Lecture 08: Interpolation cont'd

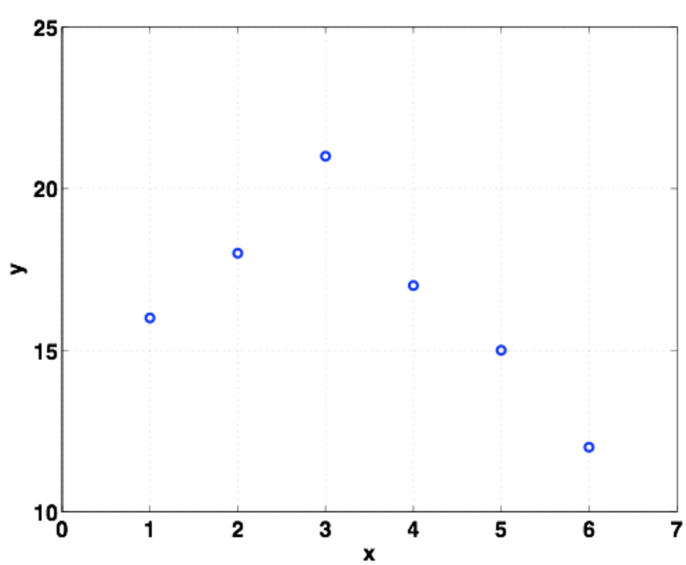
Outline

1) The Fundamental Interpolation problem
2) Beyond Full Polynomial Interpolation
3) Piecewise Polynomial Interpolation
 Basic definitions:
 Themes and Variations:
 C0: Piecewise overlapping Polynomial Interpolation
  (linear interpolation, overlapping cubic interpolation)
 C1: Piecewise Cubic Hermite Polynomials (PCHiP)
 C2: Cubic Splines
4) Choosing a good interpolant

The pure interpolation problem

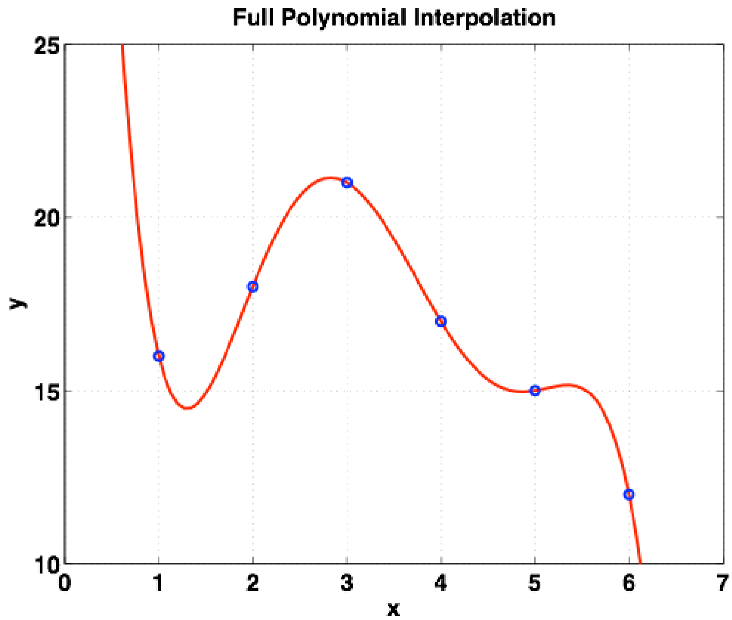
Given N points (x_i, y_i), how to choose a good interpolant to "connect the dots" with no other information about the underlying function/data.

Example: x = [1:6]', y=[16 18 21 17 15 12]'



Review full Polynomial Interpolation:

Given: N+1 points, fit all of them exactly with a single Nth order polynomial



Piecewise Polynomial Interpolation

Idea: Given N points, use lower order polynomial interpolation to fit the function in pieces.

