Interpolation

Piecewise Polynomial Interpolation

Sachin Shanbhag

Department of Scientific Computing
Florida State University,
Tallahassee, FL 32306.

References

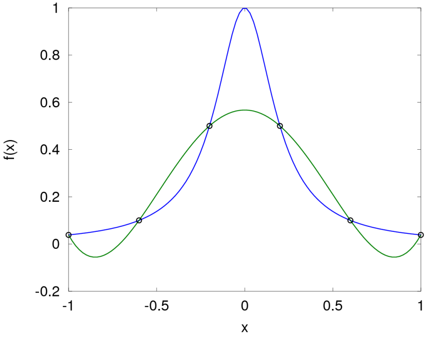
- S. Chapra and R. Canale, Numerical Methods for Engineers
- A. Greenbaum and T. Chartier, Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms
- Carnahan and Wilkes, Applied Numerical Methods, University of Michigan, Class Notes, 1996.
- Burden and Faires, Numerical Analysis, 1993
- Pal, Numerical Analysis for Scientists and Engineers, 2007.
- M. Heath, Scientific Computing: An Introductory Survey
- wikipedia.org

Runge's Phenomenon

Consider "Runge's function" over the domain $x \in (-1, 1)$,

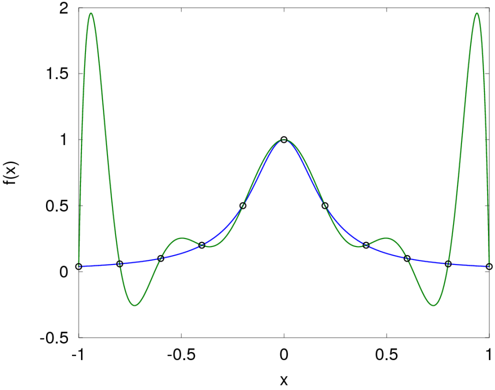
$$f(x) = \frac{1}{1 + 25x^2}$$

- Let us divide the domain into n intervals ($n + 1$ points for interpolation)
- Compute the interpolating polynomial $p_n(x)$, for $n = 5$.



Runge's Phenomenon

- For n=10



- As the number of points increase, high order polynomials cause oscillations.

Piecewise Polynomials

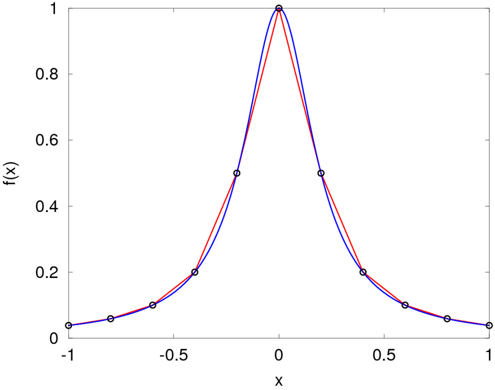
- Divide domain into subintervals
- Fit a lower-order polynomial (3rd or 4th) through each interval
- Intervals may share end-points through which information about continuity and smoothness is communicated

$$\begin{array}{cccccccccc}
x_1 & x_2 & x_3 & x_4 & & & & & & \\
 & & & x_4 & x_5 & x_6 & x_7 & & & \\
 & & & & & & x_7 & x_8 & x_9 & x_{10}
\end{array}$$

- Simplest local interpolation is linear

Piecewise Linear

- Avoids Runge's phenomenon

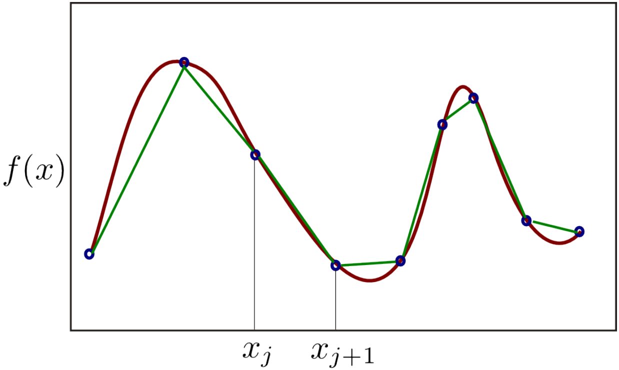


- Each interval contains the domain between two points
- The piecewise curve is continuous, but not differentiable

Piecewise Linear

- In each interval $I_j = [x_j, x_{j+1}]$, a linear polynomial passing through (x_j, f_j) and (x_{j+1}, f_{j+1}) is constructed.

