1 Interpolation and Curve Fitting

- 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.
- Data can be interpolated to estimate values
  - **Interpolating Polynomials:** Describes a straightforward but computationally awkward 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 polynomial. If the data are at evenly spaced x-values, there is some simplification.
  - **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.

1.1 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.

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

Table 1: Fitting a polynomial to data.

- Why not interpolate as if $y(x)$ was linear between the two x-values?
- 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.
- We assume that the tabulated data are exact. Later, we will consider the case where the data may have errors of measurement, which is true for most experimental results.

1.1.1 Fitting a Polynomial to Data

Suppose that we have

- 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.) 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;

$$\begin{aligned} when\ x = 3.2 &: a(3.2)^3 + b(3.2)^2 + c(3.2) + d = 22.0 \\ if\ x = 2.7 &: a(2.7)^3 + b(2.7)^2 + c(2.7) + d = 17.8 \\ if\ x = 1.0 &: a(1.0)^3 + b(1.0)^2 + c(1.0) + d = 14.2 \\ if\ x = 4.8 &: a(4.8)^3 + b(4.8)^2 + c(4.8) + d = 38.3 \end{aligned}$$

- Solving these equations gives

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

and our polynomial is

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

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?

1.1.2 Lagrangian Polynomials

- straightforward approach-the Lagrangian polynomial, the simplest way to exhibit the existence of a polynomial for interpolation with unevenly spaced data.
- 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
l	x_1	f_1
2	x_2	f_2
3	x_3	f_3

- Here we do not assume uniform spacing between the x-values, nor do we need the x-values arranged in a particular order. The x-values must all be distinct, however.
- Through these four data pairs we can pass a cubic. The Lagrangian form for this is

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

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

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

- 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.

