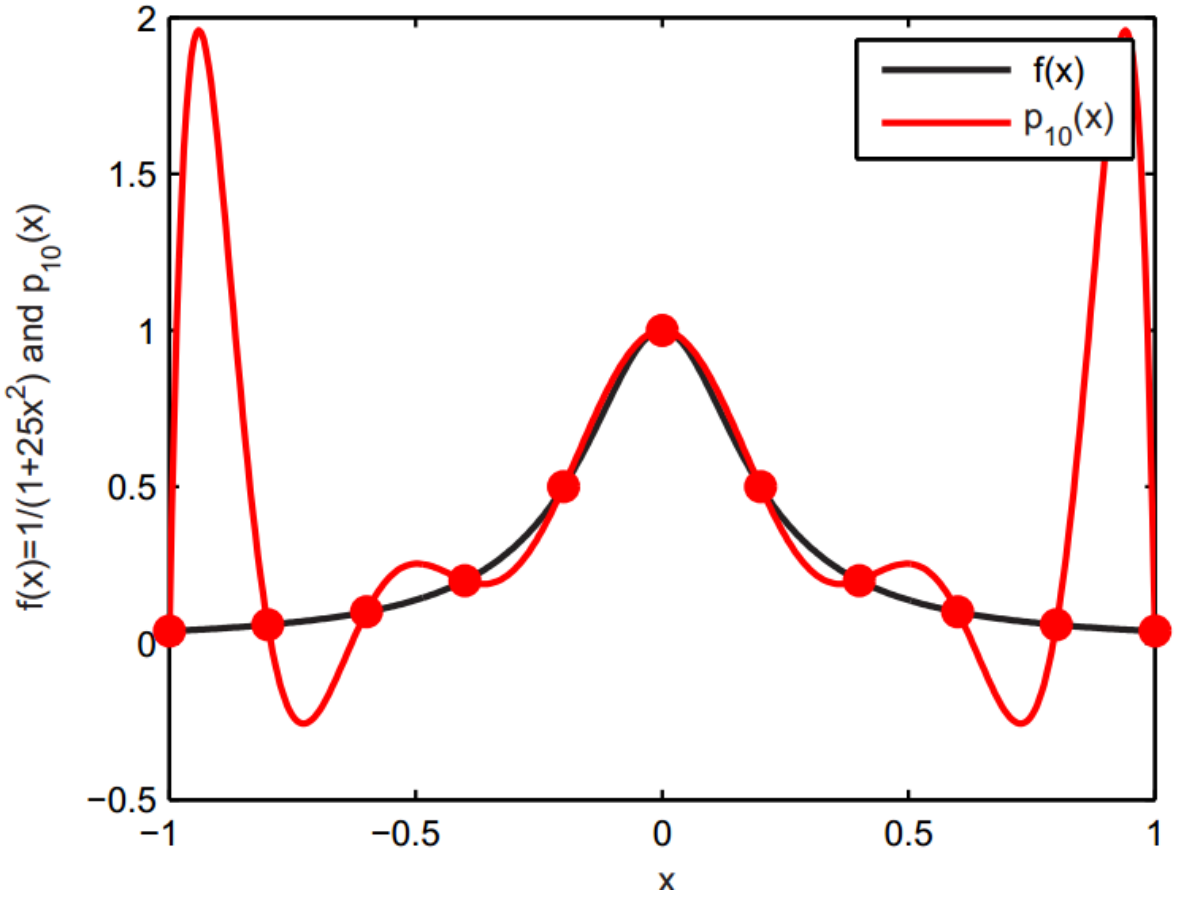
- An interpolating polynomial function can be found either by
  1. Solving a system of linear equations for the coefficients, or
  2. Constructing it as a sum of *Lagrange basis* functions.

Polynomial interpolation for *many* points?

Consider fitting a degree 10 polynomial to 11 points sampled from $f(x) = \frac{1}{1+25x^2}$.

Is this a "good" approximation?

What do we observe?