$$\begin{aligned} p_1^j(x) &= f(x_j) + f[x_j, x_{j+1}](x - x_j) \\ &= f_j + \frac{f_{j+1} - f_j}{x_{j+1} - x_j}(x - x_j) \end{aligned}$$



Postmortem

- For n intervals ($n+1$ points: x_0, ..., x_n) we define $f(x)$ as a collection of n piecewise order 1 (linear) polynomials $p_1^0(x), ..., p_1^{n-1}(x)$.
- Since $p_1^j(x) = a_0 + a_1 x$, we need to determine a_0 and a_1 (2 unknowns)
- Since we assert,

$$p_1^j(x_j) = f_j$$
$$p_1^j(x_{j+1}) = f_{j+1}$$

  we have 2 equations for each I_j.
- Number of unknowns = Number of equations (2 per interval, or $2n$ in all)

Piecewise Cubic

- Suppose, you prefer something smoother than piecewise linear (you don't like the sharp edges)
- Piecewise cubic is a popular choice

$$p_3^j = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

- 4 unknowns per interval
- If we assert,

$$p_3^j(x_j) = f_j$$
$$p_3^j(x_{j+1}) = f_{j+1}$$

  we have 2 equations for each I_j. We need two more.
- Several different possibilities

Piecewise Cubic Hermite

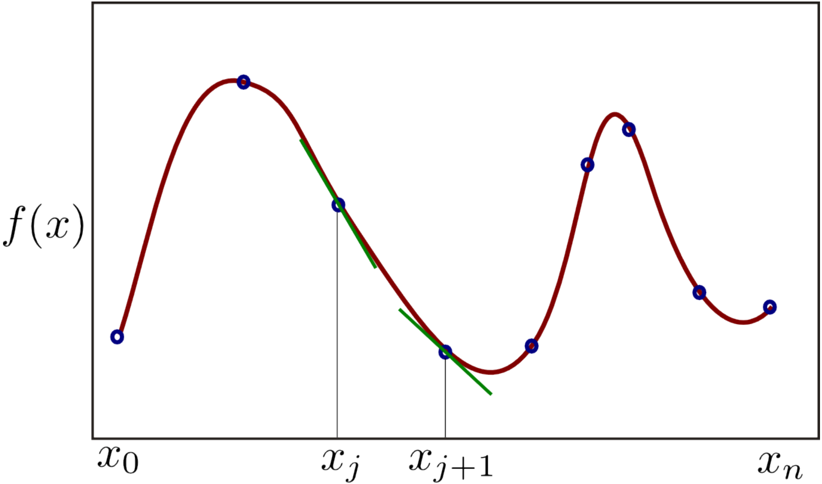
- If we know the derivatives $f'(x_j) = f'_j$ at all the interpolation points then we can additionally assert:

$$\frac{dp_3^j(x_j)}{dx} = f'_j$$
$$\frac{dp_3^j(x_{j+1})}{dx} = f'_{j+1}$$

  which gives us the required 4 equations for each I_j.
- This is called piecewise cubic Hermite interpolation
- Now that we have figured out that we have a solvable problem, let us proceed to evaulate $p_3^j(x)$
- For notational simplicity, let me drop the subscript 3, and redefine $C_j(x) = p_3^j(x)$.

Piecewise Cubic Hermite

- Consider the following picture with



Piecewise Cubic Hermite

In principle, we can write down

$$C_j(x) = a + bx + cx^2 + dx^3,$$

and impose the four conditions:

$$\begin{aligned} C_j(x_j) &= f_j \\ C_j(x_{j+1}) &= f_{j+1} \\ C_j'(x_j) &= f_j' \\ C_j'(x_{j+1}) &= f_{j+1}', \end{aligned}$$

to solve for a, b, c, and d.

A less messy method is appended at the end of the lecture notes.

