OLD DOMINION UNIVERSITY

Interpolation

A. Godunov

1. Basics of interpolation
2. Direct-fit polynomial interpolation
3. Lagrange interpolation
4. Divided difference polynomial interpolation
5. Cubic spline interpolation

1

Part 1:

Basics of interpolation

2

Data types

Data types:

- Continuous data: analytic functions (e.g. $f(x) = \sin x$)
- In many problems in engineering and science, the data being considered are known only at a set of discrete points, not as a continuous function, $f_i = f(x_i)$ $(i = 1,2, \dots)$.
  Discrete data: data tables (e.g. observations, results of calculations)

Attention:

Computers have limited memory for working with numbers.
Thus, computers operate with discrete sets of data.

3

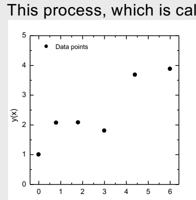
3

Discrete data

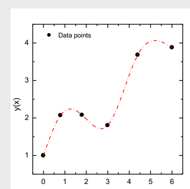
A problem arises when the value of the function is needed at any value of x between the discrete values in the table.

The actual function is not known and cannot be determined from the tabular values. However, the actual function can be approximated by some known function, and the value of the approximating function can be determined at any desired value *of* x.

This process, which is called



interpolation

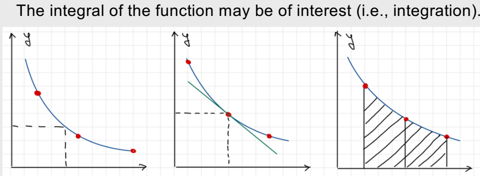


4

Applications

In many applications, the values of the discrete data at the specific points are not all that is needed.

- Values of the function at points other than the known discrete points may be needed (i.e., interpolation).
- The derivative of the function may be required (i.e., differentiation).
- The integral of the function may be of interest (i.e., integration).



(a) Interpolation (b) Differentiation (c) Integration

5

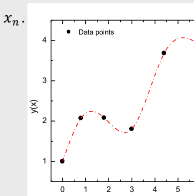
5

Key idea for interpolation

Find such approximating function $g(x)$ that

- The interpolating function passes exactly through all of the discrete points, i.e. $g(x_i) = f(x_i)$ at each data point.
- $g(x)$ is a *good** approximation for any other x between original data points

Then interpolation lets you find an approximate value for the function $f(x)$ at any point x within the interval $x_1, x_2, \dots x_n$.



Notes

* How do we know if $g(x)$ is a *good* one?

** there is a differences between: *interpolation*, *extrapolation* and *data fitting*.

6

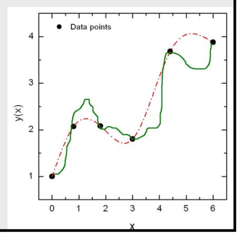
Attention!

Data points can be interpolated by an infinite number of functions since the actual function is NOT known and CANNOT be determined from the tabular data.

In fact, any analytical function can be used as an approximating function.

Interpolation ≡ Approximation

There is no exact and unique solution!



7

Equally or unequally spaced data points

A set of discrete data may be equally spaced or unequally spaced in the independent variable x.

- Unequally spaced data - several procedures can be used:
  (a) direct fit polynomials, (b) Lagrange polynomials, and (c) divided difference polynomials.
  Methods such as these require a considerable amount of effort.
- Equally spaced data - procedures based on differences can be used, for example, Newton divided difference methods.
  These methods are quite easy to apply.

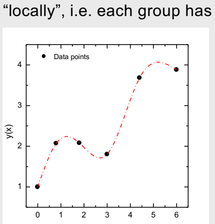
8

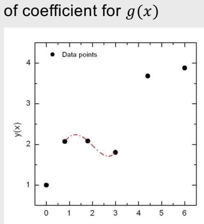
8

Interpolation is a two step procedure

1. Selects an approximating function $g(x)$
2. Find proper coefficients*.

*Attention! Normally we don't use all available N points to produce a single set of coefficients for the whole interval ("global" interpolation), instead you choose a group of points (n<N) to interpolate n-1 intervals "locally", i.e. each group has it's own set of coefficient for $g(x)$





9

9

Step 1: Selecting $g(x)$

How to choose $g(x)$?

- $g(x)$ may have some standard form (e.g. a polynomial function)
  Most interpolation methods are grounded on 'smoothness' of interpolated functions. (However, it does not work well all the time)
- or be specific for the problem (then we need some ideas about data)

Approximating functions should have the following properties:

1. The approximating function should be easy to determine.
2. It should be easy to evaluate.
3. It should be easy to differentiate.
4. It should be easy to integrate.

10

10

Linear combination

Linear combination of functions (often elementary functions) is the most common form of $g(x)$

$$g(x) = a_1 h_1(x) + a_2 h_2(x) + \cdots + a_k h_k(x)$$

where $h_i(x)$ are known functions.

