Lecture 7

Interpolation and Curve Fitting I

Interpolating Polynomials

Ceng375 *Numerical Computations* at December 2, 2010

Dr. Cem Özdoğan
Computer Engineering Department
Çankaya University

Contents

Interpolation and Curve Fitting I

- Sines, logarithms, and other nonalgebraic functions *from tables*.
- Those tables had values of the function at *uniformly spaced values* of the argument.
- Most often *interpolated* linearly:
  The value for $x = 0.125$ was computed as at the halfway point between $x = 0.12$ and $x = 0.13$.
- If the function does not vary too rapidly and the tabulated points are close enough together, this linearly estimated value would be accurate enough.
- As a conclusion:
  Data can be interpolated to estimate values.

Interpolation and Curve Fitting II

- **Interpolating Polynomials:** Describes a straightforward but computationally inconvenient way to fit a polynomial to a set of data points **so that an interpolated value can be computed**. The cost of getting the interpolant with a desired accuracy is facilitated by a variant, Neville's method.
- **Divided Differences:** These provide a more efficient way to construct an interpolating polynomial, one that allows one to readily change the degree of the polynomiall. If the data are at evenly spaced x-values, there is some simplification.

Interpolation and Curve Fitting III

- **Spline Curves:** Using special polynomials, *splines*, one can fit polynomials to data more accurately than with an interpolating polynomial. At the expense of added computational effort, some important problems that one has with interpolating polynomials is overcome.
- **Least-Squares Approximations:** are methods by which polynomials and other functions can be fitted to data that are subject to errors likely in experiments. These approximations are widely used **to analyze experimental observations**.

Interpolating Polynomials

- We have a table of x and y-values.
- Two entries in this table might be
  $y = 2.36$ at $x = 0.41$ and
  $y = 3.11$ at $x = 0.52$.
- If we desire an estimate for y at $x = 0.43$, we would use the two table values for that estimate.
- Why not interpolate as if $y(x)$ was linear between the two x-values?

where

$$y(0.43) \approx 2.36 + \frac{2}{11}(3.11 - 2.36) = 2.50 \qquad \frac{2}{11} \Longrightarrow \frac{0.43 - 0.41}{0.52 - 0.41}$$

- We will be most interested in techniques adapted to situations where the data are far from linear.
- The basic principle is to fit a polynomial curve to the data.

Interpolation versus Curve Fitting I

- Given a set of data
  $y_i = f(x_i) \qquad i = 1, \ldots, n$
  obtained from an experiment or from some calculation.
- **In curve fitting,** the approximating function **passes near the data points**, but (usually) not exactly through them. There is some uncertainty in the data.
- **In interpolation**, process inherently assumes that the data have no uncertainty. The interpolation function **passes *exactly* through** each of the known data points.

Interpolation versus Curve Fitting II

- Figure 1 shows a plot of some hypothetical experimental data, a curve fit function and interpolating with piecewise-linear function.

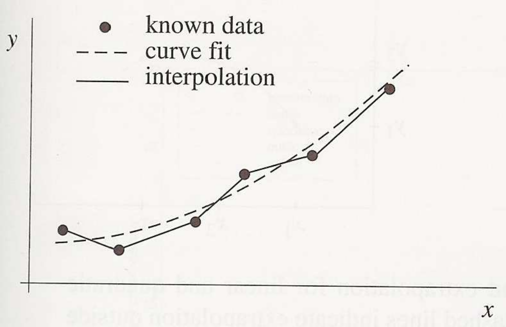

Figure: A curve fit function passes near the data points. An interpolating function passes exactly through the data points.