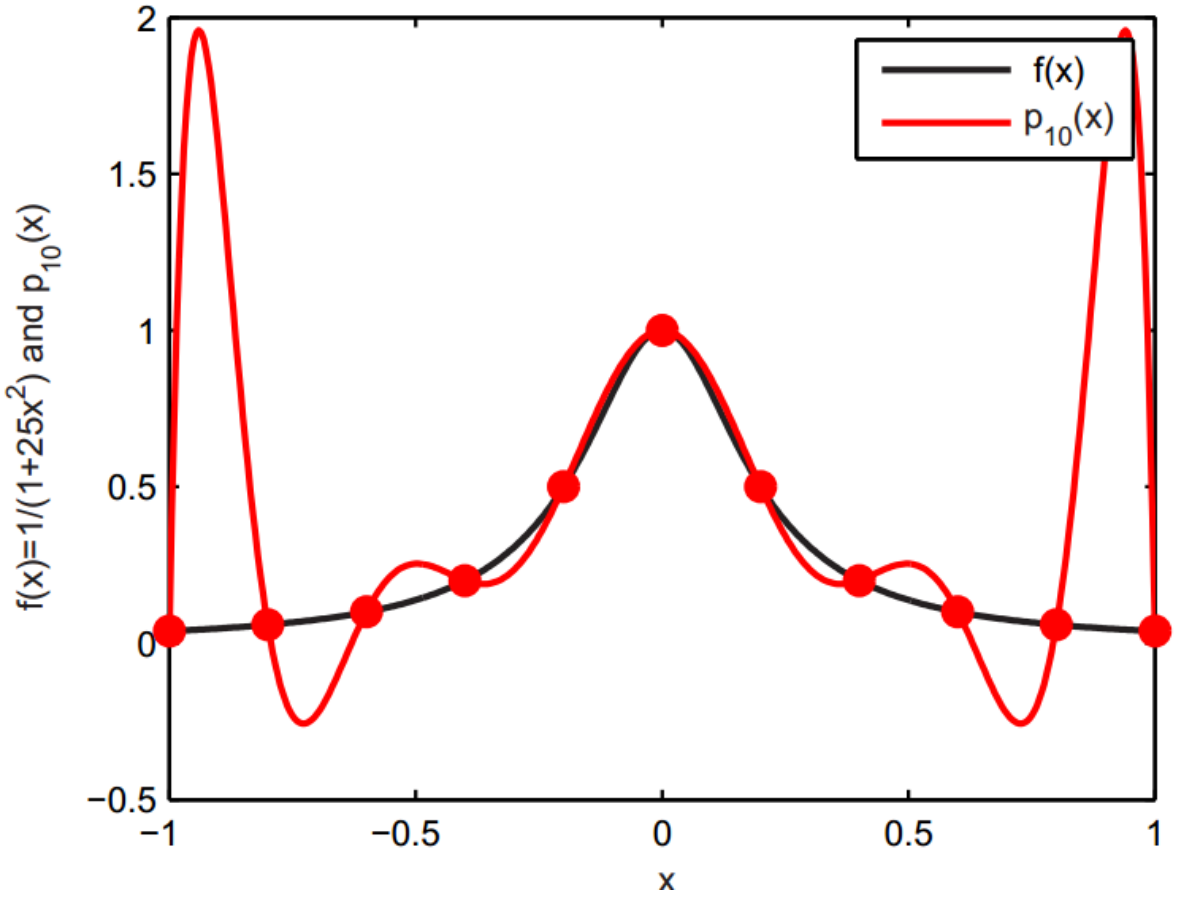


Polynomial interpolation for *many* points?

Fitting a "high degree" ($n \approx 5, 6$, and up) polynomial gives excessive oscillation/"wiggling".

Called *Runge's phenomenon*.

Moral: **Going to higher degrees doesn't necessarily yield a good quality fit…**



Strategies to address/avoid this?

- Select data/interpolation points in a “smarter” way.
  - Notice that the oscillations get worse at the edges…
  - Only works for settings where *you* can choose the points.

- Fit even higher degree polynomials, but constrain *derivatives* to somehow enforce “smoothness”.

- Fit lower degree polynomials that don’t exactly interpolate, but minimize some error measure.
  - i.e., find an approximate function.

- Use **piecewise polynomials!** → Focus of today’s lecture.

Piecewise functions

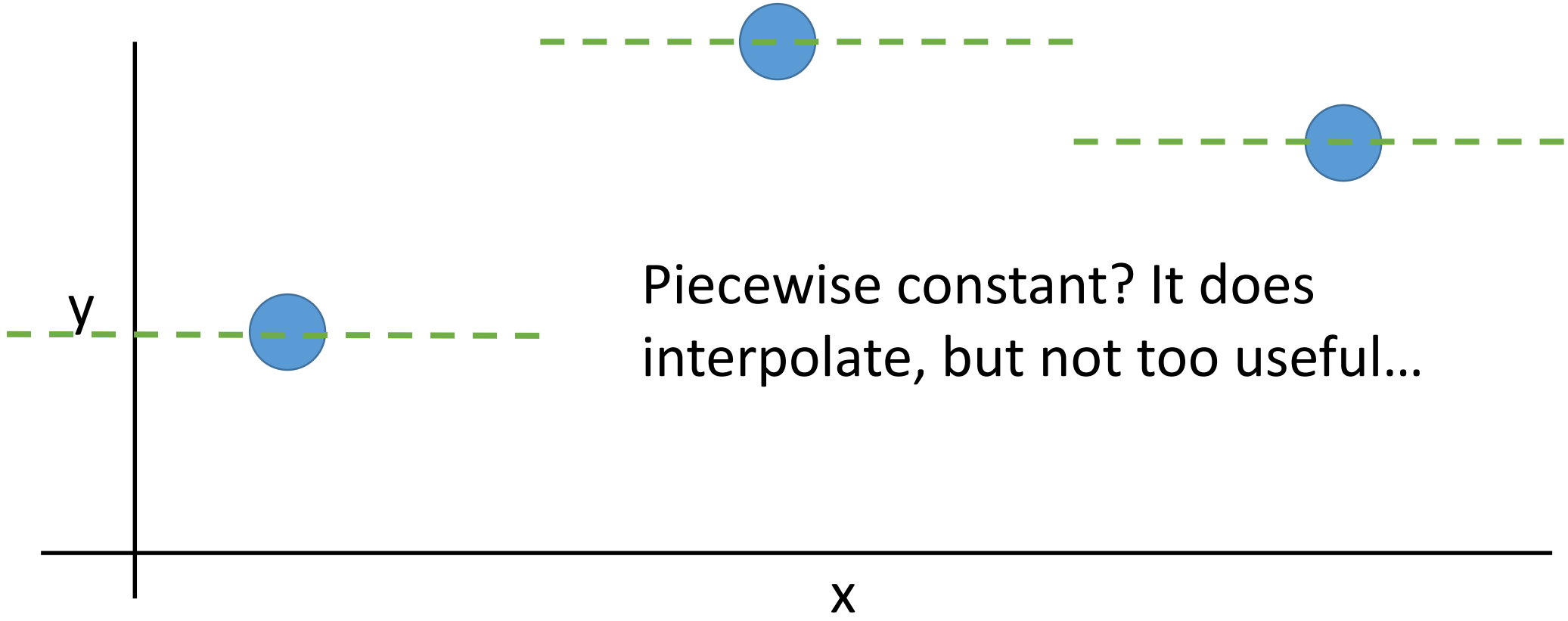
Recall that a *piecewise* function is a function with different definitions for distinct intervals of the domain.

e.g.,

$$p(x) = \begin{cases} 3 & \text{if} \quad x \leq -1 \\ x^2 & \text{if} \quad -1 < x \leq 2 \\ -x + 3 & \text{if} \quad 2 < x \end{cases}$$

