

Interpolation

Sanzheng Qiao

Department of Computing and Software
McMaster University

July, 2012

Outline

- 1 Introduction
- 2 Polynomial Interpolation
- 3 Piecewise Polynomial Interpolation
- 4 Natural Cubic Spline
- 5 Software Packages

Introduction

Problem setting: Given $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, $x_0 < x_1 < \dots < x_n$, for example, a set of measurements, construct a function f :

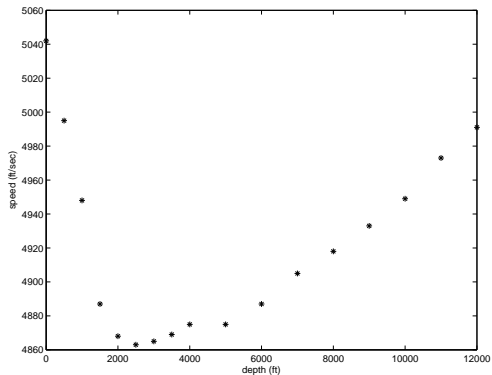
$$f(x_i) = y_i, \quad i = 0, 1, \dots, n$$

Desirable properties of f :

- smooth: analytic and $|f''(x)|$ not too large (the first and second derivatives are continuous).
- simple: polynomial of minimum degree, easy to evaluate.

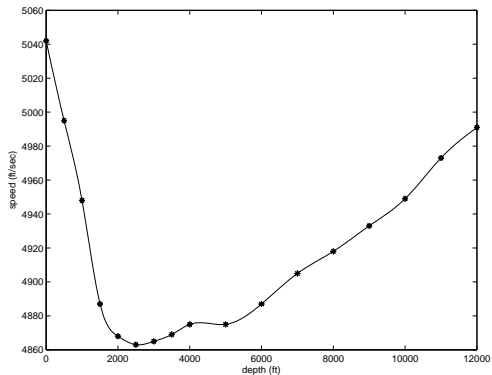
Example

Measurements of the speed of sound in ocean water



Example

Interpolation



Polynomial Interpolation

Advantages: easy to evaluate and differentiate

Weierstrass Approximation Theorem:

If f is any continuous function on the finite closed interval $[a,b]$, then for every $\epsilon > 0$ there exists a polynomial $p_n(x)$ of degree $n = n(\epsilon)$ such that

$$\max_{x \in [a,b]} |f(x) - p_n(x)| < \epsilon.$$

Impractical (degree is often too high)

A straightforward approach

A polynomial of degree n is determined by its $n + 1$ coefficients.

Given $(x_0, y_0), \dots, (x_n, y_n)$ to be interpolated, we construct the linear system (Vandermonde matrix):

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

solve for the coefficients of the polynomial

$$p_n(y)(x) = a_0 + a_1x + \cdots + a_nx^n$$

Vandermonde matrix

When x_0, \dots, x_n are distinct, the Vandermonde matrix is nonsingular. Thus the system has a unique solution (coefficients of the interpolating polynomial).

Example. Given two points $(28, 0.4695)$ and $(30, 0.5000)$, we have the system

$$\begin{bmatrix} 1 & 28 \\ 1 & 30 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 0.4695 \\ 0.5000 \end{bmatrix}$$

and the solution

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 30 & -28 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0.4695 \\ 0.5000 \end{bmatrix} = \begin{bmatrix} 0.04250 \\ 0.01525 \end{bmatrix}.$$

Vandermonde matrix

problem:

The coefficient (Vandermonde) matrix is often ill-conditioned

question

What is the condition number of the Vandermonde matrix constructed by $x_i = 2000 + i$, $i = 0, 1, \dots, 7$?

