3: Interpolation and Fitting

- Introduction
- Lagrangian polynomials
- Divided difference
- Interpolating with cubic spline
- Bezier and B-spline curve
- Polynomial approximation of surfaces
- Least square approximation

Introduction

- **Problems**
  - Given values of an unknown function corresponding to certain values of x, what is the behavior of the function?
    - **Interpolation/Extrapolation**
      - Do linear interpolation?
      - To approximate other values of the function
    - **To estimate the integral of the function and its derivative**
  - Historically a most important task, began with the early study of astronomy

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

Introduction

- **Why do we study interpolation?**
  - **Interpolation methods are the basis for many methods in numerical differentiation and integration, and ODE/PDE**
  - **The methods demonstrate some important theory about polynomials and accuracy problem**
  - **Interpolation with polynomials is important for drawing smooth curves**
  - **History itself may hold a special fascination for some**
    - **There is a rich history behind interpolation.**
    - **It really began with the early studies of astronomy**

Interpolating polynomials

- **Linear interpolation assumes that the unknown function was linear between two points**
  - **Not good if the data are far from linear**
- **Better ways**
  - **Find a polynomial that fits a selected set of points of (x, f(x))**
  - **Do the approximation**

Interpolating polynomials

- **Why polynomials?**
  - **Weierstrass approximation theorem:**

If $f(x)$ is continuous on a finite interval $[a,b]$, there exists a polynomial $P_n(x)$ of degree n such that

$$\left| f(x) - P_n(x) \right| < \text{ERROR}$$

throughout the interval $[a,b]$, for any given $\text{ERROR} > 0$.

Interpolating polynomials

- **Problems with interpolating polynomials when data are not smooth**
  - There are local irregularities
  - Fitting the data requires polynomials of high degree
    - Fitting to the irregularities, but deviate widely at other regions where the function is smooth
    - Oscillating problems
- **Piecewise approximation with different polynomials -> continuity problems**
- **Piecewise spline approximation**
  - Resolve the continuity problems

Interpolating polynomials

- **Study of piecewise spline leads to Bezier and B-spline curves**
  - Do not interpolate the data
  - Are very useful in sketching or designing smooth curves.
- **For data that are not exact**
  - Comes from experimental measurement
  - We don't need to fit the date exactly for such data
  - Least square method finds a polynomial that is more likely to approximate the curve values.

Undetermined coefficients

- **We want to fit a cubic to the data**

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

$$f(x) = ax^3 + bx^2 + cx + d \text{ with unknown coefficients}$$

- **Select 4 points to determine the cubic**

$$f(3.2) = a(3.2)^3 + b(3.2)^2 + c(3.2) + d = 22.0$$

$$f(2.7) = a(2.7)^3 + b(2.7)^2 + c(2.7) + d = 17.8$$

$$f(1.0) = a(1.0)^3 + b(1.0)^2 + c(1.0) + d = 14.2$$

$$f(4.8) = a(4.8)^3 + b(4.8)^2 + c(4.8) + d = 38.3$$

Solution : $a = -0.5275 \quad b = 6.4952 \quad c = -16.1177 \quad d = 24.3499$

At $x = 3.0$, the estimate value is 20.212

Undetermined coefficients

- This is a awkward procedure
  - Needs to re-compute if we want an interpolated polynomial of different degree or also fit at the 5th point
  - Leads to an ill-conditioned system of equations
    - The coefficient values could vary much!

      For example, $x^3=0.001$ for $x=0.1$, $x^3=1000$ for $x=10$

Lagrangian polynomials

- **Perhaps the simplest way to exhibit the existence of a polynomial for interpolating distinct, unevenly spaced data with no particular order**

x	x_0	x_1	x_2	x_3
$f(x)$	f_0	f_1	f_2	f_3

Pass a cubic through these four data pairs

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

Lagrangian polynomials

- Lagrangian polynomial passes through each of the data points
  - Easy to verify
- Interpolating polynomial is ready
- Errors occur since the underlying function is often not a polynomial of the same degree
  - If the degree is the same, the interpolating polynomial is the underlying polynomial
  - We need to have the error of interpolation

Lagrangian polynomials

- **Error of the interpolation:**

$P_n(x)$ will pass exactly through data points, how much is different from $f(x)$? We develop an error function of $P_n(x)$, that has the known property : it is zero at those data pointa :

$$E(x) = f(x) - P_n(x)$$
$$= (x - x_0)(x - x_1)\cdots(x - x_n)g(x)$$

Obviously,

$$f(x) - P_n(x) - E(x) = 0$$
$$f(x) - P_n(x) - (x - x_0)(x - x_1)\cdots(x - x_n)\, g(x) = 0$$

To determine $g(x)$, we construct an auxiliary function

$$W(t) = f(t) - P_n(t) - (t - x_0)(t - x_1)\cdots(t - x_n)g(x),$$

which is actually a function of t and x, but we are only interested in variati ons of t.