Fitting a Polynomial to Data I

- Interpolation involves constructing and then evaluating an interpolating function.
- **interpolant**, $y = F(x)$, determined by requiring that **it pass through the known data** (x_i, y_i).
- In its most general form, interpolation involves **determining the coefficients** $a_1, a_2, \ldots, a_n$
- in the linear combination of *n basis functions*, $\Phi(x)$, that constitute the interpolant

$$F(x) = a_1\Phi_1(x) + a_2\Phi_2(x) + \ldots + a_n\Phi_n(x)$$

  - such that $F(x) = y_i$ for $i = 1, \ldots, n$. The basis function may be polynomial

$$F(x) = a_1 + a_2x + a_3x^2 + \ldots + a_nx^{n-1}$$

  - or trigonometric

$$F(x) = a_1 + a_2e^{ix} + a_3e^{i2x} + \ldots + a_ne^{i(n-1)x}$$

  - or some other suitable set of functions.

Fitting a Polynomial to Data II

- Polynomials are often used for interpolation because they are easy to evaluate and easy to manipulate analytically.
- Suppose that we have

Table: Fitting a polynomial to data.

x	f(x)
3.2	22.0
2.7	17.8
1.0	14.2
4.8	38.3
5.6	51.7

- First, we need to select the points that determine our polynomial.
- The maximum degree of the polynomial is always one less than the number of points.

Fitting a Polynomial to Data III

- Suppose we choose the first four points. If the cubic is $ax^3 + bx^2 + cx + d$,
- We can write four equations involving the unknown coefficients a, b, c, and d;

$$
\begin{array}{l}
when\ x = 3.2 \Rightarrow a(3.2)^3 + b(3.2)^2 + c(3.2) + d = 22.0 \\
when\ x = 2.7 \Rightarrow a(2.7)^3 + b(2.7)^2 + c(2.7) + d = 17.8 \\
when\ x = 1.0 \Rightarrow a(1.0)^3 + b(1.0)^2 + c(1.0) + d = 14.2 \\
when\ x = 4.8 \Rightarrow a(4.8)^3 + b(4.8)^2 + c(4.8) + d = 38.3
\end{array}
$$

- Solving these equations gives

$$
\begin{aligned}
a &= -0.5275 \\
b &= 6.4952 \\
c &= -16.1177 \\
d &= 24.3499
\end{aligned}
$$

- and our polynomial is

$$-0.5275x^3 + 6.4952x^2 - 16.1177x + 24.3499$$

Fitting a Polynomial to Data IV

- At $x = 3.0$, the **estimated value** is 20.212.
- if we want a new polynomial that is also made to fit at the point $(5.6, 51.7)$?
- or if we want to see what difference it would make to use a quadratic instead of a cubic?
- Study this example in MATLAB;
  Start ⇒ *Toolboxes* ⇒ *CurveFitting* ⇒ *Curve Fitting Tool*.
  » x=[3.2 2.7 1.0 4.8 5.6];
  » y=[22 17.8 14.2 38.3 51.7];

Fitting a Polynomial to Data V

- Another example;

Table: Interpolation of gasoline prices.

year	price
1986	133.5
1988	132.2
1990	138.7
1992	141.5
1994	137.6
1996	144.2

- Use the polynomial order 5, why?

$$P = a_1 + a_2 y + a_3 y^2 + a_4 y^3 + a_5 y^4 + a_6 y^5$$

- Make a guess about the prices of gasoline at year of 2011.

Fitting a Polynomial to Data III

```
>> year=[1986 1988 1990 1992 1994 1996]'
>> format short e
>> A=[year.^5 year.^4 year.^3 year.^2 year ones(size(year))]
>> price=[133.5 132.2 138.7 141.5 137.6 144.2]'
>> a=A\price;
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 5.666972e-32.
>> fprintf('%12.4e \n',a)
  3.5033e-03
 -3.4839e+01
  1.3858e+05
 -2.7561e+08
  2.7408e+11
 -1.0902e+14
>> y=linspace(min(year),max(year));
>> p=polyval(a,y);
>> plot(year,price,'o',y,p,'-')
```

- Now, try with the shifted dates.
- Make the necessary corrections for the following lines
  $>> years = year - mean(year);$
- What differs in the plot and why?
- Study this example in MATLAB;
  Start $\Rightarrow$ *Toolboxes* $\Rightarrow$ *CurveFitting* $\Rightarrow$ *Curve Fitting Tool*.