11

11

Part 2:

Direct-fit polynomial interpolation

12

2.1 Direct-fit polynomial interpolation

The general form of n-th degree polynomial is

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^n$$

where a_i are constant coefficients

The property of polynomials that makes them suitable as approximating functions is stated by the *Weierstrass approximation theorem:*

If $f(x)$ is a continuous function in the closed interval $a \le x \le b$, then for every $\varepsilon > 0$ there exists a polynomial $P_n(x)$, where the value of n depends on the value of ε, such that for all x

$$|P_n(x) - f(x)| < \varepsilon$$

13

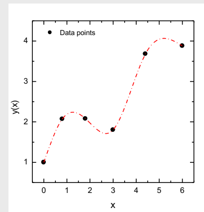
13

Uniqueness theorem

Polynomials satisfy a *uniqueness theorem:* polynomial of degree n passing exactly through $n+\ 1$ discrete points is *unique.*

The polynomial through a specific set of points may take many different forms, but all forms are equivalent.

Any form can be manipulated into any other form by simple algebraic rearrangement.



14

14

Differentiation and integration of polynomials

Differentiation and integration of polynomials is straightforward.

$$\frac{d}{dx}(a_k x^k) = k a_k x^{k-1}$$

$$\int a_k x^k dx = \frac{a_k}{k+1} x^{k+1} + constant$$

15

15

Orders of polynomial interpolation

n^th order interpolation

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^n$$

First-order (linear interpolation): two coefficients are needed

$$P_1(x) = a_0 + a_1 x$$

Second-order (quadratic interpolation)

$$P_2(x) = a_0 + a_1 x + a_2 x^2$$

Third-order (cubic interpolation)

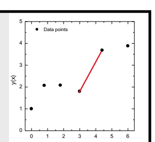
$$P_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

…

16

16

Linear interpolation



Two coefficients

$$g(x) = P_1(x) = a_0 + a_1 x.$$

The idea of linear interpolation is to approximate data at point x by a straight line passing through two data points x_j and x_{j+1} closest to x. The coefficients a_0 and a_1 can be found from the system of equations

$$g(x_j) = f_j = a_0 + a_1 x_j$$

$$g(x_{j+1}) = f_{j+1} = a_0 + a_1 x_{j+1}$$

Solving for a_0 and a_1 gives the function $g(x)$ on $[x_j, x_{j+1}]$ as

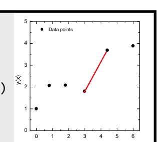
$$g(x) = f_j + \frac{x - x_j}{x_{j+1} - x_j}(f_{j+1} - f_j)$$

$$g(x) = f_j \frac{x - x_{j+1}}{x_j - x_{j+1}} + f_{j+1} \frac{x - x_j}{x_{j+1} - x_j} \quad \text{or as symmetric form}$$

17

17

Example: C++



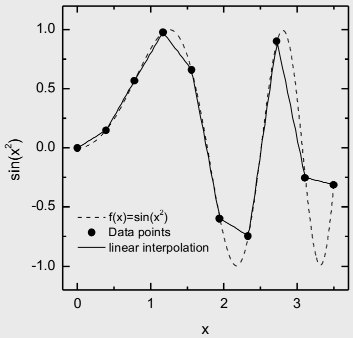
```
double int1(double x, double xi[], double yi[], int imax)
{
    double y;
    int j;
// if x is ouside the xi[] interval
    if (x <= xi[0])      return y = yi[0];
    if (x >= xi[imax-1]) return y = yi[imax-1];
// loop to find j so that x[j-1] < x < x[j]
    j = 0;
    while (j <= imax-1)
    {
     if (xi[j] >= x) break;
     j = j + 1;
     }
 y = yi[j-1]+(yi[j]-yi[j-1])*(x-xi[j-1])/(xi[j]-xi[j-1]);
    return y;
}
```

Note that a bisectional approach is more efficient to search an array.

18

18

Example:



19

19

Linear interpolation: summary

The linear interpolation may work well for very smooth functions when the second and higher derivatives are small.

It is worthwhile to note that for each data interval one has a different set of coefficients a_0 and a_1. This is the principal difference from data fitting where the same function, with the same coefficients, is used to fit the data points on the whole interval $[x_1, x_n]$.

We may improve quality of linear interpolation by increasing number of data points x_i on the interval.

HOWEVER!!! It is much better to use higher-order interpolations.

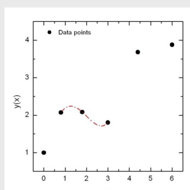
example from F. S. Acton "Numerical methods that work"
"A table of sin(x) covering the first quadrant, for example, requires 541 pages if it is to be linearly interpolable to eight decimal places. If quadratic interpolation is used, the same table takes only one page."

20

20

Quadratic interpolation as direct fit

We need three points for the second order interpolation.
What points?