Lagrangian polynomials

Now examine the zeros of $W(t)$:

1. At $t = x_0, x_1, \cdots, x_n,\ W(t) = 0.$
2. If $t = x,\ W(t) = 0,$ since
   $f(x) - P_n(x) - E(x) = 0$!!

So there are a total of $n+2$ values of t make $W(t) = 0$!!

[By law of mean value]
If $W(t)$ is continuous and differentiable, there is a zero to its derivative $W'(t)$ between each of the $n+2$ zeros of $W(t)$, a total of $n+1$ zeros.

Similarly, there will be n zeros of $W''(t)$, and likewise $n-1$ zeros of $W'''(t)$, and so on, until we reach $W^{(n+1)}(t)$, which must have at least one zero in the interval that has x_0, x_n, or x as endpoints. Call this value of $t = \xi$.

Lagrangian polynomials

We then have

$$W^{(n+1)}(\xi) = 0$$
$$= \frac{d^{(n+1)}}{dt^{(n+1)}}\left[f(t) - P_n(t) - (t-x_0)(t-x_1)\cdots(t-x_n)g(x)\right]_{t=\xi}$$
$$= f^{(n+1)}(\xi) - 0 - (n+1)!\,g(x).$$

So

$$g(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

where ξ between (x_0, x_n, x).

and the error function is

$$E(x) = (x-x_0)(x-x_1)\cdots(x-x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}$$

where ξ is in the smallest interval that contains $\{x, x_0, x_1, \cdots, x_n\}$.

Lagrangian polynomials

- **Error of the interpolation:**

$$E(x) = (x - x_0)(x - x_1)\cdots(x - x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}$$

where ξ is in the smallest interval that contains $\{x, x_0, x_1, \cdots, x_n\}$.

  - **It is interesting but is not always useful since *f* is often unknown**
  - **But we can conclude that**
    - **If the underlying function is smooth, a low-degree polynomial should work satisfactory**
    - **Extrapolation will have larger errors than interpolation**
    - **The error is smaller if *x* is centered within the x_i**

Lagrangian polynomials

- **A word of caution**
  - Never fit a polynomial of a degree higher than 4 or 5 to a set of points
    - Higher degree polynomial may oscillate and leads to large error for interpolation
  - If you need to fit to a set of more than 6 points, be sure to break up the set into subsets and fit separate polynomials to the subsets
    - A better way to fit a large number of points is to use spline curves

Neville's method

- **Problems of Lagrangian method**
  - Degree of polynomial is not known
  - If the degree is too low, the interpolating polynomial does not give good estimate of f(x)
  - With too high degree, undesirable oscillations may occur
- **Neville's method can overcome this difficulty**
  - It essentially computes the interpolated value with polynomials of successively higher degree, stopping when the successive values are close enough

Neville's method

- The successive approximations are actually computed by linear interpolation from the intermediate values:

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

- Examine the error function for the error term of Lagrange interpolation, the smallest error results when we use **data pairs** where the x_i's are closets to the x-value
  - To reduce error, Neville's method arranges data pairs so that successive values are **in order of closeness of the x_i to x.**

Neville's method

- **How to get the form?**

The Lagrange formula for linear interpolation to get $f(x)$ from two data points, (x_1, f_1) and (x_2, f_2), is

$$P_1(x) = \frac{x - x_2}{x_1 - x_2} f_1 + \frac{x - x_1}{x_2 - x_1} f_2$$

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

1st column of Neville's table:

$$P_{i,1} = \frac{(x - x_i)P_{i+1,0} + (x_{i+1} - x)P_{i,0}}{x_{i+1} - x_i}$$

i	x_i	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				

2nd column of Neville's table:

$$P_{i,2} = \frac{(x - x_i)P_{i+1,1} + (x_{i+2} - x)P_{i,1}}{x_{i+2} - x_i}$$

3rd column:

$$P_{i,3} = \frac{(x - x_i)P_{i+1,2} + (x_{i+3} - x)P_{i,2}}{x_{i+3} - x_i}$$

Example 3.2

- **Given the following data**

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

- **We want to interpolate for x=27.5. We first rearrange the data 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

Example 3.2

- **Neville's table**

i	x_i	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 top line of the table represents Lagrange interpolates at x=27.5 using polynomials of degree equal to the second subscript of the P's (i.e., j of P_{ij})**
  - **Prove this in an exercise!!**
  - **Top line values get better and better until the last, when it diverges**
    - **Correct value for f(27.5)=0.46175**

Divided-Difference method

- **Disadvantages of Lagrangian polynomial or Neville's method**
  - It involves more arithmetic operations than divided-difference method
  - Need to start over in the computations when a point is added or deleted
- **Divided-difference**
  - Note that every *n*th-degree polynomial that passes through the same *n+1* points is identical
  - Divided-difference obtain the same polynomial, but in different form
  - A clever method!!