Lagrangian Polynomials I

- *Straightforward approach*-the Lagrangian polynomial.
- The simplest way to exhibit the existence of a polynomial for interpolation with *unevenly* spaced data.
  - **Linear interpolation**
  - **Quadratic interpolation**
- Lagrange polynomials have two important advantages over interpolating polynomials.
  1. the construction of the interpolating polynomials does not require the solution of a system of equations.
  2. the evaluation of the Lagrange polynomials is much less susceptible to roundoff.

Lagrangian Polynomials II

- **Linear interpolation**

$$P_1(x) = c_1 x + c_2$$

- put the values

$$\begin{array}{rl} y_1 = & c_1 x_1 + c_2 \\ \hline y_2 = & c_1 x_2 + c_2 \\ \hline \end{array}$$

- then

$$c_1 = \frac{y_2 - y_1}{x_2 - x_1} \qquad c_2 = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1}$$

- substituting back and rearranging

$$P_1(x) = y_1 \frac{x - x_2}{x_1 - x_2} + y_2 \frac{x - x_1}{x_2 - x_1}$$

- redefining as

$$P_1(x) = y_1 L_1(x) + y_2 L_2(x)$$

- where Ls are the first-degree ***Lagrange interpolating polynomials***.

Lagrangian Polynomials III

- **Quadratic interpolation**

$$P_2(x) = y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x)$$

<u>where Ls are not the same with the previous Ls!!!</u>

$$L_1(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)}, \quad L_2(x) = \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)},$$

$$L_3(x) = \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}.$$

- In general

$$P_{n-1}(x) = \quad y_1 L_1(x) + y_2 L_2(x) + \ldots + y_n L_n(x) = \sum_{j=1}^{n} y_j L_j(x)$$

$$L_j(x) = \prod_{k=1, k \neq j}^{n} \frac{x - x_k}{x_j - x_k}$$

Lagrangian Polynomials IV

- Suppose we have a table of data with four pairs of x- and $f(x)$-values, with x_i indexed by variable i:

i	x	$f(x)$
0	x_0	f_0
1	x_1	f_1
2	x_2	f_2
3	x_3	f_3

Through these four data pairs we can pass a cubic.

- The Lagrangian form is

$$P_3(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} f_0 + \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} f_1$$

$$+ \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} f_2 + \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} f_3$$

Lagrangian Polynomials V

- This equation is made up of four terms, each of which is a cubic in x; hence the sum is a cubic.

$$P_3(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}f_0 + \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}f_1$$
$$+\frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}f_2 + \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}f_3$$

- The pattern of each term is to form the numerator as a product of linear factors of the form $(x - x_i)$, omitting one x_i in each term.
- The omitted value being used to form the denominator by replacing x in each of the numerator factors.
- In each term, we multiply by the f_i.
- It will have $n + 1$ terms when the degree is n.

Lagrangian Polynomials VI

- Fit a cubic through the first four points of the preceding Table 1 and use it to find the interpolated value for $x = 3.0$.
- Carrying out the arithmetic, $P_3(3.0) = 20.21$.
- MATLAB gets interpolating polynomials readily. The cubic fitted to the first four points;

```
>> x=[3.2 2.7 1.0 4.8]; y=[22.0 17.8 14.2 38.3];
>> p=polyfit(x,y,3)
>> xval=polyval(p,3.0)
```

- **Example m-file:** Interpolation of gasoline prices with Lagrange Polynomials. (demoGasLag.m lagrint.m)

Lagrangian Polynomials VII

- **Error of Interpolation**; When we fit a polynomial $P_n(x)$ to some data points, it will pass exactly through those points,
  - but between those points $P_n(x)$ will not be precisely the same as the function $f(x)$ that generated the points (unless the function is that polynomial).
  - How much is $P_n(x)$ different from $f(x)$?
  - How large is the error of $P_n(x)$?
- It is most important that you never fit a polynomial of a degree higher than 4 or 5 to a set of points.
- If you need to fit to a set of more than six points, be sure to break up the set into subsets and fit separate polynomials to these.
- You cannot fit a function that is discontinuous or one whose derivative is discontinuous with a polynomial.