Issue: You get to choose the order and the Continuity

Flavors:

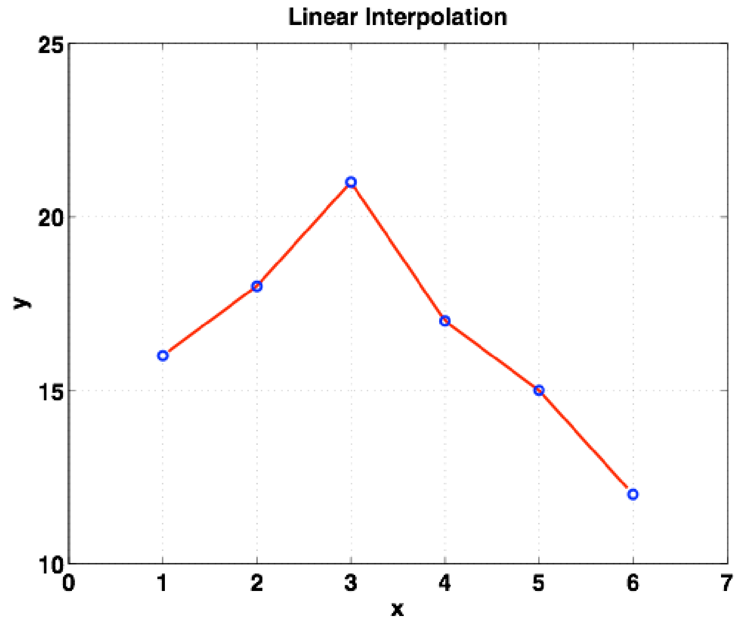
C0: (Continuous)
- Piecewise Linear interpolation: yi = interp1(x,y,xi,'linear')
- Piecewise overlapping Cubic Interpolation

C1: (Continuous first derivative)
- Piecewise Cubic Hermite Polynomials (PCHiP): yi = interp1(x,y,xi,'cubic')

C2: Continuous 2nd Derivative
- Cubic Splines: yi = interp1(x,y,xi,'spline')