Divided-Difference method

Given

$(x_0, f_0), (x_1, f_1), (x_2, f_2), (x_3, f_3), (x_4, f_4).$

Suppose that $P_2(x)$ has been derived, with addition of (x_3, y_3), we want to find $P_3(x)$ based on $P_2(x)$.

Consider

$$P_3(x) = P_2(x) + a_3(x - x_0)(x - x_1)(x - x_2)$$

We can easily verify that

$$P_3(x_0) = P_2(x_0) = f_0,\ P_3(x_1) = f_1,\ P_3(x_2) = f_2.$$

If a_3 is chosen such that $P_3(x_3) = f_3$, then $P_3(x)$ intepolates the first 4 points.

- a_i's are readily determined by using the divided differences of tabulated values
  - Without solving an equation for the unknown a_i.

Divided-Difference method

Consider the nth-degree polynomial :

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

If we chose a_i so that $P_n(x) = f(x)$
at the $n+1$ known data points, then
is $P_n(x)$ is an interpolating polynomial
for $x_0, x_1, \ldots, x_n$.

Divided-Difference method-1

- **Divided differences**

First divided difference:

$$f[x_k, x_{k+1}] = \frac{f_{k+1} - f_k}{x_{k+1} - x_k}$$

Note:

$f[x_k, x_{k+1}]$ is the slop of the line segment connecting two points.

Second divided difference:

$$f[x_k, x_{k+1}, x_{k+2}] = \frac{f[x_{k+1}, x_{k+2}] - f[x_k, x_{k+1}]}{x_{k+2} - x_k}$$

Note:

$f[x_k, x_{k+1}, x_{k+2}]$ can be interpreted as (change of slop)/(change in x)

n-th divided difference:

$$f[x_k, \cdots, x_{k+n}] = \frac{f[x_{k+1}, \cdots, x_{k+n}] - f[x_k, \cdots, x_{k+n-1}]}{x_{k+n} - x_k}$$

Divided-Difference method-2

- **Divided difference table**

Table 3.1

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	$f[x_3, x_4]$		
x_4	f_4			

Table 3.2

x_i	f_i	$f[x_i, x_{i+1}]$	$f[x_i, \ldots, x_{i+2}]$	$f[x_i, \ldots, x_{i+3}]$	$f[x_i, \ldots, x_{i+4}]$
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				

Divided-Difference method

- **The a_i's are given by these divided differences. How?**

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

Set $x = x_0, x_1, x_2, \ldots, x_n$, we have

$$x = x_0 : P_n(x_0) = a_0$$

$$x = x_1 : P_n(x_1) = a_0 + (x_1 - x_0)a_1$$

$$x = x_2 : P_n(x_2) = a_0 + (x_2 - x_0)a_1 + (x_2 - x_0)(x_2 - x_1)a_2$$

$$\vdots$$

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

If $P_n(x)$ is an interpolating polynomial, then

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

We get a triangular system, and each a_i can be computed in turn.

Divided-Difference method

If $P_n(x)$ is an interpolating polynomial, then

$$P_n(x_i) = f_i, \text{ for } i = 0,1,\ldots,n.$$

We then have

$$a_0 = f_0 = f[x_0]$$

$$a_1 = \frac{f_1 - f_0}{x_1 - x_0} = f[x_0, x_1]$$

$$a_2 = \frac{f_2 - f_0 - (x_2 - x_0)\dfrac{f_1 - f_0}{x_1 - x_0}}{(x_2 - x_0)(x_2 - x_1)}$$

$$= \frac{f_2 - f_1 + f_1 - f_0 - (x_2 - x_0)\dfrac{f_1 - f_0}{x_1 - x_0}}{(x_2 - x_0)(x_2 - x_1)}$$

$$= \frac{(f_2 - f_1) + (f_1 - f_0)\left(1 - \dfrac{x_2 - x_0}{x_1 - x_0}\right)}{(x_2 - x_0)(x_2 - x_1)}$$

$$= \frac{(f_2 - f_1) - (f_1 - f_0)\left(\dfrac{x_2 - x_1}{x_1 - x_0}\right)}{(x_2 - x_0)(x_2 - x_1)}$$

Divided-Difference method

$$a_2 = \frac{(f_2 - f_1) - (f_1 - f_0)\left(\dfrac{x_2 - x_1}{x_1 - x_0}\right)}{(x_2 - x_0)(x_2 - x_1)}$$

$$= \frac{\dfrac{f_2 - f_1}{x_2 - x_1} - \dfrac{f_1 - f_0}{x_1 - x_0}}{x_2 - x_0}$$

$$= f[x_0, x_1, x_2]$$

Similarily ,

$$a_i = f[x_0, x_1, \ldots, x_i]$$

We then have

Show this is $P_{n-1}(x)$

$$\begin{aligned}
&P_n(x) \\
&= f[x_0] + (x - x_0) f[x_0, x_1] \\
&\quad + (x - x_0)(x - x_1) f[x_0, x_1, x_2] \\
&\quad + \cdots \\
&\quad + (x - x_0)\cdots(x - x_{n-1}) f[x_0, x_1, \ldots, x_n] \\
&= P_{n-1}(x) + \\
&\quad (x - x_0)\cdots(x - x_{n-1}) f[x_0, x_1, \ldots, x_n]
\end{aligned}$$

Divided-Difference method

- **General form**

Given point $x_0, x_1, \cdots, x_n$.

