Interpolation

1

Data types in science

- Discrete data (data tables)
  - Experiment
  - Observations
  - Calculations
- Continuous data
  - Analytics functions
  - Analytic solutions

2

From continuous to discrete …

soon after hurricane Isabel (September 2003)

3

From discrete to continuous?

?

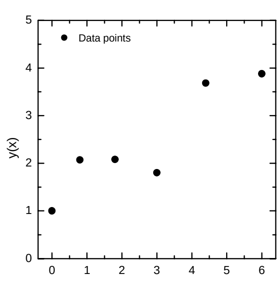
4

What do we want from discrete sets of data?

$$\{f_i(x_i)\ i = 1, n\}$$

Quite often we need to know function values at any arbitrary point *x*

Can we generate values for any *x* between x_1 and x_n from a data table?



If you think that the data values f_i in the data table are free from errors,

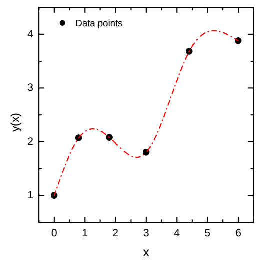
then **interpolation** lets you find an approximate value for the function ***f(x)*** at any point ***x*** within the interval $x_1 \ldots x_n$.

6

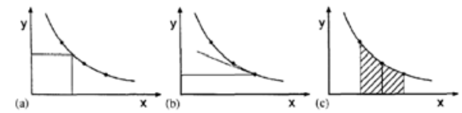
Key point!!!

The idea of interpolation is to select a function *g(x)* such that

1. $g(x_i)=f_i$ for each data point i
2. this function is a <u>good</u> approximation for any other x between original data points



Applications of approximating functions

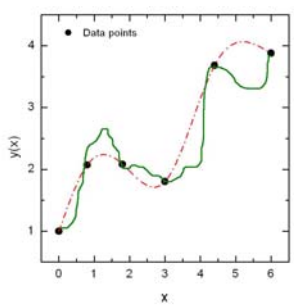


interpolation differentiation integration

8

What is a good approximation?

- What can we consider as a good approximation to the original data if we do not know the original function?
- Data points may be interpolated by an infinite number of functions



9

Important to remember

Interpolation ≡ Approximation
There is no exact and unique solution
The actual function is NOT known and CANNOT be determined from the tabular data.

10

Two step procedure

- Select a function g(x)
- Find coefficients

11

Step 1: selecting a function g(x)

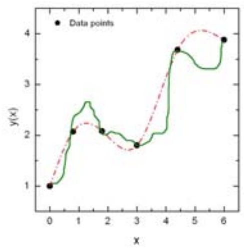
We should have a guideline to select a reasonable function *g(x)*.
We need some ideas about data!!!

- g(x) may have some standard form
- or be specific for the problem

12

Some ideas for selecting g(x)

- Most interpolation methods are grounded on '**smoothness**' of interpolated functions. However, it does not work all the time
- Practical approach, i.e. what physics lies beneath the data



Linear combination is the most common form of g(x)

- linear combination of elementary functions, or trigonometric, or exponential functions, or rational functions, …

$$\boxed{g(x) = a_1 h_1(x) + a_2 h_2(x) + a_3 h_3(x) + \ldots}$$

Three of most common approximating functions

- Polynomials
- Trigonometric functions
- Exponential functions

Approximating functions should have following properties

- It should be easy to determine
- It should be easy to evaluate
- It should be easy to differentiate
- It should be easy to integrate

Linear Interpolation: Idea

The idea of linear interpolation is to approximate data at a point x by a straight line passing through two data points x_j and x_{j+1} closest to x.

$$g(x) = a_0 + a_1 x \tag{4.1}$$

where a_0 and a_1 are coefficients of the linear functions. The coefficients can be found from a system of equations

$$g(x_j) = f_j = a_0 + a_1 x_j \tag{4.2}$$

$$g(x_{j+1}) = f_{j+1} = a_0 + a_1 x_{j+1} \tag{4.3}$$

Linear Interpolation: coefficients

Solving this system for a_0 and a_1 one have that the function $g(x)$ takes the form

$$g(x) = f_j + \frac{x - x_j}{x_{j+1} - x_j}(f_{j+1} - f_j) \tag{4.4}$$

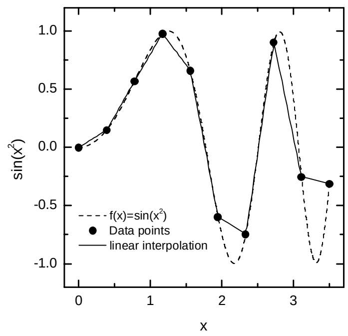
on $[x_j, x_{j+1}]$ interval.