Interpolating with Piecewise Functions

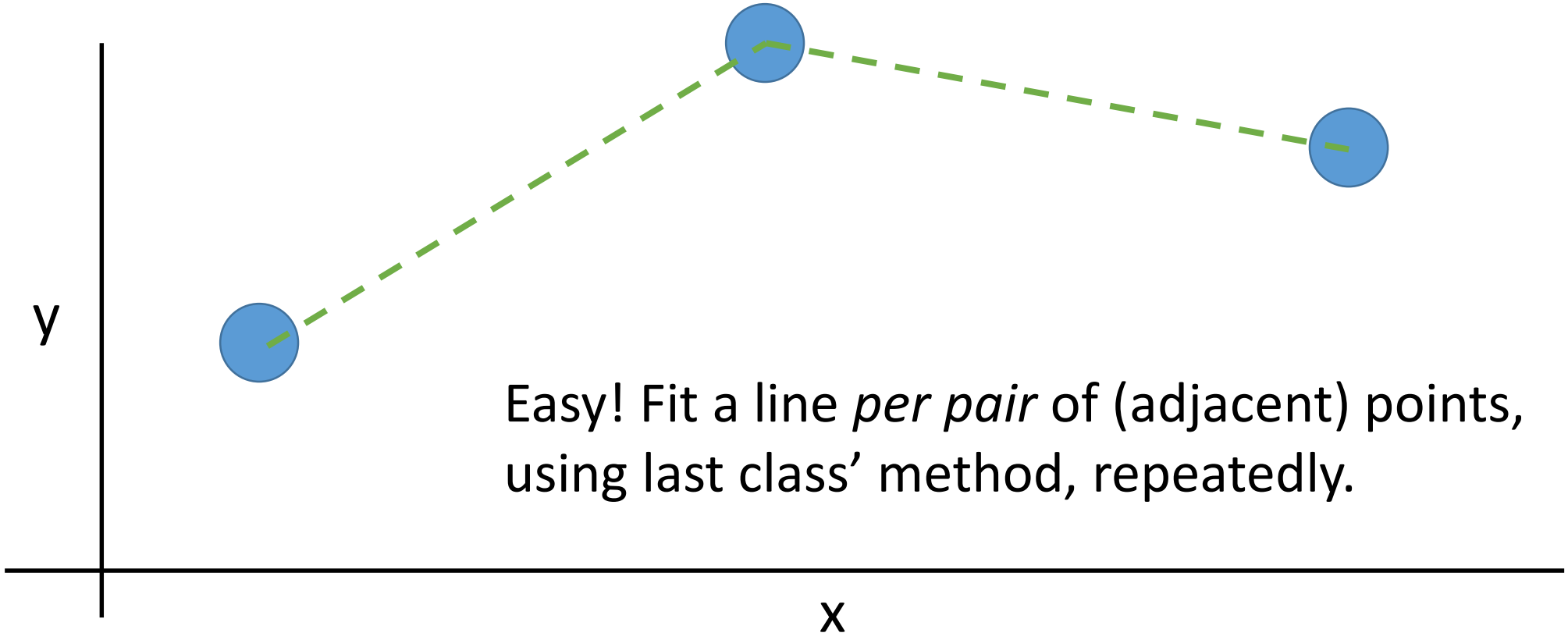
Simplest *piecewise* function that interpolates a given set of points?



Piecewise constant? It does interpolate, but not too useful…

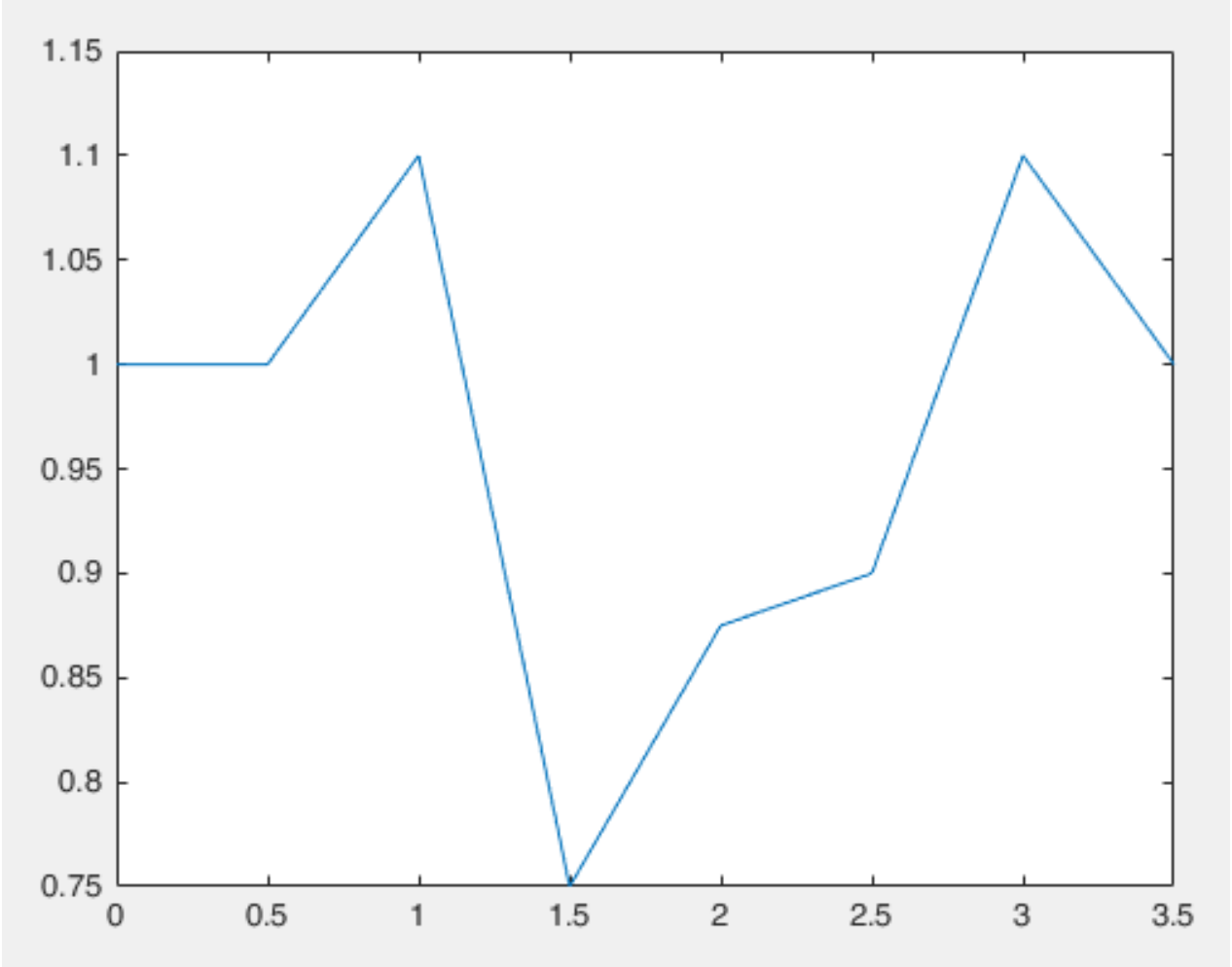
Interpolating with Piecewise Functions

Usually *at least* want ***continuity*** (i.e. no jumps). So try piecewise *linear* functions. How do we construct such a function?