Neville's Method I

- The trouble with the standard *Lagrangian polynomial technique* is that we **do not know which degree** of polynomial to use.
  - If the degree is **too low**, the interpolating polynomial does **not give good estimates** of $f(x)$.
  - If the degree is **too high**, **undesirable oscillations** in polynomial values can occur.
- **Neville's method** can overcome this difficulty.
  - It computes the interpolated value with polynomials of successively higher degree,
  - *stopping when the successive values are close together*.
- The successive approximations are actually computed by linear interpolation from the previous values.
- The Lagrange formula for linear interpolation to get $f(x)$ from two data pairs, (x_1, f_1) and (x_2, f_2), is

$$f(x) = \frac{(x - x_2)}{(x_1 - x_2)} f_1 + \frac{(x - x_1)}{(x_2 - x_1)} f_2$$

Neville's Method II

- Neville's method begins by arranging the given data pairs, (x_i, f_i).
- Such that the successive values are in order of the closeness of the x_i to x.
- Suppose we are given these data

x	$f(x)$
10.1	0.17537
22.2	0.37784
32.0	0.52992
41.6	0.66393
50.5	0.63608

and we want to interpolate for $x = 27.5$.

We first *rearrange* the data pairs in order of closeness to $x = 27.5$:

i	$\lvert x - x_i \rvert$	x_i	$f_i = P_{i0}$
0	4.5	32.0	0.52992
1	5.3	22.2	0.37784
2	14.1	41.6	0.66393
3	17.4	10.1	0.17537
4	23.0	50.5	0.63608

Neville's Method III

- Neville's method begins by renaming the f_i as P_{i0}.
- We build a table

i	x	P_{i0}	P_{i1}	P_{i2}	P_{i3}	P_{i4}
0	32.0	0.52992	0.46009	0.46200	0.46174	0.45754
1	22.2	0.37784	0.45600	0.46071	0.47901	
2	41.6	0.66393	0.44524	0.55843		
3	10.1	0.17537	0.37379			
4	50.5	0.63608				

- Thus, the value of P_{01} is computed by

$$f(x) = \frac{(27.5 - x_1)}{(x_0 - x_1)} * 0.52992 + \frac{(27.5 - x_0)}{(x_1 - x_0)} * 0.37784$$

substituting all;

$$P_{01} = \frac{(27.5 - 32.0) * 0.37784 + (22.2 - 27.5) * 0.52992}{22.2 - 32.0} = 0.46009$$

- Once we have the column of P_{i1}'s, we compute the next column.

$$P_{22} = \frac{(27.5 - 41.6) * 0.37379 + (50.5 - 27.5) * 0.44524}{50.5 - 41.6} = 0.55843$$

Neville's Method IV

- The remaining columns are computed similarly.
- The general formula for computing entries into the table is

$$p_{i,j} = \frac{(x - x_i) * P_{i+1,j-1} + (x_{i+j} - x) * P_{i,j-1}}{x_{i+j} - x_i}$$

- The **top line of the table** represents Lagrangian interpolates at $x = 27.5$ using polynomials of *degree equal to the second subscript* of the $P's$.

i	x	P_{i0}	P_{i1}	P_{i2}	P_{i3}	P_{i4}
0	32.0	0.52992	0.46009	0.46200	0.46174	0.45754

- The preceding data are for sines of angles in degrees and the correct value for $x = 27.5$ is 0.46175.