Once $P_{n-1}(x)$ is known,

$P_n(x)$ can be written as

$$P_n(x) = P_{n-1}(x) + a_n(x - x_0)\cdots(x - x_{n-1}),$$

where a_n is obtained by

$$a_n = \frac{f_n - P_{n-1}(x_n)}{(x_n - x_0)\cdots(x_n - x_{n-1})} = f[x_0, x_1, \cdots, x_n]$$

With this, we can easily show that

$P_n(x_i) = f_i$, for $i = 0, 1, \cdots, n$.

Divided-Difference method

- **Theorem**

The interpolating polynomial for $n+1$ points at $x_0, x_1, \cdots, x_n$ satisfies

$$P_n(x) = f[x_0, \cdots, x_n]\, x^n + \text{lower-degree terms.}$$

Comparing to divided difference method,

$$P_n(x) = P_{n-1}(x) + a_n(x - x_0)\cdots(x - x_{n-1})$$

We can easily verify that

$$a_n = f[x_0, \cdots, x_n]$$

Divided-Difference method Example 3.3

Find the interpolating polynomial of degree 3 that fits data in Table 3.2:

$$P_{0,3}(x) = \underline{22.0} + \underline{8.4}\,(x-3.2) + \underline{2.856}\,(x-3.2)(x-2.7) \underline{-0.528}\,(x-3.2)(x-2.7)(x-1.0)$$

Adding x_4, we have

$$P_{0,4}(x) = P_{0,3}(x) + \underline{0.256}\,(x-3.2)(x-2.7)(x-1.0)(x-4.8)$$

Table 3.2

x_i	f_i	$f[x_i, x_{i+1}]$	$f[x_i, \ldots, x_{i+2}]$	$f[x_i, \ldots, x_{i+3}]$	$f[x_i, \ldots, x_{i+4}]$
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				

Part I: Summary -1

- Interpolation/Fitting
  - Given a set of n+1 points (x, f(x)) for the unknown f(x), find a function that fits all points
- Polynomial interpolation
  - Degree n polynomial for n+1 points
  - Undetermined coefficients
    - Solving a linear system of n+1 equations
    - Often an ill-conditioned problem
  - Lagrangian polynomial
    - Polynomial involved n+1 terms
    - Need to rewrite the form once data points added or deleted
    - Appropriate degree of the polynomial is not known

Part I: Summary -2

- **Polynomial interpolation**
  - Degree n polynomial for n+1 points
  - Undetermined coefficients
  - Lagrangian polynomial
  - Neville's method (computing interpolated value)
    - Form a degree n polynomial by linear interpolation from two degree n-1 interpolating polynomial

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

i	x_i	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				

- Form a table
  - Top line represents the Lagrangian interpolates at x'
  - Stop when the successive values are close together

Part I: Summary -3

- **Polynomial interpolation**
  - Degree n polynomial for n+1 points
  - Undetermined coefficients
  - Lagrangian polynomial
  - Neville's method (computing interpolated value)
  - Divided difference

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

- **Form a table**

Table 3.1

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	$f[x_3, x_4]$		
x_4	f_4			

Part I: Summary -4

- **Polynomial interpolation**
  - Degree n polynomial for n+1 points
  - Undetermined coefficients
  - Lagrangian polynomial
  - Neville's method (computing interpolated value)
  - Divided difference
  - Above methods produce identical polynomial
  - Oscillation occurs for high degree polynomial
  - Error of polynomial interpolation
    - *f* is unknown!
    - Approximate error
      - Next-term rule

$$E(x) = (x-x_0)(x-x_1)\cdots(x-x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}$$

where ξ is in the smallest interval that contains $\{x, x_0, x_1, \cdots, x_n\}$.

DD for a polynomial

- **Suppose the underlying function is the cubic polynomial**

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

- **The divided difference table:**

x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i \ldots x_{i+2}]$	$f[x_i \ldots x_{i+3}]$	$f[x_i \ldots x_{i+4}]$	$f[x_i \ldots x_{i+5}]$
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					

DD for polynomials

- **Observations**
  - **The third divided differences are all the same ($2.0=a_3$)**
  - **3-rd derivative of f(x) is also a constant (= 3! * 2 =12 for this example)**
  - **For an nth-degree polynomial, $P_n(x)$, whose highest-power term has the coefficient a_n,**
    - **the n-th divided difference will always be equal to a_n**
    - **n-th derivative of $P_n(x)$ (or f(x))**

      = a_n n!