Easy! Fit a line *per pair* of (adjacent) points, using last class' method, repeatedly.

Example – Piecewise Linear Interpolant



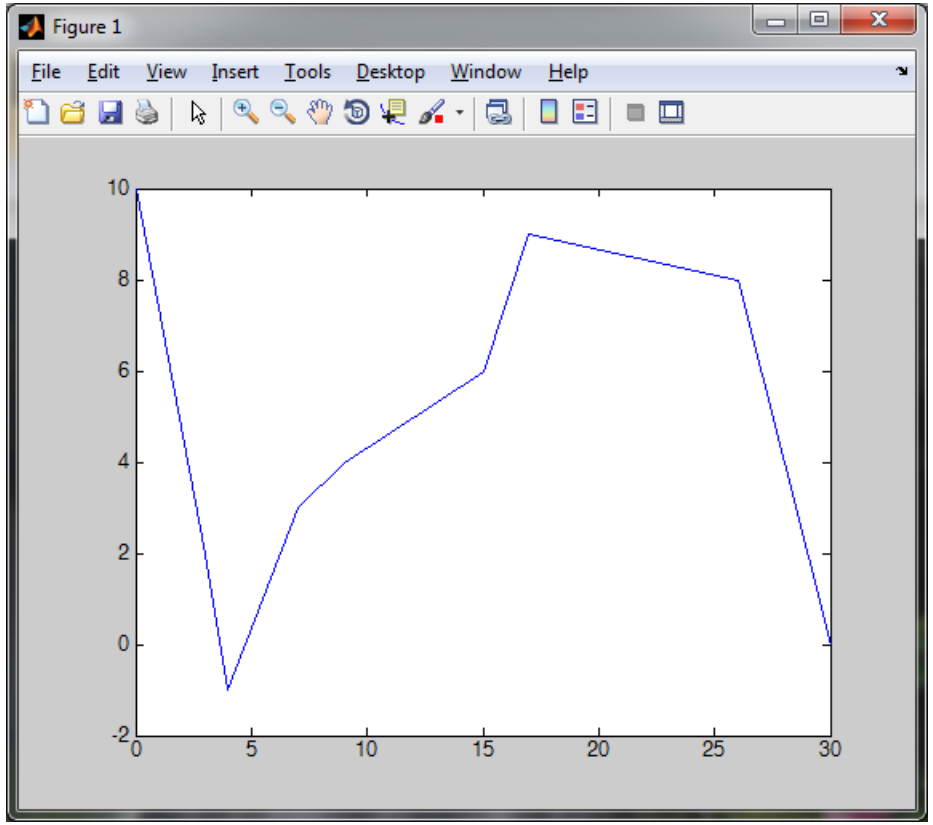
$$p(x) = \begin{cases} 1 & 0 \le x \le \frac{1}{2} \\ \frac{x}{5} + \frac{9}{10} & \frac{1}{2} \le x \le 1 \\ \frac{-7x}{10} + \frac{9}{5} & 1 \le x \le \frac{3}{2} \\ \cdots & \cdots \\ \cdots & \cdots \\ \frac{-x}{5} + \frac{17}{10} & 3 \le x \le \frac{7}{2} \end{cases}$$

Transition points have the same value, so the overall piecewise function is continuous.

Example: Matlab

If I give Matlab a set of points specified by two length n vectors $\vec{x}$ and $\vec{y}$, and plot it, it defaults to drawing a piecewise linear curve.

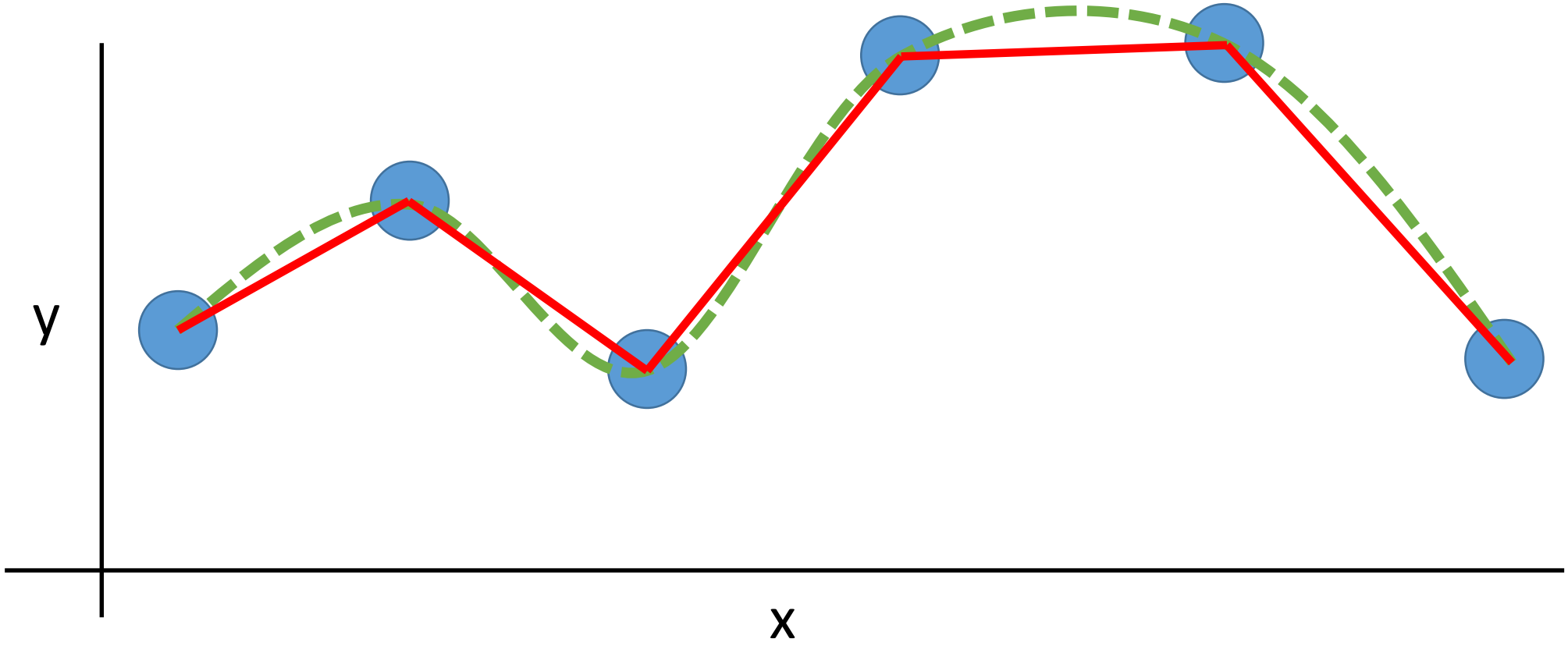
```
>> x = [0 3 4 7 9 15 17 26 30];
>> y = [10 2 -1 3 4 6 9 8 0];
>> plot(x,y)
```



Interpolating with Piecewise Functions

Often, piecewise linear is still not satisfactory…

We may want to avoid “kinks” at the data points.