Piecewise Linear Interpolation
Connect the dots

Just use linear line segments to interpolate between points

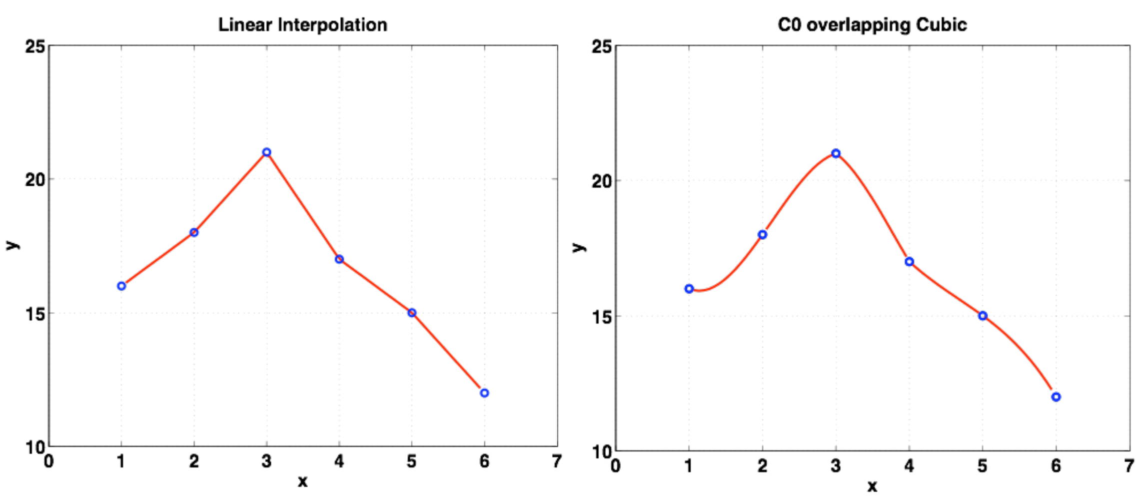


Piecewise Linear Interpolation
Connect the dots

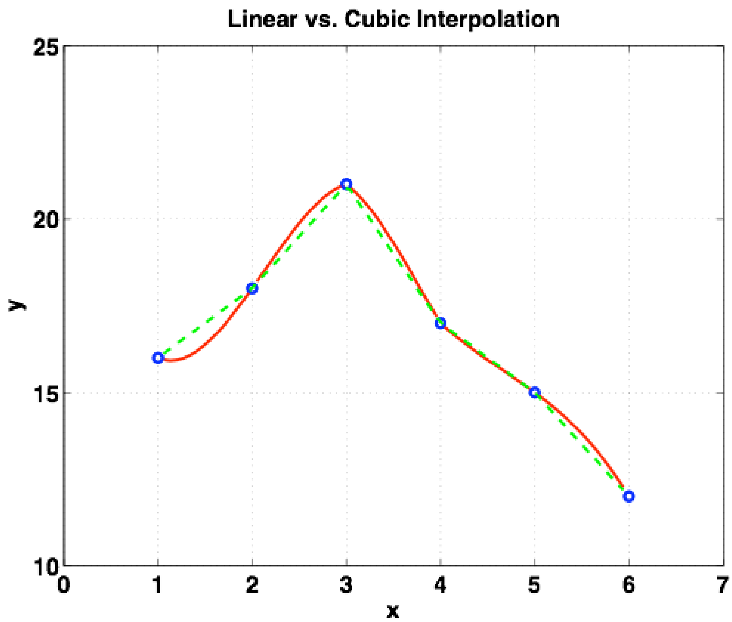
Given segment between point (x_k, y_k) and (x_{k+1}, y_{k+1})

Define:

Piecewise Overlapping Polynomial Interpolation



Piecewise Overlapping C0 Polynomial Interpolation



Issue: first derivative is discontinuous at the node boundaries

C1 Piecewise Cubic Interpolation
Hermite Polynomials

Given segment between point (x_k, y_k) and (x_{k+1}, y_{k+1})

1-D Polynomial with 4 degrees of freedom

C1 Piecewise Cubic Interpolation
Hermite Polynomials

Plug and Chug and get:

$P(s) = (1-s)^2(1+2s)y_k + s^2(3-2s)y_{k+1} + s(1-s)^2 d^*_k - s^2(1-s)d^*_{k+1}$

$P'(s) = 6s(s-1)\ y_k \quad + 6s(1-s)y_{k+1} \quad + (s-1)(3s-1)d^*_k + s(3s-2)d^*_{k+1}$

$P''(s) = 6(1-2s)(y_{k+1} - y_k) \quad + (6s-4)d^*k \quad + (6s-2)d^*_{k+1}$

(ugly but easy to check...)

can also check for continuity of 1st derivative i.e. $P'_{k-1}(1)=P'_k(0)=d_k$

C1 Piecewise Cubic Interpolation

Hermite Polynomials

Question: y_k are given by data...but how to choose d_k? (Infinite number of choices)

A recipe: PCHiP - a "visually pleasing", "shape preserving" Piecewise Cubic Hermite Polynomial:

Uses the data to define d_k and preserve monotonicity if possible (avoid overshoots) (Fritsch and Carlson; Kahaner, Moler and Nash)

The rules:

define δ_k as slope of k^{th} segment (between k and k+1)