Answer: 1.87×10^{37}

Lagrange form (conceptually simple)

Basis of polynomials: $\{l_j(\mathbf{x})\}$ ($j = 0, 1, \dots, n$) of degree n such that

$$l_j(\mathbf{x}_i) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

construct

$$l_j(\mathbf{x}) = \prod_{i \neq j} \frac{\mathbf{x} - \mathbf{x}_i}{\mathbf{x}_j - \mathbf{x}_i}$$

Thus

$$p_n(\mathbf{y})(\mathbf{x}) = \sum_{j=0}^n l_j(\mathbf{x}) y_j$$

Example

Given three points: $(28, 0.4695)$, $(30, 0.5000)$, $(32, 0.5299)$,
construct a second degree interpolating polynomial in the
Lagrange form:

$$\begin{aligned} p_2(x) &= \frac{(x-30)(x-32)}{(28-30)(28-32)} 0.4695 \\ &\quad + \frac{(x-28)(x-32)}{(30-28)(30-32)} 0.5000 \\ &\quad + \frac{(x-28)(x-30)}{(32-28)(32-30)} 0.5299 \end{aligned}$$

$$p_2(31) = 0.5150 \approx \sin(31^\circ)$$

Hard to evaluate.

Horner's rule

Evaluating $a_0x^3 + a_1x^2 + a_2x + a_3$

Horner's form: $((a_0x + a_1)x + a_2)x + a_3$

```
v = a(0);  
for (i = 1:n)  
    v = v*x + a(i);  
end
```

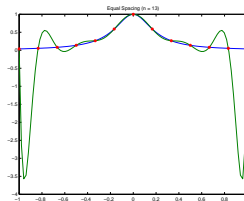
The optimal (most efficient and accurate) way of evaluating $a_0x^n + \dots + a_n$ (not in the Lagrange form).

Dangers of polynomial interpolation

An example. Runge's function (continuous derivatives of all order)

$$y(x) = \frac{1}{1 + 25x^2} \quad \text{on } [-1, 1]$$

equally spaces $x_0 = -1, x_1, \dots, x_n = 1$



It is often best not to use global polynomial interpolation.

Piecewise Polynomial Interpolation

Given the partition

$$\alpha = x_1 < x_2 < \cdots < x_n = \beta,$$

interpolate on each $[x_i, x_{i+1}]$ with a low degree polynomial.

Linear

$$L_i(z) = a_i + b_i(z - x_i), \quad z \in [x_i, x_{i+1}]$$
$$a_i = y_i, \quad b_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad 1 \leq i \leq n - 1$$

Algorithm. Piecewise linear interpolation

Given vectors x and y with interpolating points, this function returns the piecewise linear interpolation coefficients in the vectors a and b .

```
function [a,b] = pwL(x,y)
n = length(x);
a = y(1:n-1);
b = diff(y)./diff(x);
```

Evaluation

Given the piecewise linear interpolation $L(z)$ represented by the coefficient vectors a , b , how do we evaluate this function at $z \in [\alpha, \beta]$?

First, we locate $[x_i, x_{i+1}]$ such that $z \in [x_i, x_{i+1}]$. Then, we evaluate $L(z)$ using $L_i(z)$.

Search method: binary search, since x_i are sorted.

Observation: If $[x_i, x_{i+1}]$ is associated with the current z , then it is likely that this subinterval will be the one for the next value.

Algorithm. Locate

Idea: Use the previous subinterval as a guess. If not, do binary search.

Given the vector x of breakpoints and a scalar z between x_1 and x_n , this function locates i so that $x_i \leq z \leq x_{i+1}$. The optional g is a guess.

```
function i = Locate(x,z,g)
    if nargin==3      % try the guess
        if (x(g)<=z)&(z<=x(g+1))
            i = g;
            return    % quick return
        end
    end
end
```

Algorithm. Locate (cont.)

```
n = length(x);
if z==x(n)
    i = n-1;      % quick return
else             % binary search
    left = 1; right = n;
    while right > left+1
        mid = floor((left + right)/2);
        if z < x(mid)
            right = mid;
        else
            left = mid;
        end
    end
    i = left;
end
```