The system of equations

$$f_j = a_0 + a_1 x_j + a_2 x_j^2$$

$$f_{j+1} = a_0 + a_1 x_{j+1} + a_2 x_{j+1}^2$$

$$f_{j+2} = a_0 + a_1 x_{j+2} + a_2 x_{j+2}^2$$

This system can be solved analytically

$$g(x) = f_j \frac{(x - x_{j+1})(x - x_{j+2})}{(x_j - x_{j+1})(x_j - x_{j+2})} + f_{j+1} \frac{(x - x_j)(x - x_{j+2})}{(x_{j+1} - x_j)(x_{j+1} - x_{j+2})} + f_{j+2} \frac{(x - x_j)(x - x_{j+1})}{(x_{j+2} - x_j)(x_{j+2} - x_{j+1})} \quad (4.9)$$

21

21

Direct fit polynomial interpolation

The direct fit polynomial method, while quite straightforward in principle, has several disadvantages.

- It requires a considerable amount of effort to solve the system of equations for the coefficients.
- For a high-degree polynomial (n greater than about 4), the system of equations can be ill-conditioned, which causes large errors in the values of the coefficients.

22

22

Part 3:

Lagrange polynomial interpolation

23

Lagrange Polynomials

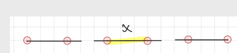
There is a simpler procedure comparing to the direct fit polynomials

Using Lagrange polynomial, which can be fit to unequally spaced data or equally spaced data.

$$g(x) = f(x_1)\lambda_1(x) + f(x_2)\lambda_2(x) + \cdots + f_n\lambda_n(x)$$

$$\lambda_n(x) = \prod_{j(\neq i)=1}^{n} \frac{x - x_j}{x_i - x_j}$$

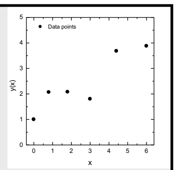
No system of equations must be solved to evaluate the polynomial.



24

24

Example: C++

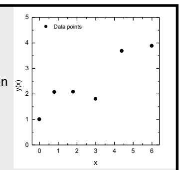



```
double polint(double x, double xi[], double yi[],
              int isize, int npoints)
{
    double lambda[isize];
    double y;
    int j, is, il;
// check order of interpolation
    if (npoints > isize) npoints = isize;
// if x is ouside the xi[] interval
    if (x <= xi[0])       return y = yi[0];
    if (x >= xi[isize-1]) return y = yi[isize-1];
// loop to find j so that x[j-1] < x < x[j]
    j = 0;
    while (j <= isize-1)
    {
     if (xi[j] >= x) break;
     j = j + 1;
     }
```

25

25

Example: C++

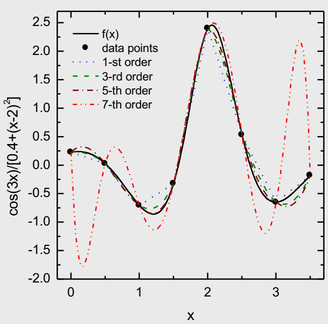



```
// shift j to correspond to (npoint-1)th interpolation
     j = j - npoints/2;
// if j is ouside of the range [0, ... isize-1]
     if (j < 0) j=0;
     if (j+npoints-1 > isize-1 ) j=isize-npoints;
     y = 0.0;
     for (is = j; is <= j+npoints-1; is = is+1)
     {
         lambda[is] = 1.0;
         for (il = j; il <= j+ npoints-1; il = il + 1)
         {
          if(il != is) lambda[is] = lambda[is]*
                       (x-xi[il])/(xi[is]-xi[il]);
         }
         y = y + yi[is]*lambda[is];
     }
    return y;
}
```

26

26

Example:



27

27

Summary

- Moving from the first -order to the third and 5th order improves interpolated values to the original function.
- However, the 7th order interpolation instead being closer to the function $f(x)$ produces wild oscillations known as Runge phenomenon – extreme "polynomial wiggle" associated with high-degree polynomial interpolation at evenly-spaced points.
- Rule of thumb: do not use high order interpolation. Fifth order may be considered as a practical limit.
- If you believe that the accuracy of the 5th order interpolation is not sufficient for you, then you should rather consider some other method of interpolation.

28

28

Advantages and disadvantages

The main advantage of the Lagrange polynomial is that the data may be unequally spaced.

There are several disadvantages.

1. All of the work must be redone for each degree polynomial.
2. All the work must be redone for each value of x.

The first disadvantage is eliminated by Neville's algorithm (which has some computational advantages over the Lagrange polynomials)

Both disadvantages are eliminated by using divided differences.

29

29

Part 4:

Divided difference polynomials

30

Divided difference coefficients

A divided difference is defined as the ratio of the difference in the function values at two points divided by the difference in the values of the corresponding independent variable.

Thus, the first divided difference at point i is defined as

$$f[x_i, x_{i+1}] = f_i^{(1)} = \frac{f_{i+1} - f_i}{x_{i+1} - x_i}$$

The second divided difference is defined as

$$f[x_i, x_{i+1}, x_{i+2}] = f_i^{(2)} = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$$

Similar expressions can be obtained for divided differences of any order.

