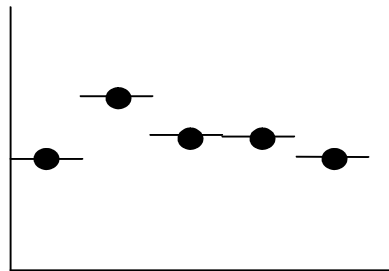


# CS205 – Class 14

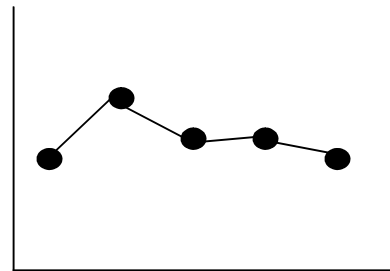
Covered in class: 1, 3, 4  
Readings: 7.4, 8.1 to 8.3

## Interpolation

1. A better solution is to use **piecewise polynomials** – a different polynomial in each subinterval  $[x_i, x_{i+1}]$ 
  - a. Simplest is piecewise constant, next is linear. As order increases number of points required increases but the accuracy also increases.
  - b. One could argue you should go to even higher order.



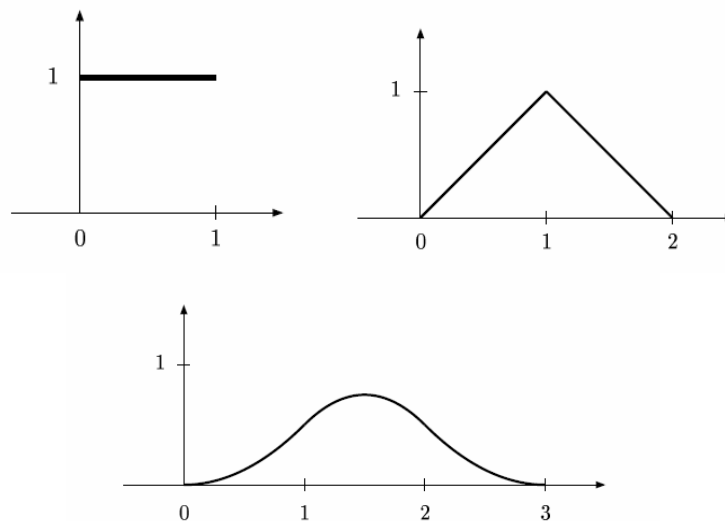
$O(\Delta x)$  Piecewise constant  
Piecewise constant



$O(\Delta x^2)$  Piecewise linear  
Linear

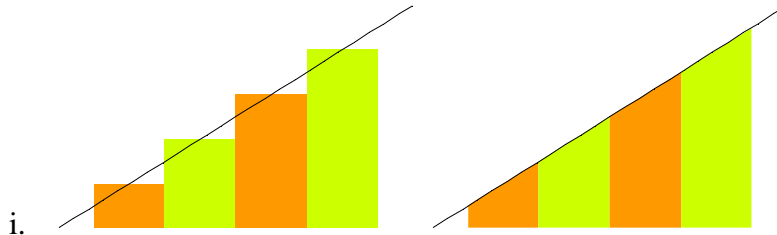
- c. Higher order is not necessarily better
    - i. Once you go to spectral you get infinite accuracy, better than any polynomial, but you need to pay with requiring more data and Gibbs' phenomena. There is some smoothness assumption about the function you are interpolating.
    - ii. In practice higher order methods overly smooth discontinuous phenomena. So they are good for smoother phenomena like simulating tree sap. However, if you had a turbulent flow the interesting and important discontinuities would get destroyed.
2. Spline interpolation, little detail here, but for more information Prof. Guibas teaches a course on it.
    - a. Defined using **control points**  $(x_i, y_i)$
    - b. **Piecewise linear** – connect the control points with straight lines
    - c. **Hermite interpolation** – specify the function values  $y_i$  and the derivatives  $y'_i$  at each control point
      - i. **Hermite cubic** – cubic polynomial on each subinterval  $[x_i, x_{i+1}]$
      - ii. If there are  $n$  control points and  $n - 1$  intervals, then there are  $n - 1$  cubics