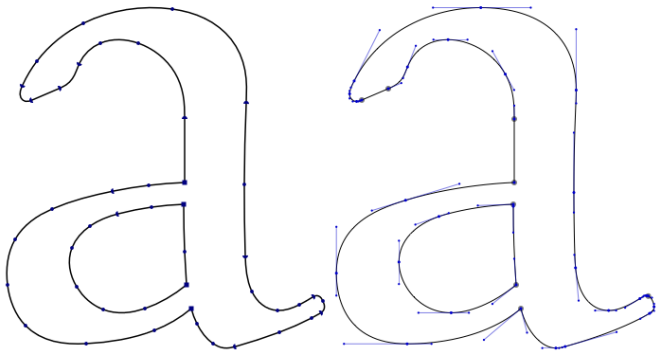
Smoothness

We want greater *smoothness*. (i.e., continuity of *derivatives*). Why?

e.g.

- For aesthetic purposes, in graphics, design, and data visualization.
- For mathematical/numerical applications needing (approximate) derivative information.

Piecewise linear functions are called C^0 – function is continuous, but derivatives are (generally) not.



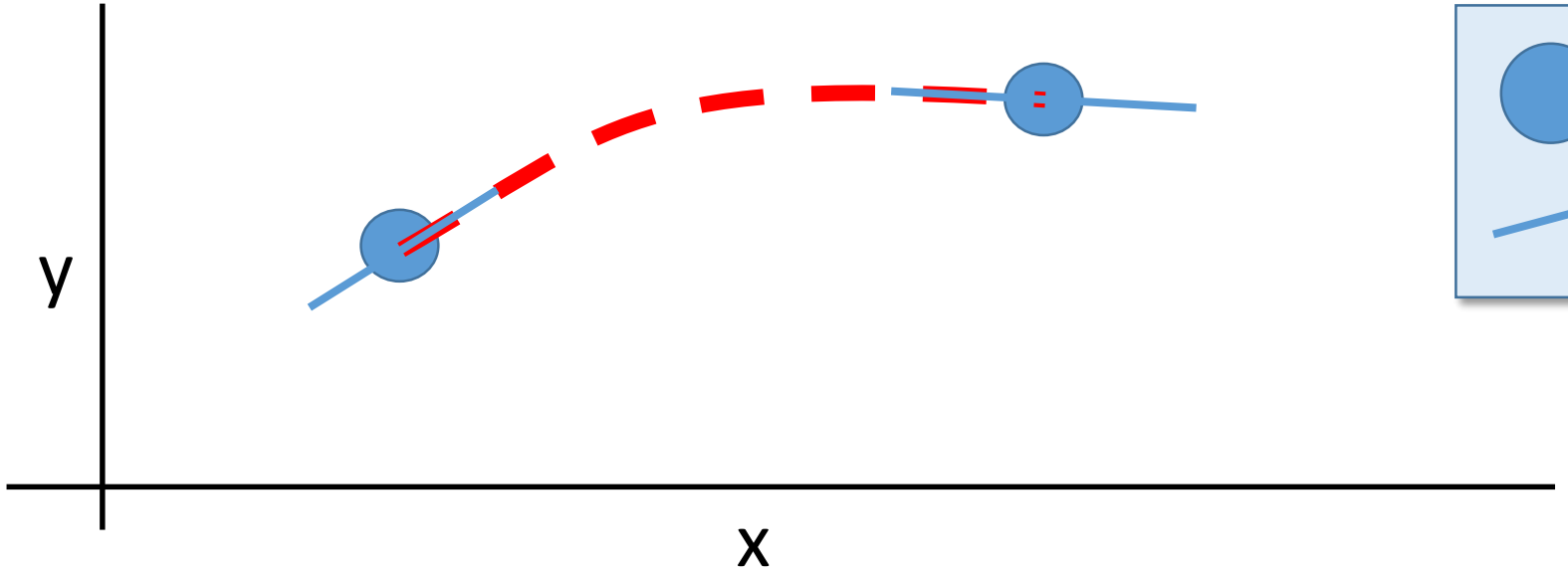
e.g. a smooth font designed using a few points.