Approximating polynomials for nonequally spaced data can be constructed using divided differences.

31

31

Tables of divided differences

Example:

x_i	$f_i^{(0)}$	$f_i^{(1)}$	$f_i^{(2)}$	$f_i^{(3)}$
x_1	$f_1^{(0)}$	$f_1^{(1)}$	$f_1^{(2)}$	$f_1^{(3)}$
x_2	$f_2^{(0)}$	$f_2^{(1)}$	$f_2^{(2)}$	
x_3	$f_3^{(0)}$	$f_3^{(1)}$		
x_4	$f_4^{(0)}$			

32

32

Example: Fortran

```
double precision d(n,n), x(n), f(n)
integer i,j
..................
d = 0.0
! initialization of d(n,1)
do i=1,n
  d(i,1) = f(i)
end do
! calculations
do j=2,n
  do i=1,n-j+1
    d(i,j)=(d(i+1,j-1)-d(i,j-1))/(x(i+1+j-2)-x(i))
  end do
end do
! print results
do i=1,n
  write(*,200) (d(i,j),j=1,n-i+1)
end do
200 format (5f10.6)
```

33

33

Example: results for 8 points

```
x:    3.20,3.30,3.35,3.40,3.50,3.60,3.65,3.70
f:    0.312500,0.303030,0.298507,0.294118,
      0.285714,0.277778,0.273973,0.270270
f         f1.       f2.       f3.       f4
0.312500 -0.094700  0.028265 -0.007311 -0.006774 ...
0.303030 -0.090460  0.026802 -0.009343  0.010684 ...
0.298507 -0.087780  0.024934 -0.006138 -0.001741 ...
0.294118 -0.084040  0.023399 -0.006660 -0.000053
0.285714 -0.079360  0.021734 -0.006676
0.277778 -0.076100  0.020399
0.273973 -0.074060
0.270270
```

34

34

Divided difference polynomials

Let's define a power series for $P_n(x)$ such that the coefficients are identical to the divided differences $f_i^{(n)}$.

$$P_n(x) = f_i^{(0)} + (x - x_0)f_i^{(1)} + (x - x_0)(x - x_1)f_i^{(2)} + \cdots$$
$$+ (x - x_0)(x - x_1) \dots (x - x_{n-1})f_i^{(n)}.$$

$P_n(x)$ is clearly a polynomial of degree n.

We can easily demonstrate that that $P_n(x)$ passes exactly through the data points $x_0, x_1, \dots$

Since $P_n(x)$ is a polynomial of degree n and passes exactly through the $n+1$ data points, it is obviously one form of the unique polynomial passing through the data points.

35

35

Three forms of polynomials (to compare)

1. Direct-fit polynomials

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^n$$

2. Lagrange polynomials

$$g(x) = f_1 \lambda_1(x) + f_2 \lambda_2(x) + \cdots + f_n \lambda_n(x)$$

$$\lambda_n(x) = \prod_{j(\neq i)=1}^{n} \frac{x - x_j}{x_i - x_j}$$

3. Divided difference polynomials

$$P_n(x) = f_i^{(0)} + (x - x_0)f_i^{(1)} + (x - x_0)(x - x_1)f_i^{(2)} + \cdots$$
$$+ (x - x_0)(x - x_1) \dots (x - x_{n-1})f_i^{(n)}.$$

36

36

Divided differences – equally spaced points

Fitting approximating polynomials to tabular data is considerably simpler when the values of the independent variable are equally spaced.

Implementation of polynomial fitting for equally spaced data is best accomplished in terms of differences.

x	f(x)			
x_0	f_0			
x_1	f_1	$f_1 - f_0$		
x_2	f_2	$f_2 - f_1$	$f_2 - 2f_1 + f_0$	$f_3 - 3f_2 + 3f_1 - f_0$
x_3	f_3	$f_3 - f_2$	$f_3 - 2f_2 + f_1$	

37

37

Three interpretations

The numbers appearing in a difference table are unique.

However, three different interpretations can be assigned to these numbers, each with its unique notation.

1. The forward difference relative to point i is
$$\Delta f_i = (f_{i+1} - f_i)$$
2. the backward difference relative to point $i+1$ is
$$\nabla f_{i+1} = (f_i - f_{i+1}) = -(f_{i+1} - f_i)$$
3. The centered difference relative to point $i + 1/2$ is
$$\delta f_{i+\frac{1}{2}} = (f_{i+1} - f_i)$$

