Computational Methods
CMSC/AMSC/MAPL 460

Piecewise Polynomial Interpolation

Ramani Duraiswami,
Dept. of Computer Science

Interpolation: the story so far

- Given a function at *N* points, find its value at other point(s)
- Last class: polynomial interpolation
  - Polynomials are guaranteed to approximate any given function in an interval as accurately as we want
- Different polynomial bases
  - Monomial or Power basis
  - Newton and Lagrange basis
- For a given set of points and function values
  - interpolating polynomial is unique
- Interpolation problem requires solution of a linear system
  - System is dense for Monomial/Power basis
  - Newton and Lagrange forms allow the direct solution of the polynomial interpolation form
  - Newton form particularly convenient to add new values
- Error for interpolation with n points is related to the value of the $(n+1)^{th}$ derivative of the underlying function

Polyinterp

- Lagrange interpolation code
  - x,y are points and function values
  - u are points where vector

```
function v = polyinterp(x,y,u)
n = length(x);
v = zeros(size(u));
for k = 1:n
%Lagrange function k at u
    w = ones(size(u));
    for j = [1:k-1 k+1:n]
        w = (u-x(j))./(x(k)-x(j)).*w;
    end
    v = v + w*y(k);
end
```

- Cost: 2 nested loops, so the cost is n^2.
- `k = 5, n =  9`
- `j= [1:k-1 k+1:n]`
- `j =  1      2      3      4      6      7      8      9`

$$p_n(x) = \sum_{i=1}^{n} L_i(x) f(x_i)$$

$$L_i(x) = \prod_{k=1, k\neq i}^{n} \frac{(x - x_k)}{(x_i - x_k)}$$

$$L_i(x_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

Examples of polynomial interpolation

- Go to MATLAB demo
  - Vandermonde
  - Polynomial interpolation for small set
  - For larger set
- See that even for six points we have a problem
  - In between the data points, (especially in first and last subintervals), function shows excessive variation.
  - overshoots changes in the data values.
  - As a result, full-degree polynomial interpolation is hardly ever used for data and curve fitting.
- However we saw polynomial interpolation works well when degree is low

Piecewise linear interpolation

- Simple idea
  - Connect straight lines between data points
  - Any intermediate value read off from straight line
- The *local variable*, s, is
- $s = x - x_k$
- The *first divided difference* is
- $\delta_k = (y_{k+1} - y_k)/(x_{k+1} - x_k)$
- With these quantities in hand, the interpolant is
- $L(x) = y_k + (x - x_k)\,(y_{k+1} - y_k)/(x_{k+1} - x_k)$
- $= y_k + s\delta_k$
- Linear function that passes through (x_k, y_k) and (x_{k+1}, y_{k+1})

Piecewise linear interpolation

- Same format as all other interpolants
- Function diff finds difference of elements in a vector
- Find appropriate sub-interval
- Evaluate
- Jargon: x is called a "knot" for the linear spline interpolant

```
function v = piecelin(x,y,u)
%PIECELIN Piecewise linear interpolation.
% v = piecelin(x,y,u) finds piecewise linear L(x)
% with L(x(j)) = y(j) and returns v(k) = L(u(k)).
% First divided difference
delta = diff(y)./diff(x);
% Find subinterval indices k so that x(k) <= u <
x(k+1)
n = length(x);
k = ones(size(u));
for j = 2:n-1
k(x(j) <= u) = j;
end
% Evaluate interpolant
s = u - x(k);
v = y(k) + s.*delta(k);
```

How good is piecewise linear interpolation?

Recall from Polynomial interpolation: If $f \in \mathcal{C}^n[I]$, then

$$f(x) - p_{n-1}(x) = \frac{(x - x_1)\dots(x - x_n)f^{(n)}(\xi)}{n!}$$

for some point ξ in the interval containing I and x.

We need to apply this to a polynomial of degree $n - 1 = 1$, so we obtain

$$f(x) - p_1(x) = \frac{(x - x_i)(x - x_{i+1})f''(\xi)}{2}$$

- So we can reduce error by choosing small intervals where 2nd derivative is higher
  - If we can choose where to sample data
  - Do more where the “action” is more

Piecewise Cubic interpolation

- While we expect function not to vary, we expect it to also be smooth
- So we could consider piecewise interpolants of higher degree
- How many pieces of information do we need to fit a cubic between two points?
  - $y=a+bx+cx^2+dx^3$
  - 4 coefficients
  - Need 4 pieces of information
  - 2 values at end points
  - Need 2 more pieces of information
  - Derivatives?