Piecewise Cubic Hermite

- The complete expression is:

$$C_j(x) = -\frac{f'_j}{2h_j}\left((x-x_{j+1})^2 - h_j^2\right) + \frac{f'_{j+1}}{2h_j}(x-x_j)^2 + \alpha(x-x_j)^2\left(\frac{x-x_j}{3} - \frac{h_j}{2}\right) + f_j$$

with $h_j = x_{j+1} - x_j$ the size of the interval, and

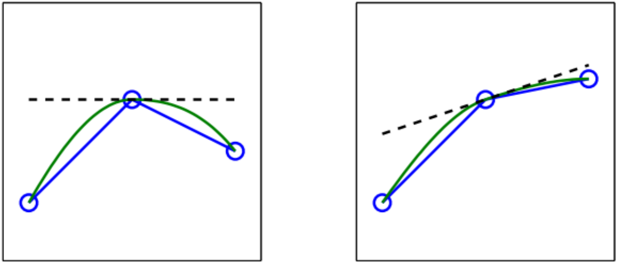
$$\alpha = \frac{3}{h_j^2}\left(f'_j + f'_{j+1}\right) + \frac{6}{h_j^3}\left(f_j - f_{j+1}\right).$$

- Note that I only need information at the end points of interval I_j to determine $C_j(x)$

Matlab's `pchip`

- Matlab's intrinsic `pchip` routine does not require derivatives, f_i', to be specified.
- They instead computed from the $\{x_i, f_i\}$ with the idea of mimicking the shape of the data
- Hence, a better label is perhaps shape-preserving piecewise cubic Hermite interpolating polynomial
- An intuitive way of understanding what it does is to consider the underlying piecewise linear interpolation
- If the slopes over the interval I_j and I_{j+1}, which share the point x_{j+1} have different signs then $f_{j+1}^{'}$ is set to zero.
- When the slopes are of the same sign, then $f_{j+1}^{'}$ is set as the weighted harmonic mean.

Matlab's pchip



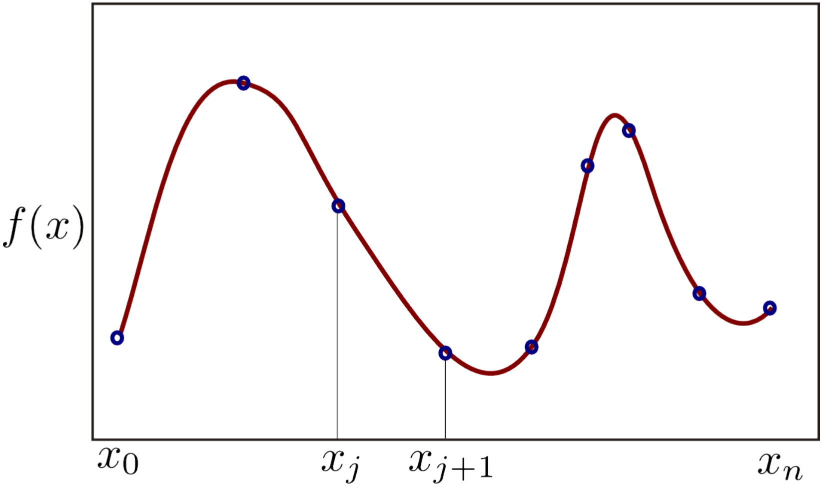
- As a result the interpolant never "overshoots" the data.
- The end points are treated in a slightly special way.
- Check out an interesting comparison between pchip and splines in the following blog post `http://blogs.mathworks.com/cleve/2012/07/16/splines-and-pchips/`

Cubic Splines

- Local Piecewise Cubic Hermite
  - builds local interpolating function
  - piecewise cubic
  - C^1 smoothness, across adjacent intervels
  - first derivatives are specified or inferred
- Cubic Splines
  - builds global interpolating function
  - piecewise cubic
  - globally C^2
  - derivatives are computed, not specified (may not match)

Cubic Splines

- Consider the following picture with



Cubic Splines

- For each of the n intervals define $0 \leq j \leq n-1$

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

- C^0 continuity
  - Match the values: $n+1$ conditions

$$\begin{aligned} S_j(x_j) &= f_j, \qquad 0 \leq j \leq n-1 \\ S_{n-1}(x_n) &= f_n \end{aligned}$$