      = n-th divided difference * n!
    - **The relationship between divided difference and derivatives will be exploited in Chap 5**

DD for polynomials

- **All n-th divided differences = a_n iff all points used to get these DD lie on the curve of an n-th degree polynomial having leading term $a_n x^n$**

If all n-th DD formed from $n+1$ consecutive points are equal to a_n, then all higher DD will be 0.
It then follows that the interpolating polynomial for all points has 0 as its coefficient of x^j term for $j > n$ and hence has degree n and has leading coefficient a_n.

Conversely, if these $n+1$ points lie on the curve of an nth-degree polynomial $p(x)$ having leading term $a_n x^n$, then $P_n(x) = p(x)$ by uniqueness; hence n-th DD $= a_n$.

Identical polynomials

- The interpolating polynomials obtained by the Lagrangian method and divided difference look different but they are really identical
- All polynomials of degree n that match at n+1 points are identical
  - Conceptually, n+1 data pairs are exactly enough to determine the n+1 coefficients, so any resulting polynomial is the same is intuitively true
- Formally, proved by contradiction

Identical polynomials

Suppose $P_n(x)$ and $Q_n(x)$ are two different polynomials of degree n that agree at $n+1$ distinct points. Consider

$$D(x) = P_n(x) - Q_n(x),$$

where $D(x)$ is a polynomial of degree at most n.

$D(x) = 0$ at all $n+1$ of these x-values; that is, $D(x)$ is of degree at most n, but has $n+1$ distinct zeros. This is impossible unless $D(x)$ is identical to zero. Hence $P_n(x) = Q_n(x)$.

- **A most important consequence of this uniqueness property**
  - **Their error terms are also identical**
  - **So we only need to derive the error term from one form of interpolating polynomial**

Error of interpolation from divided difference

- **Same as that for the equivalent Lagrangian interpolation (since all polynomials of degree n that match at n+1 points are identical)**

$$E(x) = (x - x_0)(x - x_1)\cdots(x - x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}$$

where ξ is in the smallest interval that contains $\{x, x_0, x_1, \cdots, x_n\}$.

- **Problem: the derivation of f(x) is unknown**
  - **If f(x) is almost the same as some polynomial of degree n, interpolating with an n-th degree polynomial should be nearly exact**
    - **(n+1)-th derivative of f(x) will be nearly 0, and the error of the nth degree interpolating polynomial will be very small**

Error estimation

- **What if we use a lower-degree polynomial? The error should be larger**
- **If f(x) is a known function, we can use the error term to bound the error**

Here is a divided difference table for $f(x) = x^2 e^{-x/2}$

x_i	$f(x_i)$	$f_i^{[1]}$	$f_i^{[2]}$	$f_i^{[3]}$	$f_i^{[4]}$
1.10	0.6981	0.8593	−0.1755	0.0032	0.0027
2.00	1.4715	0.4381	−0.1631	0.0191	
3.50	2.1287	−0.0511	−0.0657		
5.00	2.0521	−0.2877			
7.10	1.4480				

Find the error of intepolates for $f(1.75)$ using polynomials of degrees 1, 2, 3.

Error estimation

Table 3.3 Errors of interpolation for $f(1.75)$

Degree	Interpolated value	Actual error	$f^{(n+1)}$ maximum	$f^{(n+1)}$ minimum	Upper bound	Lower bound
1	1.25668	0.01996	−0.3679	0.0594	0.0299	−0.00483
2	1.28520	−0.00856	−0.8661	0.1249	0.0059	−0.0408
3	1.28611	−0.00947	1.1398	−0.0359	0.0014	−0.0439

The error formula $E(x)$ does bracket the actual error. The use of a cubic polynomial does not improve the accuracy, because we do not have the x-value well centered within the tabulated values; also the value of the derivative is not decreasing.

Error estimation when f(x) is unknown: Next-term rule

- **Often f(x) is unknown, but there is a way to estimate the error**

nth-order divided difference $f[x_0, x_1, \ldots, x_n]$ is itself an approximation for $f^{(n)}(x)/n!$.

This means that the error of the interpolation is given approximately by the value of the next term that would be added.

Next term rule :

$$E_n(x) = (x - x_0)(x - x_1)\cdots(x - x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}$$

$$= \text{(approximately) the value of the next term would be added to } P_n(x).$$

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

Error estimation

Table 3.3 Errors of interpolation for $f(1.75)$

Degree	Interpolated value	Actual error	$f^{(n+1)}$ maximum	$f^{(n+1)}$ minimum	Upper bound	Lower bound
1	1.25668	0.01996	−0.3679	0.0594	0.0299	−0.00483
2	1.28520	−0.00856	−0.8661	0.1249	0.0059	−0.0408
3	1.28611	−0.00947	1.1398	−0.0359	0.0014	−0.0439

Degree	Exact-error	Next-term-approximation
1	0.01996	0.02852
2	0.00856	0.00091
3	-0.00947	-0.00249