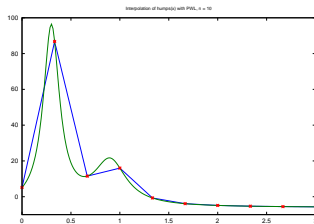
Algorithm. pwLEval

Given a piecewise linear interpolation coefficient vectors a and b from pwL and its breakpoints in x , this function returns the values of the interpolation evaluated at the points in z .

```
function v = pwLEval(a,b,x,z)
    m = length(z);
    v = zeros(m,1);
    g = 1;
    for j=1:m
        i = Locate(x,z(j),g);
        v(j) = a(i) + b(i)*(z(j) - x(i));
        g = i;
    end
```

Example

$$y = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04} - 6$$



Problem setting

Given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, find $s(x)$:

- in each subinterval $[x_i, x_{i+1}]$, $s(x)$ is cubic
- $s(x_i) = y_i, i = 1, \dots, n$
- $s'(x)$ and $s''(x)$ are continuous at x_2, x_3, \dots, x_{n-1}
- $s''(x_1) = s''(x_n) = 0$

The second derivative of $s(x)$ is zero at the end points means that $s(x)$ is linear at the end points.

A straightforward approach

Suppose $a_i + b_i x + c_i x^2 + d_i x^3$ on $[x_i, x_{i+1}]$, $i = 1, \dots, n - 1$.
 $4(n - 1)$ unknowns to be determined.

Interpolation:

$$a_i + b_i x_i + c_i x_i^2 + d_i x_i^3 = y_i, \quad i = 1, \dots, n - 1$$

$$a_i + b_i x_{i+1} + c_i x_{i+1}^2 + d_i x_{i+1}^3 = y_{i+1}, \quad i = 1, \dots, n - 1$$

Continuous first derivative (consider $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$):

$$b_{i-1} + 2c_{i-1}x_i + 3d_{i-1}x_i^2 = b_i + 2c_i x_i + 3d_i x_i^2, \quad i = 2, \dots, n - 1$$

Continuous second derivative:

$$2c_{i-1} + 6d_{i-1}x_i = 2c_i + 6d_i x_i, \quad i = 2, \dots, n - 1$$

Two end conditions:

$$2c_1 + 6d_1 x_1 = 0 \quad \text{and} \quad 2c_{n-1} + 6d_{n-1} x_n = 0$$

Total of $4(n - 1)$ equations, a dense system.

A clever approach: Constructing $s(x)$

In the subinterval $[x_i, x_{i+1}]$, let $h_i = x_{i+1} - x_i$ and introduce new variables:

$$w = (x - x_i)/h_i, \quad \bar{w} = 1 - w.$$

Note: $w(x_i) = 0$, $w(x_{i+1}) = 1$ and $\bar{w}(x_i) = 1$, $\bar{w}(x_{i+1}) = 0$, (linear Lagrange polynomials).

Thus $wy_{i+1} + \bar{w}y_i$ is the (linear) Lagrange interpolation on $[x_i, x_{i+1}]$.

Construct

$$s(x) = wy_{i+1} + \bar{w}y_i + h_i^2[(w^3 - w)\sigma_{i+1} + (\bar{w}^3 - \bar{w})\sigma_i]$$

where σ_i to be determined, so that the properties (the first and second derivatives are continuous) are satisfied.

Properties of $s(x)$

Using $w' = 1/h_i$ and $\bar{w}' = -1/h_i$, we can verify

- 1 $s(x_i) = y_i$, $s(x_{i+1}) = y_{i+1}$, independent of σ , that is, $s(x)$ interpolates (x_i, y_i) .
- 2 $s''(x) = 6w\sigma_{i+1} + 6\bar{w}\sigma_i$, linear Lagrange interpolation at the points $(x_i, 6\sigma_i)$ and $(x_{i+1}, 6\sigma_{i+1})$.

Clearly $s''(x_i) = 6\sigma_i$, which implies that $s''(x)$ is continuous.

Is $s'(x)$ continuous?

Properties of $s(x)$ (cont.)

It remains to determine σ_i so that $s'(x)$ is continuous.
Consider, on $[x_i, x_{i+1}]$,

$$s'(x) = \frac{y_{i+1} - y_i}{h_i} + h_i[(3w^2 - 1)\sigma_{i+1} - (3\bar{w}^2 - 1)\sigma_i]$$

Let $\Delta_i = (y_{i+1} - y_i)/h_i$.