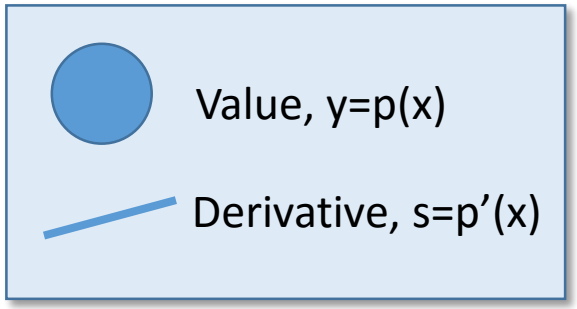
Hermite Interpolation

Charles Hermite

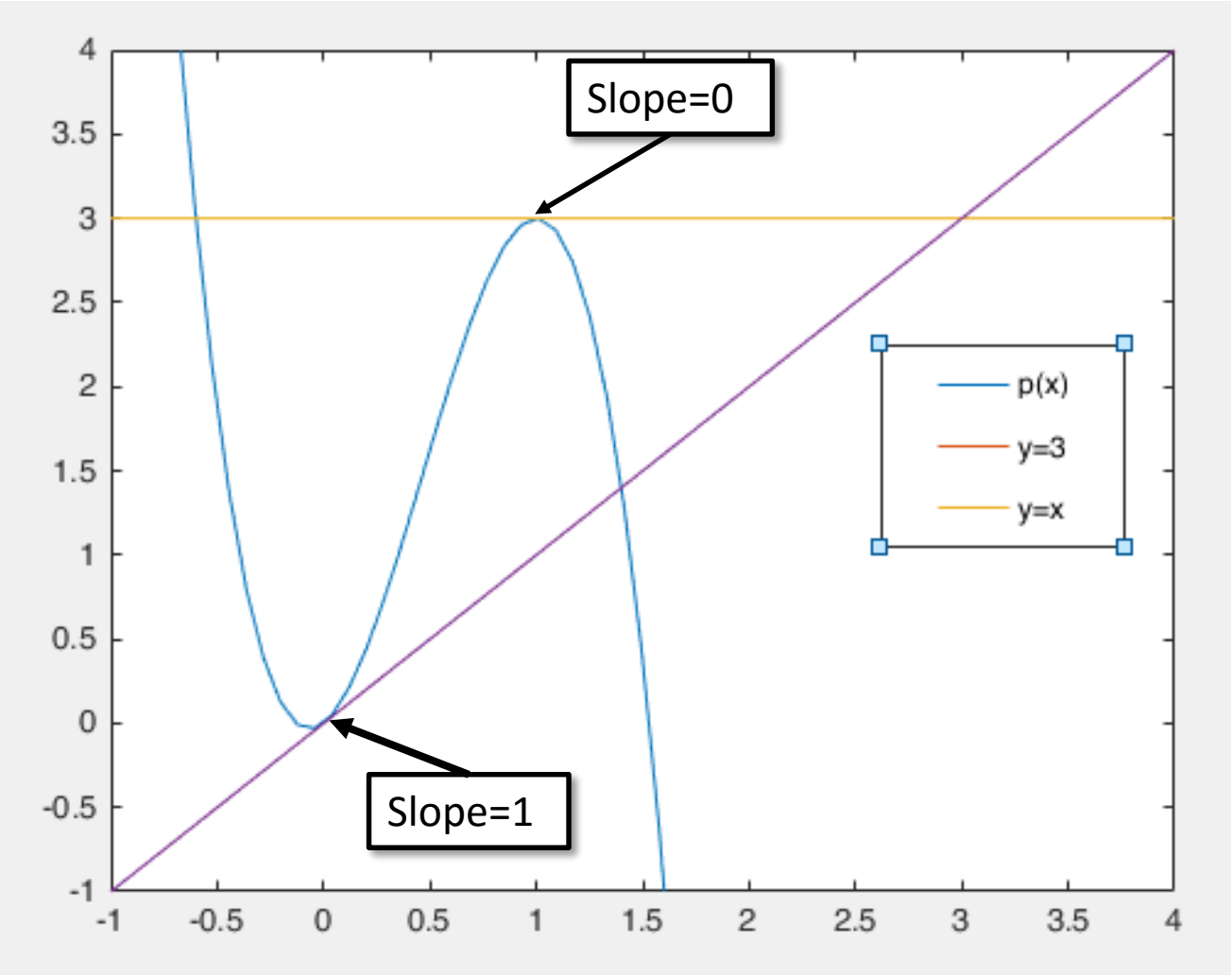
Greater smoothness requires controlling *derivatives* of the polynomial.

"Hermite interpolation" is the problem of fitting a polynomial given function values ***and derivatives***.





Hermite interpolation – Example



$p(x) = x + 7x^2 - 5x^3$
satisfies…
$p\,(0) = 0$
$p'(0) = 1$
$p\,(1) = 3$
$p'(1) = 0.$

Aside: Piecewise Quadratics?

Why did we jump to cubics for the 2-point Hermite interpolation problem? Why not quadratics?

Quadratics have only 3 coefficients per interval.

→Insufficient # of unknowns to satisfy ***both*** 2 values and 2 derivatives.

[Quadratics can still be used for related problems; the course notes include a practice question involving a *quadratic spline*.]

Fitting polynomials with higher derivative data

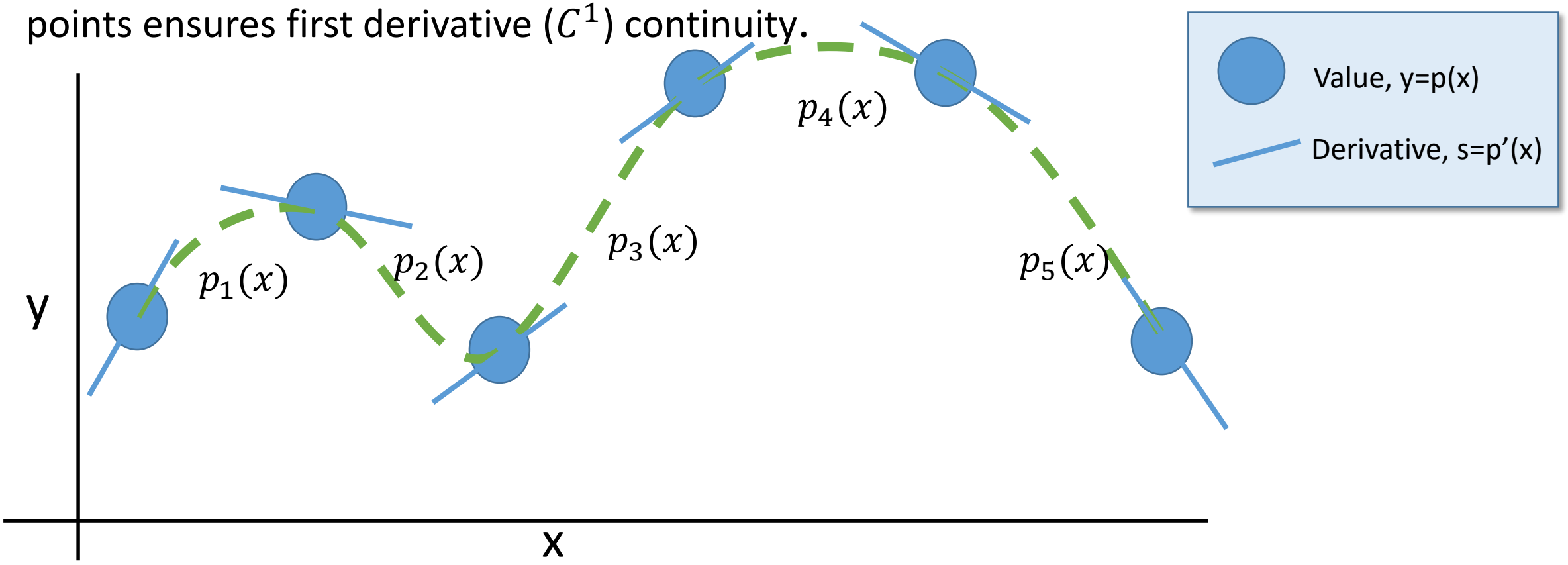
Same basic strategy can allow fitting higher degree polynomials, or fitting with higher derivative data.