  - Adjacent cubics match values at shared points: $n-1$ conditions

$$S_{j+1}(x_{j+1}) = S_j(x_{j+1}), \qquad 0 \leq j \leq n-2$$

- Total number of conditions from C^0 continuity is $2n$

Cubic Splines

- C^1 continuity
  - Adjacent cubics match derivatives at shared points: $n-1$ conditions

$$S'_{j+1}(x_{j+1}) = S'_j(x_{j+1}) \qquad 0 \leq j \leq n-2$$

- C^2 continuity
  - Adjacent cubics match derivatives at shared points: $n-1$ conditions

$$S''_{j+1}(x_{j+1}) = S''_j(x_{j+1}) \qquad 0 \leq j \leq n-2$$

Cubic Splines

- Number of unknowns, four for each of S_0 to $S_{n-1} = 4n$
- Number of equations

$$2n + 2(n-1) = 4n - 2$$

- That is we have two unknowns more than we have equations
- Need two more boundary conditions at extremities
- a popular choice: “natural” boundary conditions

$$S_1''(x_1) = S_{n-1}''(x_n) = 0$$

Cubic Splines: Coefficients

- Interpolant

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

- Solution

$$a_j = f_j$$

- Triadiagonal system of equations in c_j

$$\begin{aligned} h_{j-1}c_{j-1} \quad + \quad & 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \\ & \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}) \end{aligned}$$

Cubic Splines: Coefficients

- Can then get the d_j

$$d_j = \frac{c_{j+1} - c_j}{3h_j}$$

- and the b_j

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1})$$

Matlab Example

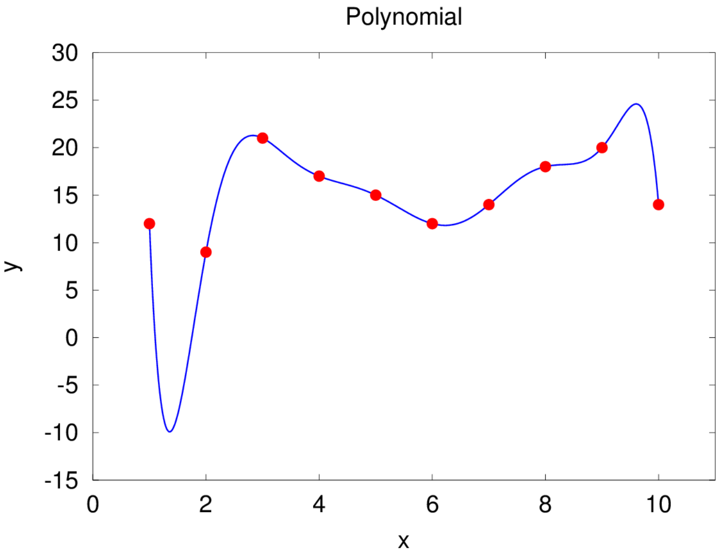
Consider an example with the follwing data

x	1	2	3	4	5	6	7	8	9	10
y	12	9	21	17	15	12	14	18	20	14

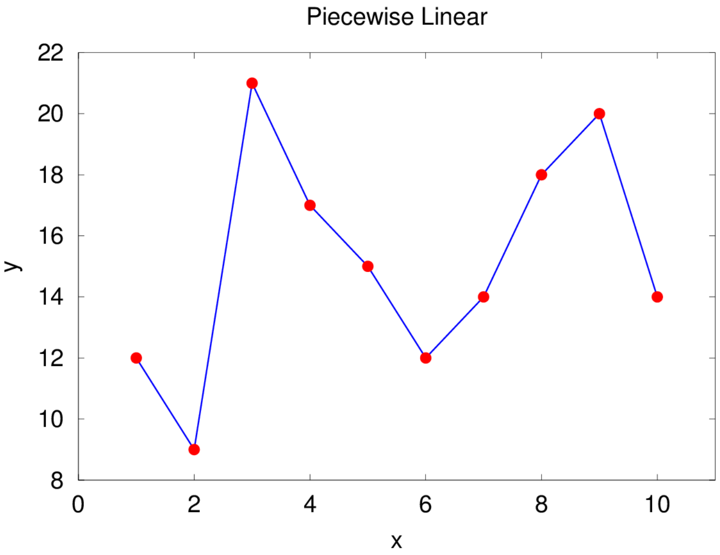
Let us look at the following interpolants to this data