$$g(x) = f_j \frac{x - x_{j+1}}{x_j - x_{j+1}} + f_{j+1} \frac{x - x_j}{x_{j+1} - x_j} \tag{4.5}$$

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;
}
```

bisection approach is much more efficient to search an array



19

Linear interpolation: conclusions

- 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 the 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-odrder interpolations.

20

Linear vs. Quadratic 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 having entries at one-degree intervals."

21

Polynomial Interpolation

Polynomial interpolation is a very popular method due, in part, to simplicity

$$g(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n. \qquad (4.6)$$

The condition that the polynomial $g(x)$ passes through sample points $f_j(x_j)$

$$f_j(x_j) = g(x_j) = a_0 + a_1 x_j + a_2 x_j^2 + ... + a_n x_j^n. \qquad (4.7)$$

The number of data points minus one defines the order of interpolation.
Thus, linear (or two-point interpolation) is the first order interpolation

22

Properties of polynomials

Weierstrass theorem:

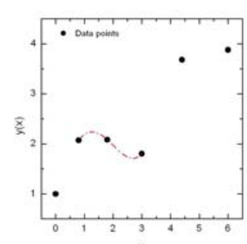
If f(x) is a continuous function in the closed interval $a \le x \le b$ then for every $\varepsilon > 0$ there exists a polynomial $P_n(x)$, where the value on n depends on the value of ε, such that for all x in the closed interval $a \le x \le b$

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

23

System of equations for the second order

$$\begin{aligned} f_j &= a_0 + a_1 x_j + a_2 x_j^2 \\ f_{j+1} &= a_0 + a_1 x_{j+1} + a_2 x_{j+1}^2 \\ f_{j+2} &= a_0 + a_1 x_{j+2} + a_2 x_{j+2}^2 \end{aligned}$$



- We need three points for the second order interpolation
- Freedom to choose points?
- This system can be solved analytically

24

Solutions

for the second order

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

and any arbitrary order (Lagrange interpolation)

$$f(x) \approx f_1\lambda_1(x) + f_2\lambda_2(x) + \ldots f_n\lambda_n(x)$$

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

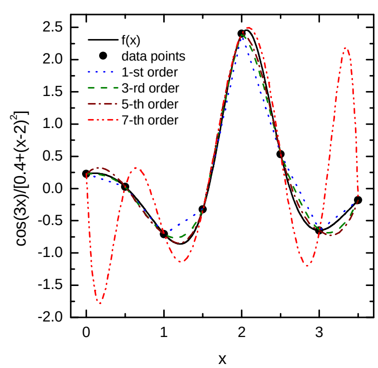
25

Lagrange Interpolation: 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;
     }
```

Example: C++ (cont.)

```
// 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;
}
```



28

note on Lagrange interpolation

- It is possible to use Lagrange formula straightforwardly, like in the example above
- There is a better algorithm - Neville's algorithm (for constructing the same interpolating polynomial)
- Sometimes Neville's algorithm confused with Aitken's algorithm
  (the latter now considered obsolete).

29

Important!

- 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.
- This situation is not uncommon for high-order polynomial interpolation.
- 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.

30

Cubic Spline

- The idea of spline interpolation is reminiscent of very old mechanical devices used by draftsmen to get a smooth shape.
- It is like securing a strip of elastic material (metal or plastic ruler) between knots (or nails).
- The final shape is quite smooth.
- Cubic spline interpolation is used in most plotting software. (cubic spline gives most smooth result)

31

Spline vs. Polynomials

- One of the principal drawbacks of the polynomial interpolation is related to discontinuity of derivatives at data points x_j.
- 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.

32

Equations

the interpolated function on $[x_j, x_{j+1}]$ interval is presented in a cubic spline form

$$g(x) = f_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

What is different from polynomial interpolation?

… the way we are looking for the coefficients!

Polynomial interpolation: 3 coefficient for 4 data points

Spline interpolation: 3 coefficients for the each interval

33

Coefficients for spline interpolation

- For the each interval we need to have a set of three parameters b_j, c_j and d_j.
  Since there are (n-1) intervals, one has to have 3n-3 equations for deriving the coefficients for j=1,n-1.
- The fact that $g_j(x_j)=f_j(x_j)$ imposes (n-1) equations.