1. Work out the required derivatives of $p(x)$ in terms of (unknown) polynomial coefficients.
2. Plug in the given value and derivative data.
3. Solve for the polynomial coefficients.

Hermite interpolation – Many points

Fit many points (given values **and** 1st deriv.) with ***piecewise*** Hermite interpolation?

Use ***one cubic per pair*** of (adjacent) points. The matching/shared derivative data at points ensures first derivative (C^1) continuity.



Hermite interpolation – General solution

If we (instead) define the polynomial on the i^{th} interval, $p_i(x)$, as

$$p_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

there exist *direct* formulas for the polynomial coefficients:

$$a_i = y_i$$

$$b_i = s_i$$

$$c_i = \frac{3y_i' - 2s_i - s_{i+1}}{\Delta x_i}$$

$$d_i = \frac{s_{i+1} + s_i - 2y_i'}{\Delta x_i^2}$$

where we define

$$\Delta x_i = x_{i+1} - x_i$$

Spacing of points in x.

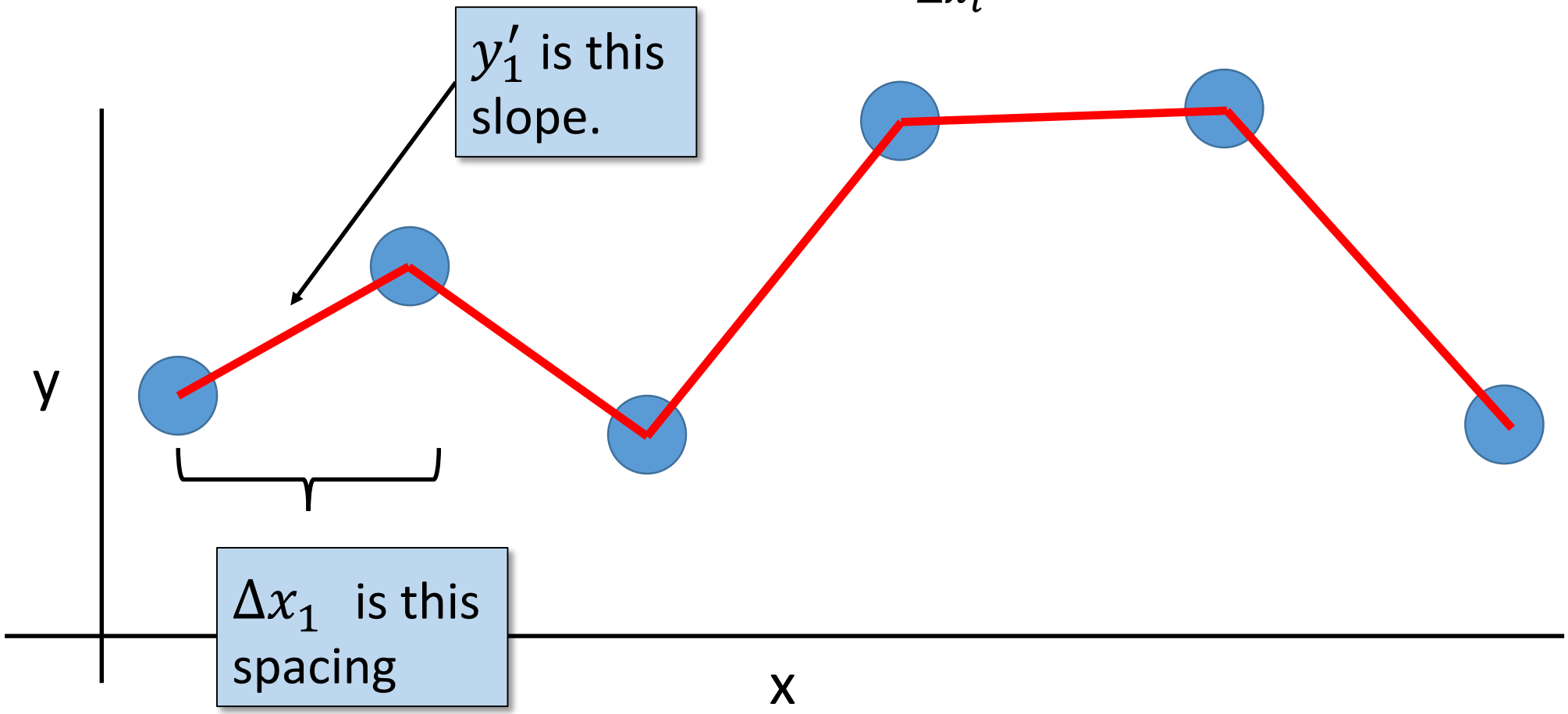
and

$$y_i' = \frac{y_{i+1} - y_i}{\Delta x_i}$$

Slopes for a piece-wise *linear* fit.

Hermite interpolation – General solution

We defined $\Delta x_i = x_{i+1} - x_i$ and $y_i' = \frac{y_{i+1} - y_i}{\Delta x_i}$.



Some Terminology

Knots: Points where the interpolant transitions from one polynomial / interval to another.

Nodes: Points where some control points/data is specified.