Example : $a_3 = 0.00091$

$$0.00091 = 0.0032\,(1.75-1.10)(1.75-2.00)(1.75-3.50)$$

Interpolation near the end of a table

- **Interpolations using DD do not work well at end of the table**

Newton forward formula :

$$P_n(x) = P_{n-1}(x) + (x - x_0)\cdots(x - x_{n-1})\, f[x_0, \cdots, x_n]$$

Table 3.4(a) Conventional divided-difference table

x_0	f_0	$f_0^{[1]}$	$f_0^{[2]}$	$f_0^{[3]}$	$f_0^{[4]}$
x_1	f_1	$f_1^{[1]}$	$f_1^{[2]}$	$f_1^{[3]}$	
x_2	f_2	$f_2^{[1]}$	$f_2^{[2]}$		
x_3	f_3	$f_3^{[1]}$			
x_4	f_4				

Table 3.4(b) Divided-difference table indexed upwardly

x_4	f_4				
x_3	f_3	$f_3^{[1]}$			
x_2	f_2	$f_2^{[1]}$	$f_2^{[2]}$		
x_1	f_1	$f_1^{[1]}$	$f_1^{[2]}$	$f_1^{[3]}$	
x_0	f_0	$f_0^{[1]}$	$f_0^{[2]}$	$f_0^{[3]}$	$f_0^{[4]}$

Example

Forward divided difference formula : (Table 3.2)

$$P_{0,3}(x) = \underline{22.0} + \underline{8.4}(x-3.2)$$

$$+\underline{2.856}(x-3.2)(x-2.7)$$

$$\underline{-0.528}(x-3.2)(x-2.7)(x-1.0)$$

$$P_{0,4}(x) = P_{0,3}(x) +$$

$$\underline{0.256}(x-3.2)(x-2.7)(x-1.0)(x-4.8)$$

Backward divided difference formula :

$$P_{0,4}(x) = P_{1,4}(x) +$$

$$\underline{0.256}(x-2.7)(x-1.0)(x-4.8)(x-5.6)$$

Table 3.2

x_i	f_i	$f[x_i, x_{i+1}]$	$f[x_i, \ldots, x_{i+2}]$	$f[x_i, \ldots, x_{i+3}]$	$f[x_i, \ldots, x_{i+4}]$
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				

Check the error difference for $P_{0,4}(5.2)$

Evenly spaced data

- **For evenly space data, getting an interpolating polynomial is considerably simplified**
- **Instead of using divided differences, "ordinary difference" is used**

Given $(x_i, f_i), i = 0, \cdots N.$

First - order difference :

$$\Delta f_i = f_{i+1} - f_i, i = 0, \cdots, N-1.$$

Second - order difference :

$$\Delta^2 f_i = \Delta f_{i+1} - \Delta f_i = f_{i+2} - 2 f_{i+1} + f_i, \quad i = 0, \cdots, N-2.$$

nth - order difference :

$$\Delta^n f_i = f_{i+n} - n f_{i+n-1} + \frac{n(n-1)}{2!} f_{i+n-2} - \ldots \pm f_i, \quad i = 0, \cdots, N-n.$$

The coefficien ts are the familiar binomial coefficien ts.

Evenly spaced data

$$P_n(x)$$
$$= f[x_0] + (x-x_0)\frac{f_1-f_0}{x_1-x_0}$$
$$+(x-x_0)(x-x_1)\frac{f_2-2f_1+f_0}{(x_2-x_0)h}$$
$$+\cdots$$
$$+(x-x_0)\cdots(x-x_{n-1})f[x_0,x_1,\ldots,x_n]$$

$$\frac{(x_s-x_0)(x_s-x_1)}{(x_2-x_0)h} = \frac{(hs)[-(h-(x_s-x_0))]}{2h^2}$$
$$= \frac{(hs)[-(h-(hs))]}{2h^2}$$
$$= \frac{(hs)[-h(1-s)]}{2h^2}$$
$$= \frac{h^2 s(s-1)}{2h^2} = \frac{s(s-1)}{2}$$

An interpolated polynomial of degree n, with x evaluated at x_s:

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

where $s = (x_s - x_0)/h$, with $h = \Delta x$.

Note:

1. It is called Newton-Gregory forward polynomial.
2. The coefficients are the binomial coefficients.

The next-term rule:

The error of interpolation is approximated by the next term that would be added.

Example

x	$f(x)$	Δf	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$
0.0	0.000	0.203	0.017	0.024	0.020
0.2	0.203	0.220	0.041	0.044	0.052
0.4	0.423	0.261	0.085	0.096	0.211
0.6	0.684	0.346	0.181	0.307	
0.8	1.030	0.527	0.488		
1.0	1.557	1.015			
1.2	2.572				

Find $f(0.73)$ using a cubic interpolant:

In order to center the x-values around 0.73, we must use $x = 0.4, 0.6, 0.8, 1.0$.