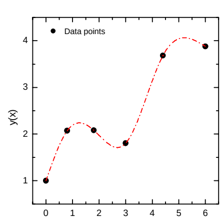
34

Central idea to spline interpolation

- the interpolated function g(x) has continuous the first and second derivatives at each of the n-2 interior points x_j.

$$g'_{j-1} x_j = g'_j x_j$$

$$g''_{j-1} x_j = g''_j x_j.$$



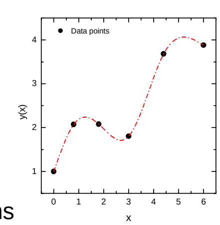
Now we have more equations

- (n-1) + 2(n-2) = 3n-5 equations

$$g_j(x_j) = f_j(x_j)$$

$$g'_{j-1} x_j = g'_j x_j$$

$$g''_{j-1} x_j = g''_j x_j.$$



- but we need 3n-3 equations

36

Cubic spline boundary conditions

Possibilities to fix two more conditions

- Natural spline - the second order derivatives are zero on boundaries.
- Input values for the first order $f^{(1)}(x)$ derivates at boundaries
- Input values for the second order $f^{(2)}(x)$ derivates at boundaries

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 $\quad s_i(x_i) = s_{i+1}(x_i)$

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

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)

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

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

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

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

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

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

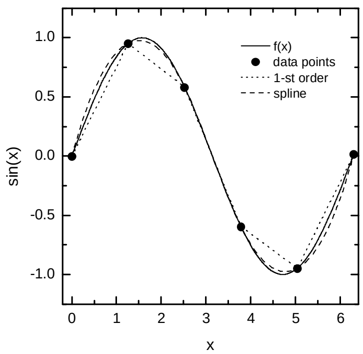
then $d_i = \dfrac{c_i - c_{i-1}}{h_i}, \qquad b_i = \dfrac{h_i}{2}c_i - \dfrac{h_i^2}{6}d_i + \dfrac{f_i - f_{i-1}}{h_i}$ for $i = 1,2,\ldots N,$

Implementation

Recommendation – do not write your own spline program but get one from ….

- Advanced program libraries!
- Books

Example: Spline interpolation



Comments to Spline interpolation

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.

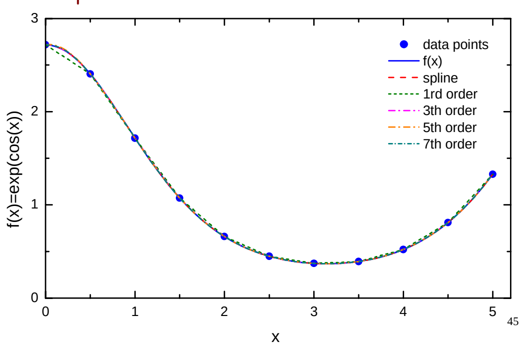
The advantage of spline comes in to the picture when dealing with "sparse" data, when

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

Example: spline and polynomial interpolation

Quality of interpolation: average difference from f(x)

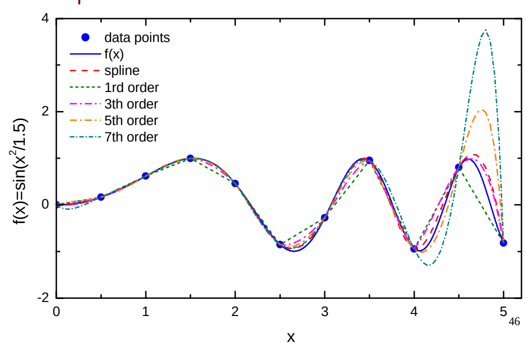
spline	1st	3rd	5th	7th
0.0004	0.0171	0.0013	0.0012	0.0004



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



Divided Differences

the first divided difference at point i

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

the second divided difference

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

in general

$$f[x_i, x_{i+1}, \ldots x_n] = \frac{f[x_{i+1}, x_{i+2}, \ldots x_n] - f[x_i, x_{i+1}, \ldots x_{n+i-1}]}{x_n - x_i}$$

other notations

$$f_i^{(0)} = f_i, \quad f_i^{(1)} = f[x_i, x_{i+1}], \quad f_i^{(2)} = f[x_i, x_{i+1}, x_{i+2}]$$

Divided Difference Polynomials

Let's define a polynomials $P_n(x)$ where the coefficients are divided differences

$$P_n(x) = f_i^{(0)} + (x - x_i) f_i^{(1)} + (x - x_i)(x - x_{i+1}) f_i^{(2)} + \ldots$$
$$+ (x - x_i)(x - x_{i+1}) \ldots (x - x_{i+n-1}) f_i^{(n)}$$

It is easy to show that $P_n(x)$ passes exactly through the data points

$$P_n(x_i) = f_i, \quad P_n(x_{i+1}) = f_{i+1} \ldots$$

Comments

Polynomials satisfy a uniqueness theorem – a polynomial of degree *n* passing exactly through *n+1* points is unique.