38

38

Example $f(x) = \frac{1}{x}$ for $3.1 < x < 3.9$

x	$f(x)$	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$	$\Delta^4 f(x)$	$\Delta^5 f(x)$
3.1	0.322581					
		−0.010081				
3.2	0.312500		0.000611			
		−0.009470		−0.000053		
3.3	0.303030		0.000558		0.000003	
		−0.008912		−0.000050		0.000007
3.4	0.294118		0.000508		0.000010	
		−0.008404		−0.000040		−0.000010
3.5	0.285714		0.000468		0.000000	
		−0.007936		−0.000040		0.000008
3.6	0.277778		0.000428		0.000008	
		−0.007508		−0.000032		−0.000008
3.7	0.270270		0.000396		0.000000	
		−0.007112		−0.000032		
3.8	0.263158		0.000364			
		−0.006748				
3.9	0.256410					

observations: the first and second differences are quite smooth. The third differences, while monotonic, are not very smooth. The fourth differences are not monotonic, and the fifth differences are extremely ragged. The magnitudes of the higher-order differences decrease rapidly. If the differences are not smooth and not decreasing, several possible explanations exist: 1. The original data set has errors. 2. The increment Δx may be too large. 3. There may be a singularity in f(x) or its derivatives in the range of the table.

Difference tables are useful for evaluating the quality of a set of tabular data.

39

39

The Newton Forward-Difference Polynomial

Given $n+1$ data points, then one form of the unique nth-degree polynomial that passes through the n+1 points (separated by $\Delta x = h$) is given by

$$P_n(x) = f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 + \frac{s(s-1)(s-2)}{3!}\Delta^3 f_0 + \\ + \cdots + \frac{s(s-1)(s-2)\ldots(s-(n-1))}{n!}\Delta^n f_0$$

where

$$s = \frac{x - x_0}{\Delta x} = \frac{x - x_0}{h}, \qquad x = x_0 + sh$$

Equation does not look anything like the direct fit polynomial, the Lagrange polynomial, or the divided difference polynomial. However, if it is a polynomial of degree n and passes exactly through the n + l data points, it must be one form of the unique polynomial that passes through this set of data.

40

40

Comparing Lagrange and divided differences

```
Quality of Lagrange interpolation:
average difference from f(x)
   1st         3rd         5th         7th
0.1410      0.0848      0.1434      0.2808

Quality of interpolation: average difference from f(x)
Orders of divided difference interpolation
     1           2           3           4          5
0.1410      0.1484      0.0848      0.1091     0.1433
     6           7           8           9
0.2022      0.2808      0.3753      0.4526
```

41

41

The Newton Backward-Difference Polynomial

The Newton forward-difference polynomial, can be applied at the top or in the middle of a set of tabular data, where the downward-sloping forward differences exist. However, at the bottom of a set of tabular data, the required forward differences do not exist, and we need to used the Newton backward-difference polynomial.

$$P_n(x) = f_0 + s\nabla f_0 + \frac{s(s-1)}{2!}\nabla^2 f_0 + \frac{s(s-1)(s-2)}{3!}\nabla^3 f_0 + \\ + \cdots + \frac{s(s-1)(s-2)\ldots(s-(n-1))}{n!}\nabla^n f_0$$

where

$$s = \frac{x - x_0}{\Delta x} = \frac{x - x_0}{h}, \qquad x = x_0 + sh$$

42

42

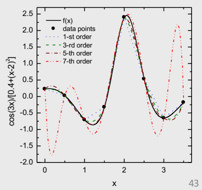
Summary

A major advantage of the Newton forward and backward difference polynomials is that each higher order polynomial is obtained from the previous lower-degree polynomials simply by adding the new term

However, both polynomial, Lagrange and divided difference polynomials share the same problem called Runge phenomenon – extreme "polynomial wiggle" associated with high-degree polynomial interpolation at evenly-spaced points.

Other difference polynomials:

- Stirling centered-difference polynomials
- Bessel centered-difference polynomials



43

43

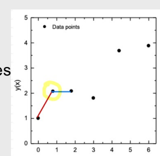
Part 5:

Cubic spline interpolation

44

Problems with polynomial approximation

1. Problems can arise when a single high-degree polynomial is fit to a large number of points.
   High-degree polynomials would obviously pass through all the data points themselves, but they can oscillate wildly between data points due to round-off errors and overshoot.
2. In such cases, lower-degree polynomials can be fit to subsets of the data points.
   If the lower-degree polynomials are independent of each other, a piecewise approximation is obtained.
3. One of the principal drawbacks of the polynomial interpolation is related to discontinuity of derivatives at connecting data points x_j.



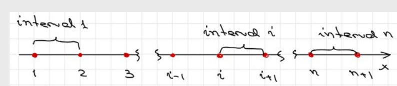
45

Spline

An alternate approach is to fit a lower-degree polynomial to connect each pair of data points, i.e., $P_n(x)$ for every interval and to require the set of lower-degree polynomials to be consistent with each other in some sense.

This type of polynomial is called a spline function, or simply a spline.

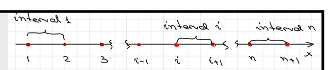
The procedure for deriving coefficients of spline interpolations uses information from all data points, i.e. nonlocal information to guarantee global smoothness in the interpolated function up to some order of derivatives.