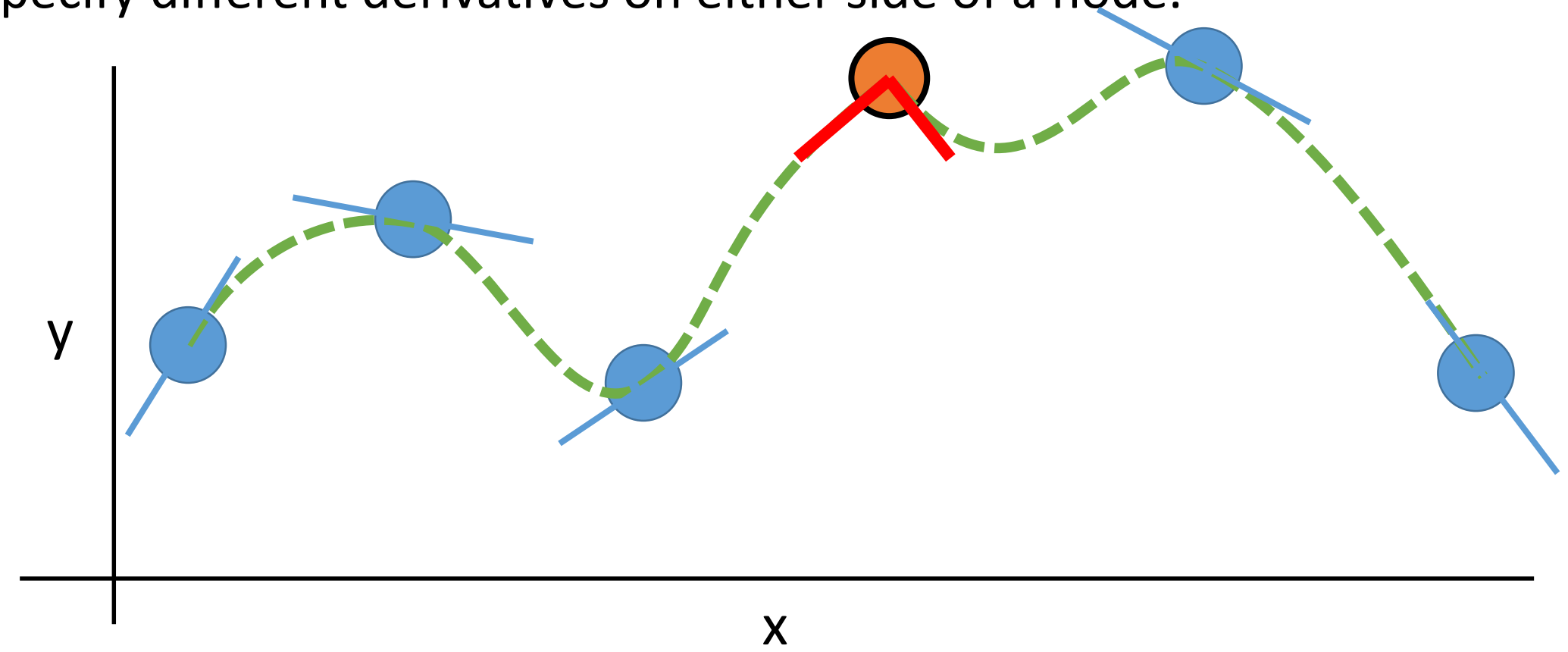
For Hermite interpolation, these are the same.

For other curve types, they may differ…

Hermite interpolation - Kinks

For some applications, we might still **intend** to create/control kinks. How?

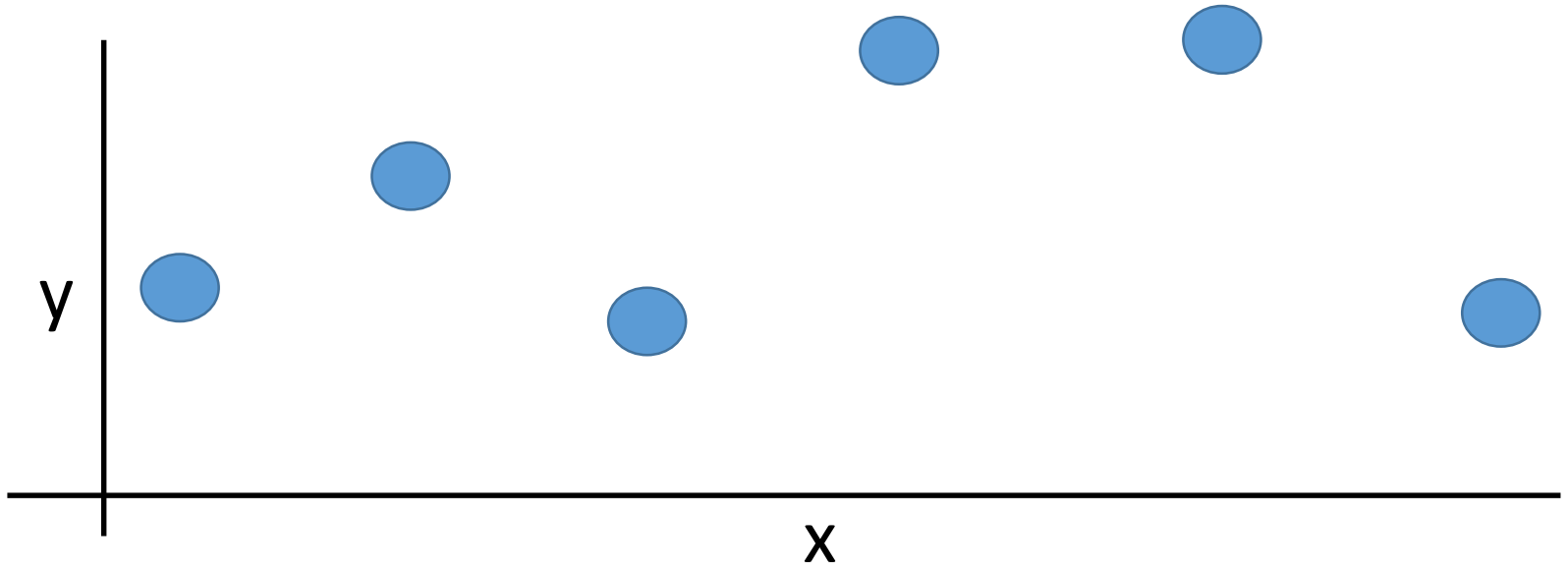
Specify different derivatives on either side of a node.



Piecewise cubic (spline) interpolation

More common setting: ***no derivative information*** is given, just points.

Can we still fit a piecewise cubic to the set of points?



Yes, but each “piece” needs data from more than just its 2 endpoints…

The intervals cannot be computed independently!

Physical (Drafting) Splines

spline (n.)
long, thin piece of wood or metal, 1756, from East Anglian dialect, of uncertain origin.

Drafting “whales” or “ducks”