On $[x_i, x_{i+1}]$, $w(x_i) = 0$ and $\bar{w}(x_i) = 1$,

$$s'_+(x_i) = \Delta_i + h_i(-\sigma_{i+1} - 2\sigma_i).$$

Properties of $s(x)$ (cont.)

On $[x_{i-1}, x_i]$,

$$s'(x) = \frac{y_i - y_{i-1}}{h_{i-1}} + h_{i-1}[(3w^2 - 1)\sigma_i - (3\bar{w}^2 - 1)\sigma_{i-1}]$$

and $w(x_i) = 1$, $\bar{w}(x_i) = 0$. Thus

$$s'_-(x_i) = \Delta_{i-1} + h_{i-1}(2\sigma_i + \sigma_{i-1}).$$

Making $s'(x)$ continuous

Setting

$$s'_+(x_i) = s'_-(x_i), \quad i = 2, 3, \dots, n-1,$$

we get $n-2$ equations:

$$h_{i-1}\sigma_{i-1} + 2(h_{i-1} + h_i)\sigma_i + h_i\sigma_{i+1} = \Delta_i - \Delta_{i-1}$$

for $i = 2, 3, \dots, n-1$.

Solve for $\sigma_2, \dots, \sigma_{n-1}$, recalling that $\sigma_1 = \sigma_n = 0$ (natural cubic spline).

Matrix form

diagonal: $[2(h_1 + h_2), \dots, 2(h_{n-2} + h_{n-1})]$

super/subdiagonal: $[h_2, \dots, h_{n-2}]$

unknowns: $[\sigma_2, \dots, \sigma_{n-1}]^T$

right-hand side: $[\Delta_2 - \Delta_1, \dots, \Delta_{n-1} - \Delta_{n-2}]^T$

The matrix is

- symmetric
- tridiagonal
- diagonally dominant ($|a_{i,i}| > \sum_{j \neq i} |a_{i,j}|$), when $x_1 < x_2 < \dots < x_n$

Can apply Gaussian elimination without pivoting, working on (two) three vectors with $O(n)$ operations.

Modeling a problem

Note. Had we taken the straightforward approach to determining the coefficients of the piecewise cubic polynomials, four coefficients for each of $n - 1$ cubic polynomials, we would have ended up with a large $(4(n - 1) \times 4(n - 1))$ and dense system requiring $O(n^3)$ operations.

Now we have an $O(n)$ method.

Evaluating $s(x)$

If $s(x)$ is evaluated many times, arrange $s(x)$ so that

$$s(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

and rearrange it in the Horner's form, for $x_i \leq x \leq x_{i+1}$ and calculate and store b_i, c_i, d_i (instead of σ_i)

$$b_i = \frac{y_{i+1} - y_i}{h_i} - h_i(\sigma_{i+1} + 2\sigma_i)$$

$$c_i = 3\sigma_i \quad d_i = \frac{\sigma_{i+1} - \sigma_i}{h_i}$$

for $i = 1, 2, \dots, n - 1$

Algorithm. Natural cubic spline

`ncspline`

Given a vector x with breakpoints and vector y with function values, this algorithm computes the coefficients b , c , d of natural spline interpolation.

- 1 Compute h_i and Δ_i ;
- 2 Form the tridiagonal matrix (two arrays) and the right hand side;
- 3 Solve for σ_i ;
- 4 Compute the coefficients b , c , and d .

Software packages

IMSL csint, csdec, csher, csval

MATLAB polyfit, spline, ppval

NAG e01aef, e01baf, e01bef, e02bbf, e01bff

Octave interp1

Summary

- Polynomial interpolation: General idea and methods, Lagrange interpolation
- Piecewise polynomial interpolation: Construction of piecewise polynomial (linear and cubic), evaluation of a piecewise function, `ncspline`, `seval`

References

- [1] George E. Forsyth and Michael A. Malcolm and Cleve B. Moler. Computer Methods for Mathematical Computations. Prentice-Hall, Inc., 1977.
Ch 4.