46

46

Degrees of splines



Linear splines are not really splines since points are connected by straight line segments (aka linear interpolation).

Quadratic splines - second-order approximating polynomials for every interval $f_i(x) = a_i + b_i x + c x_i^2$ $(i = 1,2,\dots,n)$. The slopes of the quadratic splines can be forced to be continuous at each data point, but the curvatures (i.e., the second derivatives) are still discontinuous.

A cubic spline yields a third-degree polynomial connecting each pair of data points.

$$f_i(x) = a_i + b_i x + c x_i^2 + d_i x_i^3 \quad (i = 1,2,\dots,n)$$

The slopes and curvatures of the cubic splines can be forced to be continuous at each data point.
Looks very promising, no need to go to higher degrees of splines!

47

47

History …

The name spline comes from the thin flexible rod, called a spline, used by draftsmen to draw smooth curves through a series of discrete points.

The spline is placed over the points and either weighted or pinned at each point. Due to the flexure properties of a flexible rod (typically of rectangular cross section), the slope and curvature of the rod are continuous at each point.

A smooth curve is then traced along the rod, yielding a spline curve.

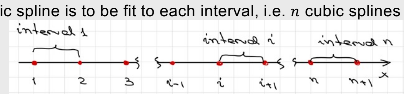
48

48

Definitions

$n+1$ total points, x_i $(i = 1,2,\ldots,n+1)$,
n intervals
$n-1$ interior grid points, x_i $(i = 2,3,\ldots,n)$.

A cubic spline is to be fit to each interval, i.e. n cubic splines



$$f_i(x) = a_i + b_i x + c x_i^2 + d x_i^3 \quad (i = 1,2,\ldots,n)$$

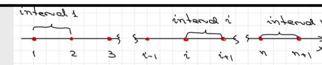
Since each cubic spline has four coefficients and there are n cubic splines, there are 4n coefficients to be determined.

Thus, 4n boundary conditions, or constraints, must be available.

49

49

Finding coefficients



1. The function values, $f(x_i) = f_i$ $(i = 2,3,\ldots,n)$, must be the same in the two splines on either side of x_i at all of the $n-1$ interior points. This constraint yields $2(n-1)$ conditions.
2. The first and last spline must pass through the first x_1 and last x_{n+1} points. That is, $f_1(x_1) = f_1$, $f_{n+1}(x_{n+1}) = f_{n+1}$ This constraint yields 2 conditions.
3. The first derivative of the two splines on either side of point x_i must be equal at all of the $n-1$ interior points. This constraint yields $(n-1)$ conditions.
4. The second derivative of the two splines on either side of point x_i must be equal at all of the $n-1$ interior points. Another $(n-1)$ conditions.

TOTAL = $2(n-1)+2+(n-1)+(n-1) = 4n-2$ conditions. We need $4n$.

How to set two additional conditions?

50

50

The last two conditions (many possibilities)

1. **Natural spline** - the second order derivatives are zero on boundaries $f_1'' = 0$ and $f''_{n+1} = 0$
2. **Curvature-adjusted cubic spline**.
   Input some values for the second order derivatives f_1'' and f''_{n+1} at boundaries.
3. **Clamped cubic spline**
   Input values for the first order f_1' and f'_{n+1} derivates at boundaries
4. **Parabolically terminated cubic spline**
   Specifying $d_1 = d_n = 0$ that is the same as $c_1 = c_2, c_{n-1} = c_n$
5. **Not-a-knot cubic spline**.
   $d_1 = d_2, d_{n-1} = d_n$
6. And many more …

Note that MATLAB's default spline command constructs a not-a-knot spline when given four or more points.

51

51

a little math …

$$s_i(x) = a_i + b_i(x-x_i) + \frac{c_i}{2}(x-x_i)^2 + \frac{d_i}{6}(x-x_i)^3 \qquad (1)$$

$x_{i-1} \le x \le x_i \qquad i = 1,2,\ldots N.$

Need to find a_i, b_i, c_i, d_i.

$$s_i'(x) = b_i + c_i(x-x_i) + \frac{d_i}{2}(x-x_i)^2$$

$$s_i''(x) = c_i + d_i(x-x_i)$$

$$s_i'''(x) = d_i$$

by the definition (interpolation) $a_i = f(x_i)$

1) $s(x)$ must be continuous at x_i $\quad s_i(x_i) = s_{i+1}(x_i)$

$$a_i = a_{i+1} + b_{i+1}(x_i - x_{i+1}) + \frac{c_{i+1}}{2}(x_i - x_{i+1})^2 + \frac{d_{i+1}}{6}(x_i - x_{i+1})^3$$