- iii. We need to specify  $4(n-1)$  parameters, i.e. 4 parameters for each cubic
- iv. Interpolating the function values  $y_i$  gives  $2(n-1)$  conditions, i.e. 2 for each subinterval
- v. Requiring the derivative to be continuous is  $n-2$  conditions, one for each interior control point
- vi.  $4(n-1) - 2(n-1) - (n-2) = n$  more conditions need to be specified
- d. **Spline** a piecewise polynomial of degree  $k$  that is differentiable  $k-1$  times
- e. **Cubic spline** continuous 1<sup>st</sup> and 2<sup>nd</sup> derivatives at the control points
  - i.  $2(n-1)$  conditions to interpolate the  $y_i$
  - ii.  $n-2$  conditions for continuous 1<sup>st</sup> derivatives
  - iii.  $n-2$  conditions for continuous 2<sup>nd</sup> derivatives
  - iv. Total of  $4n-6$  conditions – we need 2 more conditions
    - 1. **Hermite cubic spline** specify the 1<sup>st</sup> derivative at  $x_1$  and  $x_n$  (endpoints)
    - 2. **Periodic cubic spline** forcing the 1<sup>st</sup> and 2<sup>nd</sup> derivatives to match at  $x_1$  and  $x_n$
    - 3. **Natural cubic spline** set the 2<sup>nd</sup> derivative to zero at  $x_1$  and  $x_n$
    - 4. Set up the equations and solve
- f. **B splines** the basis function  $B_i^k$  is a piecewise polynomial of degree  $k$ 
  - i. Piecewise constant  $B_i^0(x) = 1$  for  $x \in [x_i, x_{i+1})$  and 0 otherwise
  - ii. Recursively  $B_i^k(x) = v_i^k(x)B_i^{k-1}(x) + (1 - v_{i+1}^k(x))B_{i+1}^{k-1}(x)$  with linear functions  $v_i^k(x) = \frac{x - x_i}{x_{i+k} - x_i}$
  - iii.  $B_i^1$  are piecewise linear,  $B_i^2$  are piecewise quadratic,  $B_i^3$  are piecewise cubic, etc.



The  $B^0, B^1, B^2$  polynomials

3. **Numerical quadrature** approximate  $I = \int_a^b f(x)dx$  for a given  $f$ 
  - a. These  $f$ 's might be arbitrarily difficult to compute and only available by running a program.
  - b. General approach : Subdivide  $[a,b]$  into  $n$  intervals  $[x_i, x_{i+1}]$  with  $x_0 = a$  and  $x_n = b$  and consider each subinterval separately
4. **Newton-Cotes quadrature** for each subinterval  $[x_i, x_{i+1}]$ , choose  $n$  equally spaced points and use  $k-1$  degree polynomial interpolation to approximate the integral
  - a. Exact on polynomials of degree  $k-1$  when  $k$  is even, as expected
  - b. Exact on polynomials of degree  $k$  when  $k$  is odd, from symmetric cancellation



- c. **local accuracy** an exact method on  $k$  degree polynomials has a local error that scales like  $O(h^{k+2})$  in each subinterval where  $h$  is the length of the subinterval
- d. **global accuracy** since there are  $O(1/h)$  subintervals, the total error scales like  $O(h^{k+1})$ 
  - i. doubling the number of subintervals, sends  $h \rightarrow h/2$ , and reduces the error by  $(1/2)^{k+1}$
  - ii. order of accuracy is  $k+1$
- e. **Midpoint rule**  $M = \sum (x_{i+1} - x_i) f\left(\frac{x_i + x_{i+1}}{2}\right)$ 
  - i.  $k=1$  point, piecewise constant, exact for piecewise linear functions, 2<sup>nd</sup> order accurate
- f. **Trapezoidal rule**  $T = \sum \left(\frac{x_{i+1} - x_i}{2}\right) (f(x_i) + f(x_{i+1}))$ 
  - i.  $k=2$  points, piecewise linear, exact for piecewise linear functions, 2<sup>nd</sup> order accurate
- g. **Simpson's rule**  $S = \sum \left(\frac{x_{i+1} - x_i}{6}\right) \left(f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right)$ 
  - i.  $k=3$  points, piecewise quadratic, exact for piecewise cubic functions, 4<sup>th</sup> order accurate.
  - ii. Notice the huge jump from Trapezoidal rule. Basically Trapezoidal is where it should be and midpoint actually got promoted and here Simpson's got promoted.
- h. Careful not to evaluate endpoints twice - factor out  $h = x_{i+1} - x_i$  when possible

- i. Typically in most problems we don't get the extra order for free. Typically lots of work is spent just trying to get the simplest first order accurate methods to work. Then there are a bunch of researchers trying to extend those methods, and there is very few people working on getting up to typically third order accuracy.