- Fit a cubic through the first four points of the preceding table 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)
```

- **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 $P_n(x)$?

- **An algorithm for interpolation from a Lagrange polynomial:**

Given a set of $n + l$ points $[(x_i, f_i), i = 0, \ldots, n]$ and a value for x at which the polynomial is to be evaluated:
Set Sum = 0.
For i=0 To n Step 1 Do
Set P = 1.
For j=0 to n step 1 Do
If (j $\neq$ i) Then
Set $P = P * (x - x_j)/(x_i - x_j)$
End If.
End Do j.
Set Sum = Sum + $P * f_i$
End Do i.
Sum is the interpolated value at 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. This is because every polynomial is everywhere continuous and has continuous derivatives.

1.1.3 Neville's Method

- 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 begins by arranging the given data pairs, (x_i, f_i), so 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 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				

The general formula for computing entries into the table is

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

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

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

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

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

The remaining columns are computed similarly.
The top line of the table represents Lagrangian interpolates at $x = 27.5$ using polynomials of degree equal to the second subscript of the $P's$.

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

1.2 Divided Differences

- There are two disadvantages to using the Lagrangian polynomial or Neville's method for interpolation.

  - First, it involves more arithmetic operations than does the divided-difference method we now discuss.
  - Second, and more importantly, if we desire to add or subtract a point from the set used to construct the polynomial, we essentially have to start over in the computations. Both the Lagrangian polynomials and Neville's method also must repeat all of the arithmetic if we must interpolate at a new x-value. The divided-difference method avoids all of this computation.

- Actually, we will not get a polynomial different from that obtained by Lagrange's technique.

- Every n^{th}-degree polynomial that passes through the same $n+1$ points is identical. Only the way that the polynomial is expressed is different.

- The function, $f(x)$, is known at several values for x:

$$\begin{array}{ll} x_0 & f_0 \\ x_1 & f_1 \\ x_2 & f_2 \\ x_3 & f_3 \end{array}$$

- We do not assume that the x's are evenly spaced or even that the values are arranged in any particular order.

- Consider the n^{th}-degree polynomial written as:

  $P_n(x) = a_0 + (x - x_0)a_1 + (x - x_0)(x - x_1)a_2 + (x - x_0)(x - x_1)\ldots(x - x_{x-x_{n-1}})a_n$

  If we chose the a_i so that $P_n(x) = f(x)$ at the $n+1$ known points, then $P_n(x)$ is an interpolating polynomial. The a_i's are readily determined by using what are called the *divided differences of the tabulated values.* A special standard notation for divided differences is

$$f[x_0, x_1] = \frac{f_1 - f_0}{x_1 - x_0}$$

  called the *first divided difference* between x_1 and x_2. And, $f[x_0] = f_0 = f(x_0)$. In general,

$$f[x_s, x_t] = \frac{f_t - f_s}{x_t - x_s}$$

  Second- and higher-order differences are defined in terms of lower-order differences.

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

x_i	f_i	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, x_{i+1}, x_{i+2}, x_{i+3}]$
x_0	f_0	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$	$f[x_0, x_1, x_2, x_3]$
x_1	f_1	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$	$f[x_1, x_2, x_3, x_4]$
x_2	f_2	$f[x_2, x_3]$	$f[x_2, x_3, x_4]$	
x_3	f_3	$f[x_3, x_4]$		

Table 2: Divided-difference table in symbolic form.

x_i	f_i	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, \ldots, x_{i+3}]$	$f[x_i, \ldots, x_{i+4}]$
3.2	22.0	8.400	2.856	-0.528	0.256
2.7	17.8	2.118	2.012	0.0865	
1.0	14.2	6.342	2.263		
4.8	38.3	16.750			
5.6	51.7				

Table 3: Divided-difference table in numerical values.

For n-terms,

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, x_2, \ldots, f_n] - f[x_0, x_1, \ldots, f_{n-1}]}{x_n - x_0}$$

and zero-order difference

$$f[x_s] = f_s$$

- Using the standard notation, a divided-difference table is shown in symbolic form in Table 2. Table 3 shows specific numerical values.

$$\begin{aligned}
x = x_0: &\quad P_n(x_0) = a_0 \\
x = x_1: &\quad P_n(x_1) = a_0 + (x_1 - x_0)a_1 \\
x = x_2: &\quad P_n(x_2) = a_0 + (x_2 - x_0)a_1 + (x_2 - x_0)(x_2 - x_1)a_2 \\
\vdots &\quad \vdots \\
x = x_n: &\quad P_n(x_n) = a_0 + (x_n - x_0)a_1 + (x_n - x_0)(x_n - x_1)a_2 + \ldots + (x_n - x_0)\ldots(x_n - x_{n-1})a_n
\end{aligned}$$

If $P_n(x)$ is to be an interpolating polynomial, it must match the table for all $n + 1$ entries:

$$P_n(x_i) = f_i \; for \; i = 0, 1, 2, \ldots, n.$$

Each $P_n(x_i)$ will equal f_i if $a_i = f[x_0, x_1, \ldots, x_i]$. We then can write:

$$P_n(x) = f[x_0] + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2]$$

$$+(x-x_0)(x-x_1)(x-x_2)f[x_0,\ldots,x_3]$$
$$+(x-x_0)(x-x_1)\ldots(x-x_{n-1})f[x_0,\ldots,x_n]$$

- Write interpolating polynomial of degree-3 that fits the data of Table 3 at all points $x_0 = 3.2$ to $x_3 = 4.8$.

$$P_3(x) = 22.0 + 8.400(x-3.2) + 2.856(x-3.2)(x-2.7)$$

$$-0.528(x-3.2)(x-2.7)(x-1.0)$$

What is the fourth-degree polynomial that fits at all five points? We only have to add one more term to $P_3(x)$

$$P_4(x) = P_(3) + 0.2568(x-3.2)(x-2.7)(x-1.0)(x-4.8)$$

If we compute the interpolated value at $x = 3.0$, we get the same result: $P_3(3.0) = 20.2120$. This is not surprising, because all third-degree polynomials that pass through the same four points are identical. They may look different but they can all be reduced to the same form.

- **An algorithm for constructing n divided-difference table:**

Given a set of $n+1$ points $[(x_i, f_i), i = 0, \ldots, n]$ and a value $x = u$ at which the interpolating polynomial is to be evaluated:
We first find the coefficients of the interpolating polynomial. These are stored in vector *dd*.
For $i = 0$ To n Step 1 Do
Set dd[i]=f[i]
End For i.
For $j = 1$ To n Step 1 Do
Set temp1=dd[j - 1]
For $k = j$ To n Step 1 Do
Set temp2=dd[k].
Set dd[k]=(dd[k] - temp1)/(x[k]-x[k - j]).
temp1= temp2
End For k.
End For j.
Now we compute the value of the polynomial at u. We do this by nested multiplication from the highest term.
Set sum=0
For i=n DownTo 1 Step 1 Do
Set sum=(sum + dd[i])*(u - x[i - 1])
Set sum=sum + dd[0]
End For i.
ddvalue =sum.
ddvalue is the value of the polynomial at $u, P_n(u)$.

- **Divided differences for a polynomial**
  It is of interest to look at the divided differences for $f(x) = P_n(x)$. Suppose that $f(x)$ is the cubic

$$f(x) = 2x^3 - x^2 + x - 1.$$

Here is its divided-difference table:

x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, \ldots, x_{i+3}]$	$f[x_i, \ldots, x_{i+4}]$	$f[x_i, \ldots, x_{i+5}]$
0.30	−0.7360	2.4800	3.0000	2.0000	0.0000	0.0000
1.00	1.0000	3.6800	3.6000	2.0000	0.0000	
0.70	−0.1040	2.2400	5.4000	2.0000		
0.60	−0.3280	8.7200	8.2000			
1.90	11.0080	21.0200				
2.10	15.2120					

Observe that the third divided differences are all the same. (It then follows that all higher divided differences will be zero.)