52

52

using $h_i = x_i - x_{i-1}$

$$h_i b_i - \frac{h_i^2}{2}c_i + \frac{h_i^3}{6}d_i = f_i - f_{i-1} \qquad (2)$$

2) $s_i'(x_i) = s_{i+1}'(x_i) \qquad i = 1,2,\ldots N-1$

$$c_i h_i - \frac{d_i}{2}h_i^2 = b_i - b_{i-1} \qquad i = 2,3,\ldots N \qquad (3)$$

3) $s_i''(x_i) = s_{i+1}''(x_i)$

$$d_i h_i = c_i - c_{i-1} \qquad i = 2,3,\ldots N \qquad (4)$$

additional equations (conditions at the ends)

$$s''(a) = s''(b) = 0$$

$$s_1''(x_0) = 0, \quad s_N''(x_N) = 0 \qquad c_1 - d_1 h_1 = 0, \quad c_N = 0$$

53

53

The system of equations to find coefficients

$$h_i d_i = c_i - c_{i-1} \qquad i = 1,2,\ldots N \quad c_0 = c_N = 0 \qquad (5)$$

$$h_i c_i - \frac{h_i^2}{2}d_i = b_i - b_{i-1} \qquad i = 2,3,\ldots N \qquad (6)$$

$$h_i b_i - \frac{h_i^2}{2}c_i + \frac{h_i^3}{6}d_i = f_i - f_{i-1} \qquad i = 1,2,\ldots N \qquad (7)$$

Solve the system for c_i $\quad i = 1,2,\ldots N-1$

Consider two equations (7) for points i and $i-1$

$$b_i = \frac{h_i}{2}c_i + \frac{h_i^2}{6}d_i = \frac{f_i - f_{i-1}}{h_i}$$

$$b_{i-1} = \frac{h_{i-1}}{2}c_{i-1} + \frac{h_{i-1}^2}{6}d_{i-1} = \frac{f_{i-1} - f_{i-2}}{h_{i-1}}$$

and sustract the second equation from the first

$$b_i - b_{i-1} = \frac{1}{2}(h_i c_i - h_{i-1}c_{i-1}) - \frac{1}{6}(h_i^2 d_i - h_{i-1}^2 d_{i-1}) + \frac{f_i - f_{i-1}}{h_i} - \frac{f_{i-1} - f_{i-2}}{h_{i-1}}$$

54

54

Use the difference $b_i - b_{i-1}$ in the right side of (6)

$$h_i c_i + h_{i-1} c_{i-1} - \frac{h_{i-1}^2}{2} d_{i-1} - \frac{2h_i^2}{3} d_i = 2\left(\frac{f_i - f_{i-1}}{h_i} - \frac{f_{i-1} - f_{i-2}}{h_{i-1}}\right) \qquad (8)$$

Then from (5)

$h_i^2 d_i = h_i (c_i - c_{i-1}),$

$h_{i-1}^2 d_{i-1} = h_{i-1}(c_{i-1} - c_{i-2})$

Substitute in (8)

$$h_{i-1} c_{i-2} + 2(h_{i-1} + h_i) c_{i-1} + h_i c_i = 6\left(\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i}\right) \qquad (9)$$

for $i = 1, 2, \ldots N-1, \quad c_0 = c_N = 0$

This is a tridiagonal system of equations (use Thomas method)

then $d_i = \frac{c_i - c_{i-1}}{h_i}, \quad b_i = \frac{h_i}{2} c_i - \frac{h_i^2}{6} d_i + \frac{f_i - f_{i-1}}{h_i}$ for $i = 1, 2, \ldots N,$

55

55

Implementations

Many numerical libraries have spline interpolation programs

Both C++ and Fortran versions of the spline interpolation van be found at

C++

https://ww2.odu.edu/~agodunov/book/programs.html

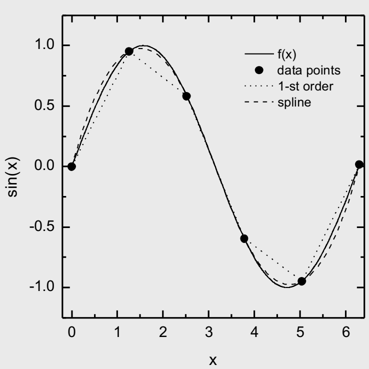
Fortran

https://ww2.odu.edu/~agodunov/computing/programs/index.html

56

56

Example



57

57

Comments

Generally, spline does not have advantages over polynomial interpolation when used for smooth, well-behaved data, or when data points are close on x scale.

The advantage of spline comes into the picture when dealing with "sparse" data, when

- there are only a few points for smooth functions
- or when the number of points is close to the number of expected maximums.