So $x_0 = 0.4$ and $s = (0.73 - 0.4)/0.2 = 1.65$.

$$f(0.73) = 0.423 + 1.65 * 0.261 + \frac{1.65 * 0.65}{2!} 0.085 + \frac{1.65 * 0.65 * -0.35}{3!} 0.096 = 0.893$$

The function is actually $f(x) = \tan(x)$, so the error is $\tan(0.73) - 0.893 = 0.002$.

The next-term rule estimates the error as

$$\frac{1.65 * 0.65 * -0.35 * -1.35}{4!} 0.211 = 0.00445,$$

which is a very good estimate.

Function difference (FD) vs. DD

- **Function difference and divided difference tables are the same when the x-values are evenly spaced**
- **Column 3:**
  - Entries on DD=2, leading coefficient of f(x)
  - Entries on FD=DD*(3!)(h^3)
  - =2*6*0.5^3=1.5
- **Function difference vs. DD**

$$f[x_i,\dots x_n]=\frac{\Delta^n f_i}{n!\,h^n}$$

Table 3.5a Table of function differences for $f(x) = 2x^3$, $h = 0.5$

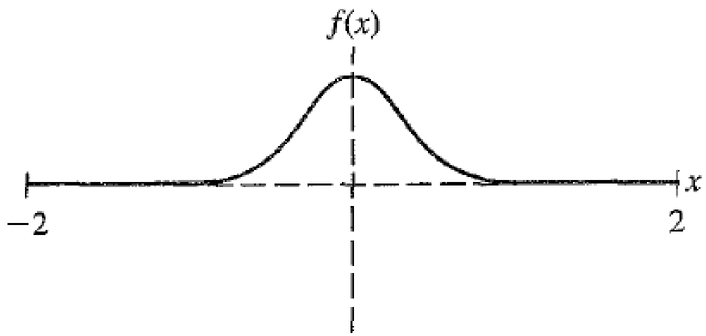
x_i	f_i	Δf_i	$\Delta^2 f_i$	$\Delta^3 f_i$	$\Delta^4 f_i$	$\Delta^5 f_i$
0.00	0.00	0.25	1.50	1.50	0.00	0.00
0.50	0.25	1.75	3.00	1.50	0.00	0.00
1.00	2.00	4.75	4.50	1.50	0.00	
1.50	6.75	9.25	6.00	1.50		
2.00	16.00	15.25	7.50			
2.50	31.25	22.75				
3.00	54.00					

Table 3.5b Table of divided differences for $f(x) = 2x^3$, $h = 0.5$

x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i \dots x_{i+2}]$	$f[x_i \cdots x_{i+3}]$	$f[x_i \dots x_{i+4}]$	$f[x_i \dots x_{i+5}]$
0.00	0.00	0.50	3.00	2.00	0.00	0.00
0.50	0.25	3.50	6.00	2.00	0.00	
1.00	2.00	9.50	9.00	2.00		
1.50	6.75	18.50	12.00	2.00		
2.00	16.00	30.50	15.00			
2.50	31.25	45.50				
3.00	54.00					

Interpolate w/ spline curves

- **Problem of interpolating with a single polynomial: Oscillation**
- **Example:**
  - $f(x)=\cos^{10}(x)$, has a maximum at $x=0$ and is near to x-axis for $|x|>1$



(a) Original function

Interpolate w/ cubic spline

- **Polynomials of degrees 2, 4, 6, and 8 that fits at evenly spaced points over [-2, 2]**