if $\mathrm{sgn}(\delta_k) \neq \mathrm{sgn}(\delta_{k-1})$ $d_k=0$

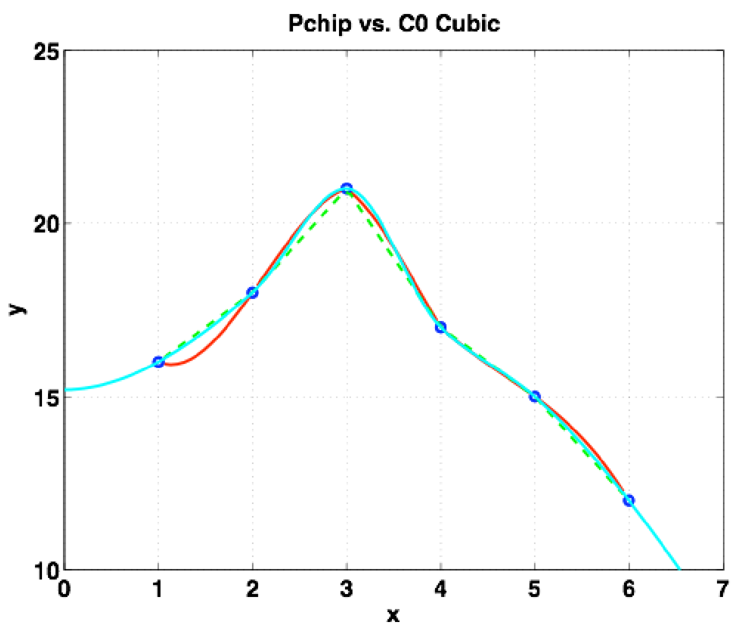
if $\delta_k=0$ or $\delta_{k-1}=0$ $d_k=0$

else use weighted harmonic mean

let $w_1 = 2h_k+h_{k+1}$, $w_2 = h_k+2h_{k-1}$ (equal spacing $w_1=w_2=3h$)

$1/dk= 1/(w_1+w_2)\,[\,w_1/\delta_{k-1} + w_2/\delta_{k-1}\,]$

C1 "PChip" Polynomial Interpolation



Matlab implementation

yi = interp1(x,y,xi,'pchip')

C2 Piecewise Cubic Interpolation
Cubic Splines...

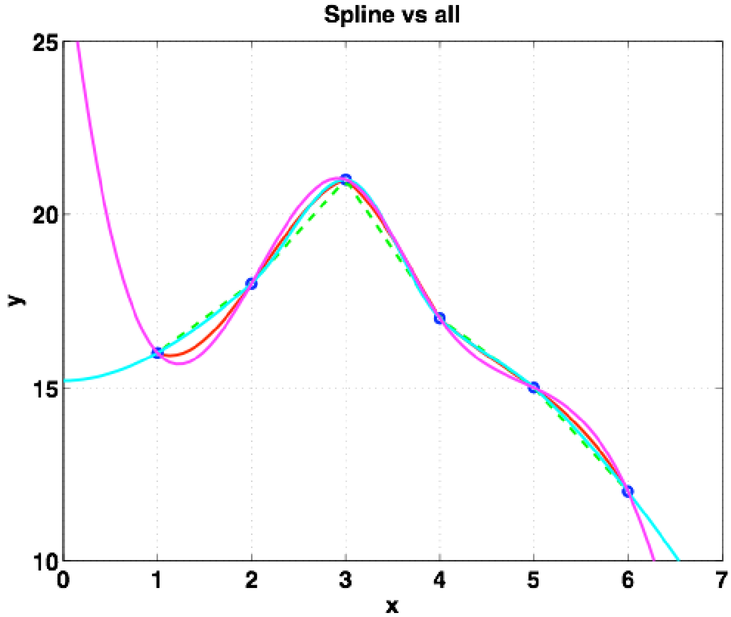
Same idea (and cubic polynomial) as PChip but now enforce continuity of second derivatives

1) need to set $P''(x)_{k-1} = P''(x)_k$

But: $P''(s) = 6(1-2s)(y_{k+1} - y_k) + (6s - 4)d^*k + (6s - 2)d^*_{k+1}$

or

C2 Cubic Spline Interpolation



Matlab implementation

yi = interp1(x,y,xi,'spline')

More fun: Moler's interpgui, pchiptx, splinetx

Heath's spline demos

Philosophical Question: Which interpolant is "best"?