Cubic interpolation

- ordinary cubic polynomials: 3 continuous nonzero derivatives.
- **cubic splines**: 2 continuous nonzero derivatives.
- **Hermite cubics**: 1 continuous nonzero derivative.

- However for Hermite, the derivative needs to be specified
- Cubic splines, the derivative is not specified but enforced

Hermite Cubic interpolation

- Define the following interpolant
- Called Hermite or “osculatory” interpolant
- Will work if we know function and derivative values
- Often only function values are known

$$\delta_k = \frac{y_{k+1} - y_k}{h_k}$$

Let d_k denote the slope of the interpolant at x_k.

$$d_k = P'(x_k)$$

terms of local variables $s = x - x_k$ and $h = h_k$

$$P(x) = \frac{3hs^2 - 2s^3}{h^3} y_{k+1} + \frac{h^3 - 3hs^2 + 2s^3}{h^3} y_k + \frac{s^2(s-h)}{h^2} d_{k+1} + \frac{s(s-h)^2}{h^2} d_k$$

$$P(x_k) = y_k, \ P(x_{k+1}) = y_{k+1}$$
$$P'(x_k) = d_k, \ P'(x_{k+1}) = d_{k+1}$$

How do we specify 2 additional conditions?

- We don’t know derivatives
- But we can require that they be continuous!
- Requiring first derivative be continuous provides one relation at a “knot”
- Requiring second derivative be continuous provides one relation at a knot

Cubic splines

Notation:

- $h_{i+1} = x_{i+1} - x_i$, $i = 1, \ldots, n-1$
- $k_{i+1} = f_{i+1} - f_i$, $i = 1, \ldots, n-1$
- $I_{i+1} = [x_i, x_{i+1}]$, $i = 1, \ldots, n-1$

We will set $s(x)$ equal to $s_{i+1}(x)$ on interval I_{i+1}, where

$$s_{i+1}(x) = m_i \frac{(x_{i+1} - x)^3}{6h_{i+1}} + m_{i+1} \frac{(x - x_i)^3}{6h_{i+1}} + a_i(x - x_i) + b_i$$

Imposing the continuity conditions

$$s_{i+1}(x) = m_i \frac{(x_{i+1} - x)^3}{6h_{i+1}} + m_{i+1} \frac{(x - x_i)^3}{6h_{i+1}} + a_i(x - x_i) + b_i$$

1. For $i = 1, \ldots, n-1$,

$$s_{i+1}(x_i) = f_i = m_i \frac{h_{i+1}^3}{6h_{i+1}} + m_{i+1}0 + a_i 0 + b_i \,.$$

Therefore,

$$b_i = f_i - m_i \frac{h_{i+1}^2}{6} \,.$$

$$s_{i+1}(x) = m_i \frac{(x_{i+1} - x)^3}{6h_{i+1}} + m_{i+1} \frac{(x - x_i)^3}{6h_{i+1}} + a_i(x - x_i) + b_i$$

Using function continuity

2. For $i = 1, \ldots, n-1$,

$$s_{i+1}(x_{i+1}) = f_{i+1} = m_i 0 + m_{i+1} \frac{h_{i+1}^3}{6h_{i+1}} + a_i h_{i+1} + b_i \,.$$

Therefore,

$$a_i = \frac{f_{i+1} - b_i - m_{i+1} \frac{h_{i+1}^2}{6}}{h_{i+1}} \,,$$

so

$$a_i = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{h_{i+1}}{6}(m_{i+1} - m_i)$$

So we have formulas for all of the as and bs in terms of the ms, and we have ensured that s is continuous.

First Derivative continuity

$$s_{i+1}(x) = m_i \frac{(x_{i+1} - x)^3}{6h_{i+1}} + m_{i+1} \frac{(x - x_i)^3}{6h_{i+1}} + a_i(x - x_i) + b_i$$

3. For $i = 1, \ldots, n-1$,

$$s'_{i+1}(x) = -\frac{m_i}{2h_{i+1}}(x_{i+1} - x)^2 + \frac{m_{i+1}}{2h_{i+1}}(x - x_i)^2 + a_i \,.$$