The data points in the divided difference polynomials do not have to be in any specific order. However, more accurate results are obtained for interpolation if the data are arranged in order of closeness to the interpolated point

Divided Difference coefficients

for four points

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

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)
```

Example:

```
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


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
```


Equally spaced x_i

Fitting approximating polynomials to tabular data is much easier when the values x_i are equally spaced

the forward difference relative to point i

$$f_{i+1} - f_i = \Delta f_i$$

the backward difference relative to point $i+1$

$$f_{i+1} - f_i = \nabla f_{i+1}$$

the central difference relative to point $i+1/2$

$$f_{i+1} - f_i = \delta f_{i+1/2}$$

example: table of differences

for four points

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

x	f	Δf	Δ²f	Δ³f
x_0	f_0			
		Δf_0		
x_1	f_1		$\Delta^2 f_0$	
		Δf_1		$\Delta^3 f_0$
x_2	f_2		$\Delta^2 f_1$	
		Δf_2		
x_3	f_3			

x	f	∇f	∇²f	∇³f
x_{-3}	f_{-3}			
		∇f_{-2}		
x_{-2}	f_{-2}		$\nabla^2 f_{-1}$	
		∇f_{-1}		$\nabla^3 f_0$
x_{-1}	f_{-1}		$\nabla^2 f_0$	
		∇f_0		
x_0	f_0			

x	f	δf	δ²f	δ³f
x_{-1}	f_{-1}			
		$\delta f_{-1/2}$		
x_0	f_0		$\delta^2 f_0$	
		$\delta f_{1/2}$		$\delta^3 f_{1/2}$
x_1	f_1		$\delta^2 f_1$	
		$\delta f_{3/2}$		
x_2	f_2			

55

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

smooth difference => “good” data

not monotonic differences => possible errors in the original data, or the step in x is too large, or may be a singularity in $f(x)$

56

The Newton Difference Polynomials

the Newton forward difference polynomials

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

the Newton backward difference polynomials

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

where $s = (x - x_0)/h, \quad x = x_0 + sh$

57

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

Other difference polynomials:

Stirrling centered-difference polynomials

Bessel centered-difference polynomials

58

Example: compare Lagrange and divided differences interpolations

```
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
```

59

Rational Function Interpolation

- A rational function g(x) is a ratio of two polynomials
- Often rational function interpolation is a more powerful method compared to polynomial interpolation.

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

60

When using rational functions is a good idea?

Rational functions may well interpolate functions with poles

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

that is with zeros of the denominator

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

61

The procedure has two principal steps

- On the first step we need to choose powers for the numerator and the denominator, i.e. n and m. You may need to attempt a few trials before coming to a conclusion.
- Once we know the number of parameters we need to find the coefficients

62

Example for n=2 and m=1

$$g(x) = \frac{a_0 + a_1 x + a_2 x^2}{b_0 + b_1 x}$$

- We need five coefficients i.e. 5 data points
- We should fix one of the coefficients since only the ratio makes sense.
- If we choose, for example, b_0 as a fixed number c then we need 4 data points to solve a system of equations to find the coefficients

63

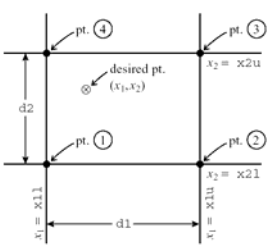
Finding coefficients using 4 data points

$$f(x_1)(c + b_1 x_1) = a_0 + a_1 x_1 + a_2 x_1^2$$
$$f(x_2)(c + b_1 x_2) = a_0 + a_1 x_2 + a_2 x_2^2$$
$$f(x_3)(c + b_1 x_3) = a_0 + a_1 x_3 + a_2 x_3^2$$
$$f(x_4)(c + b_1 x_4) = a_0 + a_1 x_4 + a_2 x_4^2$$

- After a few rearrangements the system may be written in the traditional form for a system of linear equations.
- Very stable and robust programs for rational function interpolation can be found in many standard program libraries.

64

Interpolation in two or more dimensions



- bilinear interpolation
- bicubic interpolation
- bicubic spline
- …

65

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.

66

Are we there yet?

67

What is extrapolation?

- If you are interested in function values outside the range $x_1 \dots x_n$ 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.

68

What is 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.

69