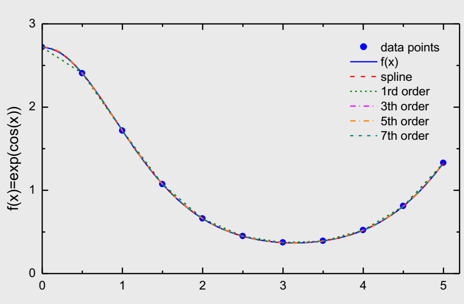
58

58

Example: spline and polynomial interpolation

Quality of interpolation: average difference from f(x)

spline	1st	3rd	5th	7th
0.0004	0.0171	0.0013	0.0012	0.0004



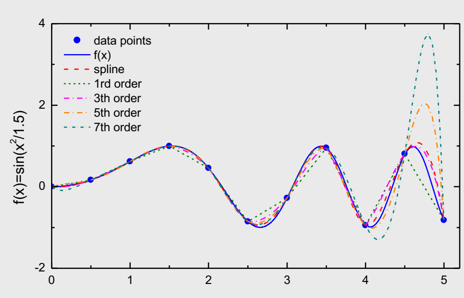
59

59

Example: spline and polynomial interpolation

Quality of interpolation: average difference from f(x)

spline	1st	3rd	5th	7th
0.0526	0.1410	0.0848	0.1434	0.2808



60

60

Bézier curves

Bézier curves are splines that allow the user to control the slopes at the knots. In return for the extra freedom, the smoothness of the first and second derivatives across the knot, which are automatic features of the cubic splines of the previous section, are no longer guaranteed.

Bézier splines are appropriate for cases where corners (discontinuous first derivatives) and abrupt changes in curvature (discontinuous second derivatives) are occasionally needed.

Bézier curves are named after French engineer Pierre Bézier, who used it in the 1960s for designing curves for the bodywork of Renault cars.

Bézier curves – enormous number of applications (computer fonts, computer-aided design, animation, user interfaces, robotics, ...)

61

61

Bézier curves (history)

Bézier popularized but did not actually create the Bézier curve.
He using such curves to design automobile bodies for Renault (French car manufacturer) .

The curves were first developed in 1959 by Paul de Casteljau using de Casteljau's algorithm (that time he work for Citroën – a rival French car manufacturer).

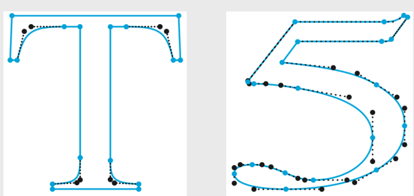
There are comments that the method was developed independently by Bézier and Casteljau but Renault and Citroën companies wanted to keep it secret.

62

62

BeÉxample - fonts

Times-Roman T and 5 made with Bézier splines. Blue circles are spline endpoints, and black circles are control points.



note: PostScript fonts are built directly from Bézier curves.

63

63

Part 6:

And more …

64

Chebyshev's interpolation

It turns out that the choice of base point spacing can have a significant effect on the interpolation error.

Chebyshev interpolation refers to a particular optimal way of spacing the points.

Chebyshev's polynomials (defined on $[-1, +1]$)

$T_n(x) = \cos(n \arccos x)$

$T_1(x) = x$

$T_2(x) = 2x^2 - 1$

$T_3(x) = 4x^3 - 3x$

…

65

65

Chebyshev's interpolation (cont.)

Chebyshev interpolation is a good way to turn general functions into a small number of floating-point operations, for ease of computation.

An upper bound for the error made is easily available, is usually smaller than for evenly spaced interpolation, and can be made as small as desired.

Chebyshev polynomials are widely used in physics!

66

66

Rational function interpolation

Rational functions may well interpolate functions with poles

$$f(x) = \frac{a_0 + a_1 x + a_2 x^2 + \cdots a_n x^n}{b_0 + b_1 x + b_2 x^2 + \cdots b_m x^m}$$

that is with zeros of the denominator

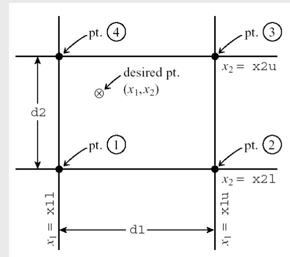
$$b_0 + b_1 x + b_2 x^2 + \cdots b_m x^m = 0$$

67

67

Interpolation in two or more dimensions

- bilinear interpolation
- bicubic interpolation
- bicubic spline
- …



68

68

Applications for Interpolation

Interpolation has many applications both in physics, science, and engineering.

Interpolation is a corner's stone in numerical integration (integrations, differentiation, Ordinary Differential Equations, Partial Differential Equations).

Two-dimensional interpolation methods are widely used in image processing, including digital cameras.

69

69

Extrapolation

If you are interested in function values outside the range $x_1, \ldots x_n$ then the problem is called extrapolation.

Generally, this procedure is much less accurate than interpolation.

You know how it is difficult to extrapolate (foresee) the future, for example, for the stock market.

70

70

Data fitting

If data values $f_i(x_i)$ are a result of experimental observation with some errors, then data fitting may be a better way to proceed.

In data fitting we let $g(x_i)$ to differ from measured values f_i at x_i points having one function only to fit all data points; i.e., the function $g(x)$ fits all the set of data.

Data fitting may reproduce well the trend of the data, even correcting some experimental errors.

71

71