Interpolation types

1. **Polynomials** (by far the most used)

   However, high degree polynomial interpolants are "bad": they oscillate too much, and roundoff error increases with N (condition # $\frac{2^N+1}{2}$)

2. Remedy: **piecewise** polynomial interpolation (low degree every few points)

   e.g. a line every 2 nodes, a parabola every 3 nodes, ...

   But these look bad, overall interpolant not smooth (pieces may join non-smoothly at knots)

3. **Bezier parametric curves**: piecewise (quadratics or) cubics in a parameter t, $0 \le t \le 1$ joined smoothly at knots (cont's derivative), passing through end points P_0, P_3 with "control points" P_1, P_2

$$\begin{cases} x(t) \\ y(t) \\ z(t) \end{cases} = \vec{B}(t) = \binom{3}{0}(1-t)^3 P_0 + \binom{3}{1}(1-t)^2 \cdot t \cdot P_1 + \binom{3}{2}(1-t)\, t^2 P_2 + \binom{3}{3} t^3 P_3 \,, \quad 0 \le t \le 1$$

   They look nice! most popular in CAD-CAM, computer graphics, Adobe Illustrator, Flash

   k-th degree $\vec{B}_k(t) = \sum_{i=0}^{k} \binom{k}{i} t^i (1-t)^{k-i} P_i$ using Bernstein polynomials

4. **Cubic Splines**: piecewise cubics passing through data points, joined smoothly at knots (cont's 1^{st} and 2^{nd} derivatives)

5. **Hermite interpolation**: when slopes are also available at $N+1$ points:

   Polynomial of degree $\le 2N+1$ matching values and slopes

   Most common: piecewise Hermite cubics ($N=1$) every 2 points.

   pchip of Matlab: shape preserving piecewise Hermite cubics, visually pleasing (slopes at knots chosen to not overshoot the values)

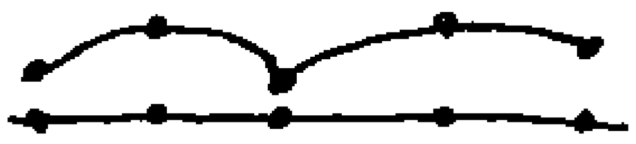
6. **Taylor polynomial**: matches value and derivatives at a single point! extreme case

7. **Trig. polynomials** or **rational functions** or exponentials or ...
   (Fourier expansions) (Padé approx.)

Piecewise Polynomial Interpolation

To avoid high degree polynomials, use different low degree every few nodes (knots)

piecewise linear: a line on $\{x_i, x_{i+1}\}$  what plotters do

piecewise quadratic: a parabola on $\{x_{i-1}, x_i, x_{i+1}\}$ 

piecewise cubic: a cubic on $\{x_i, x_{i+1}, x_{i+2}, x_{i+3}\}$

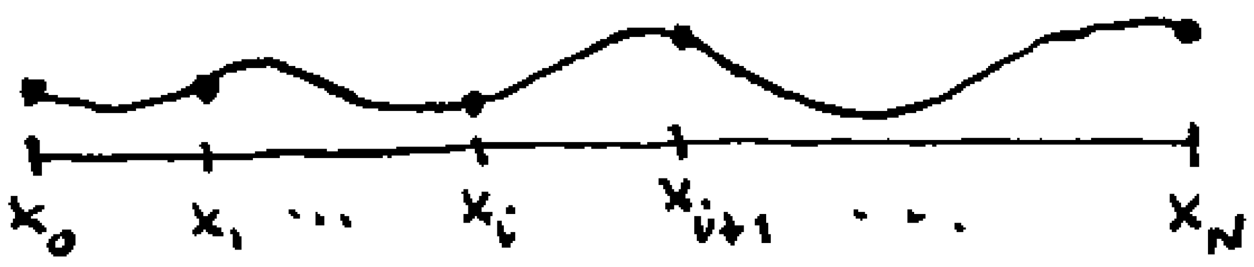
piecewise Hermite cubic: a cubic on $\{x_i, x_{i+1}\}$ matching $f_i, f_{i+1}, f'_i, f'_{i+1}$

Advantages: low degree, so small condition #, and choosing small spacing (if possible) also small approximation error

Disadvantages: overall interpolant <u>not</u> smooth (corners at knots)
good for integration but terrible for derivatives!
More expensive to evaluate: must locate which subinterval to evaluate
kills vectorization!

Cubic Splines: piecewise cubics joined smoothly at the knots

spline has <u>continuous</u> 1^{st} and 2^{nd} derivatives, but only values of f are matched, <u>not</u> its derivatives

Construction: 

$S(x) = S_i(x)$ on $[x_i, x_{i+1}]$ $i = 1, \dots, N$ (cubic)

$$S_i(x) = A_{0i} + A_{1i}(x - x_i) + A_{2i}(x - x_i)^2 + A_{3i}(x - x_i)^3$$

the $4N$ coefficients are determined from the conditions:

$$\begin{cases} S(x_i) = f(x_i), & i = 0, 1, 2, \dots, N-1, N \quad : \quad 2(N-1)+2 = 2N \text{ equs} \\ S'_i(x_i) = S'_{i+1}(x_i), & i = 1, 2, \dots, N-1 \quad : \quad N-1 \text{ equs} \\ S''_i(x_i) = S''_{i+1}(x_i), & i = 1, 2, \dots, N-1 \quad : \quad \underline{N-1} \text{ equs} \\ & \qquad\qquad\qquad\qquad\qquad 4N-2 \text{ equs} \\ \text{2 end-point constraints, of various types:} & \end{cases}$$

natural $S''(x_0) = 0$, $S''(x_N) = 0$

quadratic $S_1(x)$ and $S_{N-1}(x)$ are quadratics (2 fewer coeffs)

not-a-knot $S_1'''(x_1) = S_2'''(x_1)$, $S_{N-2}'''(x_N) = S_{N-1}'''(x_N)$

periodic $S_1'(x_0) = S_{N-1}'(x_N)$, $S_1''(x_0) = S_{N-1}''(x_N)$

This $4N \times 4N$ linear system for coeffs A_{ki} is tridiagonal, solved efficiently by the Tridiagonal Algorithm

But evaluation at a given x is expensive: must locate which interval x lies in (binary search)
then evaluate a cubic (in nested form, of course)
Kills vectorization!