Therefore, $s'_{i+1}(x_i) = s'_i(x_i)$ if

$$-\frac{m_i}{2h_{i+1}} h_{i+1}^2 + a_i = \frac{m_i}{2h_i} h_i^2 + a_{i-1} \,, i = 2, \ldots, n-1.$$

Since $a_i = \frac{k_{i+1}}{h_{i+1}} - \frac{h_{i+1}}{6}(m_{i+1} - m_i)$, we have

$$-\frac{m_i}{2} h_{i+1} + \frac{k_{i+1}}{h_{i+1}} - \frac{h_{i+1}}{6}(m_{i+1} - m_i) = \frac{m_i}{2} h_i + \frac{k_i}{h_i} - \frac{h_i}{6}(m_i - m_{i-1}) \,.$$

Second derivative continuity

$$s'_{i+1}(x) = -\frac{m_i}{2h_{i+1}}(x_{i+1} - x)^2 + \frac{m_{i+1}}{2h_{i+1}}(x - x_i)^2 + a_i \,.$$

4. For $i = 1, \ldots, n-1$,

$$s''_{i+1}(x) = +\frac{m_i}{h_{i+1}}(x_{i+1} - x) + \frac{m_{i+1}}{h_{i+1}}(x - x_i) \,.$$

Therefore, $s''_{i+1}(x_i) = m_i = s''_i(x_i)$ **for** $i = 2, \ldots, n-1$, so continuity of this derivative is built into the representation!

Note that

$$\begin{aligned} s''(x_1) &= s_1(x_1) = m_1 \\ s''(x_n) &= s_n(x_n) = m_n \end{aligned}$$

Solving for m

Our function s is an **interpolating cubic spline** if, for $i = 2, \ldots, n-1$,

$$-\frac{m_i}{2}h_{i+1} + \frac{k_{i+1}}{h_{i+1}} - \frac{h_{i+1}}{6}(m_{i+1} - m_i) = \frac{m_i}{2}h_i + \frac{k_i}{h_i} - \frac{h_i}{6}(m_i - m_{i-1}) .$$

and thus the parameters m_i, which are second derivatives at the knots, can be determined from the linear equations

$$\frac{h_i}{6}m_{i-1} + \frac{h_{i+1} + h_i}{3}m_i + \frac{h_{i+1}}{6}m_{i+1} = -\frac{k_i}{h_i} + \frac{k_{i+1}}{h_{i+1}} \equiv -\gamma_i + \gamma_{i+1} .$$

If we set $c_i = h_i/6$, then we can write the system as

$$\begin{bmatrix} c_2 & 2(c_2+c_3) & c_3 & & & & \\ & c_3 & 2(c_3+c_4) & c_4 & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & c_{n-1} & 2(c_{n-1}+c_n) & c_n \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \cdot \\ \cdot \\ \cdot \\ m_n \end{bmatrix} = \begin{bmatrix} -\gamma_2 + \gamma_3 \\ -\gamma_3 + \gamma_4 \\ \cdot \\ \cdot \\ \cdot \\ -\gamma_{n-1} + \gamma_n \end{bmatrix}$$

- n-2 equations in n unknowns

- Need to add two conditions
- Usually at end points

Common choices of end conditions

- The **natural** cubic spline interpolant: $s''(a) = s''(b) = 0$
- The **periodic** cubic spline interpolant: $s^{(k)}(a) = s^{(k)}(b)$, $k = 0, 1, 2$.
- The **complete** cubic spline interpolant: $s'(a)$ and $s'(b)$ given.
- The **not-a-knot** cubic spline interpolant: make the third derivative of s continuous at x_2 and x_{n-1} so that these points are not knots.

Solving a cubic spline system

- Assume natural splines

$$\begin{bmatrix} 2(c_2+c_3) & c_3 & & & \\ c_3 & 2(c_3+c_4) & c_4 & & \\ \cdot & \cdot & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & c_{n-1} & 2(c_{n-1}+c_n) \end{bmatrix} \begin{bmatrix} m_2 \\ m_3 \\ \cdot \\ \cdot \\ \cdot \\ m_{n-1} \end{bmatrix} = \begin{bmatrix} -\gamma_2+\gamma_3 \\ -\gamma_3+\gamma_4 \\ \cdot \\ \cdot \\ \cdot \\ -\gamma_{n-1}+\gamma_n \end{bmatrix}$$

- This is a tridiagonal system
- Can be solved in *O(n)* operations
- How?
  - Do LU and solve
  - With tridiagonal structure requires O(7n) operations