- polynomial
- piecewise linear
- pchip (matlab)
- spline

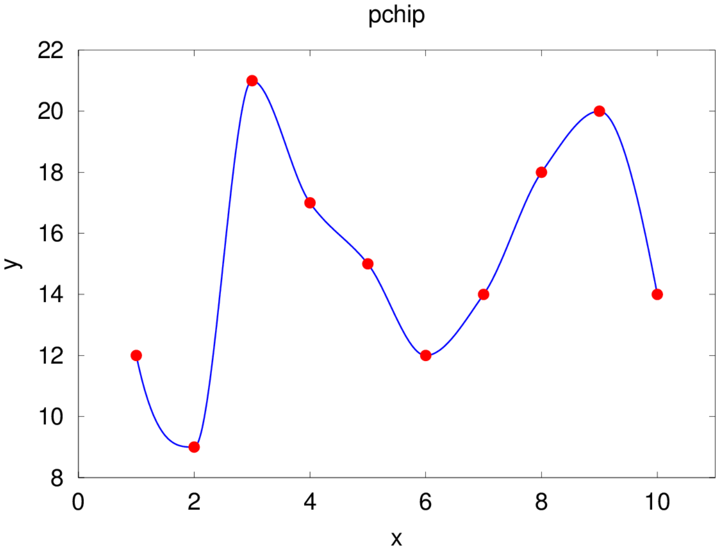
Matlab Example



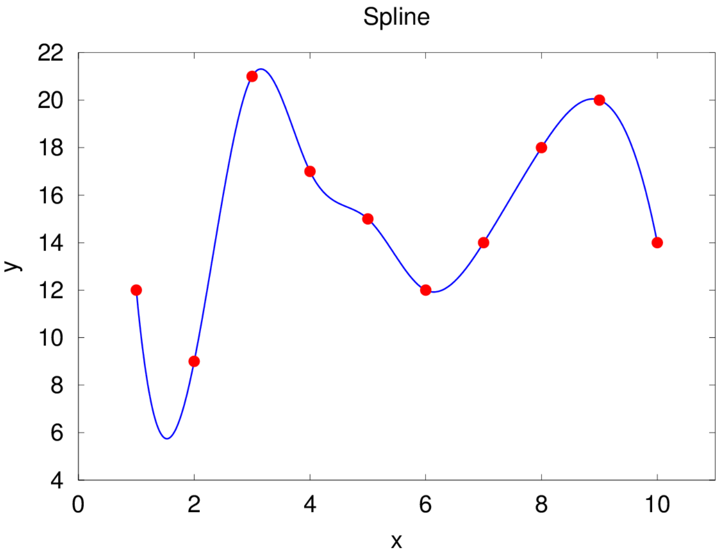
Matlab Example



Matlab Example



Matlab Example



Cubic Hermite versus Cubic Splines

- The local error over the interval $I_j = [x_j, x_{j+1}]$ can be shown to be
  - Cubic Hermite
$$\frac{1}{384}||f^{(4)}(\xi)||_\infty (x_{j+1} - x_j)^4, \qquad \xi \in I_j$$
  - Splines
$$\frac{5}{384}||f^{(4)}(\xi)||_\infty (x_{j+1} - x_j)^4, \qquad \xi \in I_j$$
- Cubic Hermite are "very local". Changing a single (x_i, f_i) causes change only in the two adjacent sub-intervals.
- Cubic Splines are "global". Changing a single (x_i, f_i) changes the tridiagonal system of equations. All the piecewise curves have to be recomputed.

Appendix: Piecewise Cubic Hermite Derivation

- Since $C_j(x)$ is order 3, its derivative $C_j'(x)$ is a quadratic (order 2) polynomial.
- Let us write $C_j'(x)$ as

$$C_j'(x) = f_j' \frac{x - x_{j+1}}{x_j - x_{j+1}} + f_{j+1}' \frac{x - x_j}{x_{j+1} - x_j} + \alpha(x - x_j)(x - x_{j+1})$$

- Note that $C_j'(x)$ passes through (x_j, f_j'), and (x_{j+1}, f_{j+1}')
- The additional parameter α will allow us to match function values. Note that this last piece is zero at both the end points of I_j
- Let us integrate the equation above

$$C_j(x) = \int C_j'(x) dx + \text{constant}$$

Appendix: Piecewise Cubic Hermite Derivation

- If $h_j = x_{j+1} - x_j$, this yields

$$C_j(x) = -\frac{f'_j}{h_j}\int_{x_j}^{x}(t - x_{j+1})dt + \frac{f'_{j+1}}{h_j}\int_{x_j}^{x}(t - x_j)dt + \alpha\int_{x_j}^{x}(t - x_j)(t - x_{j+1})dt + \text{constant}$$

- Requiring $C_j(x_j) = f_j \implies$ constant $= f_j$
- If we perform the integration, and assert the final condition $C_j(x_{j+1}) = f_{j+1}$ we can determine α

$$\alpha = \frac{3}{h_j^2}\left(f'_j + f'_{j+1}\right) + \frac{6}{h_j^3}\left(f_j - f_{j+1}\right)$$