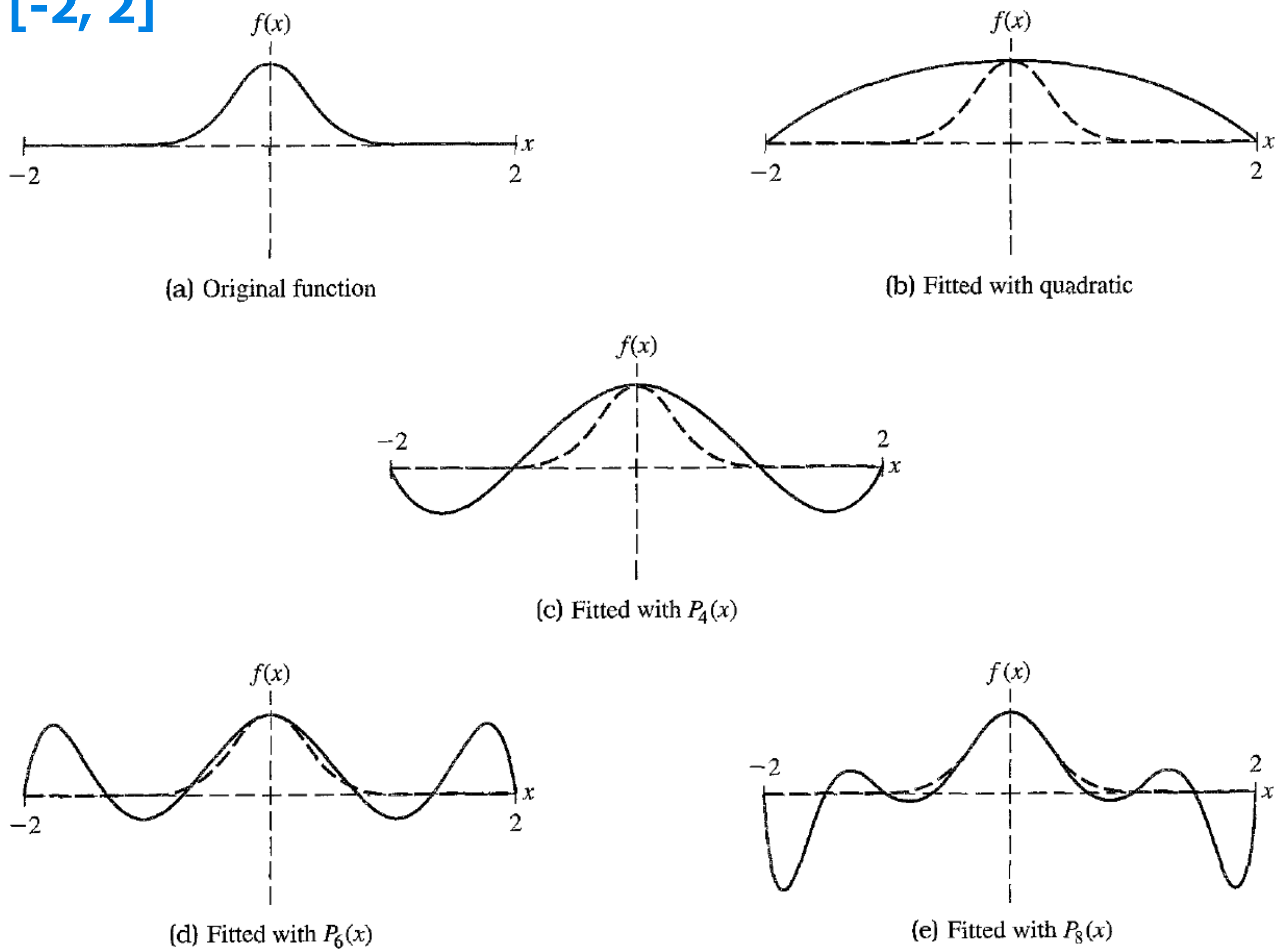


Figure 3.1

Interpolate w/ cubic spline

- **Break up the interval [-2, 2] into subintervals and fit separate polynomials to the function in these subintervals**
  - **A quadratic polynomial for [-0.65, 0.65]**
  - **P(x)=0 outside [-0.65, 0.65]**

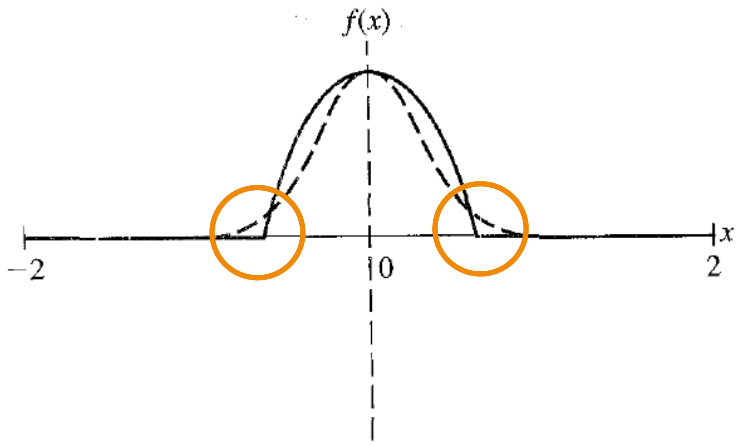
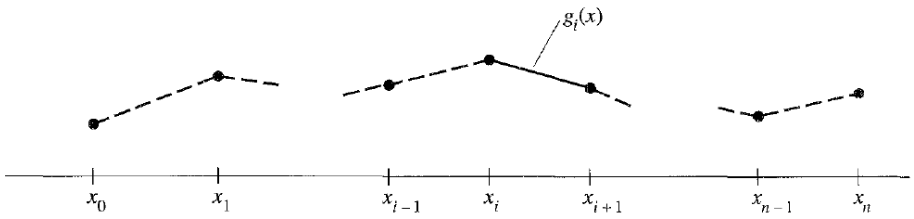


Figure 3.2

- **Problem: discontinuities in the slop between polynomials**

Interpolate w/ cubic spline

- **Both slope and curvatures at points must be continuous.**
  - Linear splines
    - Discontinuous at joins



  - To have property that both slope and curvature everywhere continuous
    - At least degree 3 is required
    - Cubic splines are the most popular
      - We create a succession of cubic splines over successive intervals of the data
      - Each spline must join with its neighboring cubic polynomials at the knots where they join with the same slope and curvature
      - End spline: slope and curvature is not so constrained

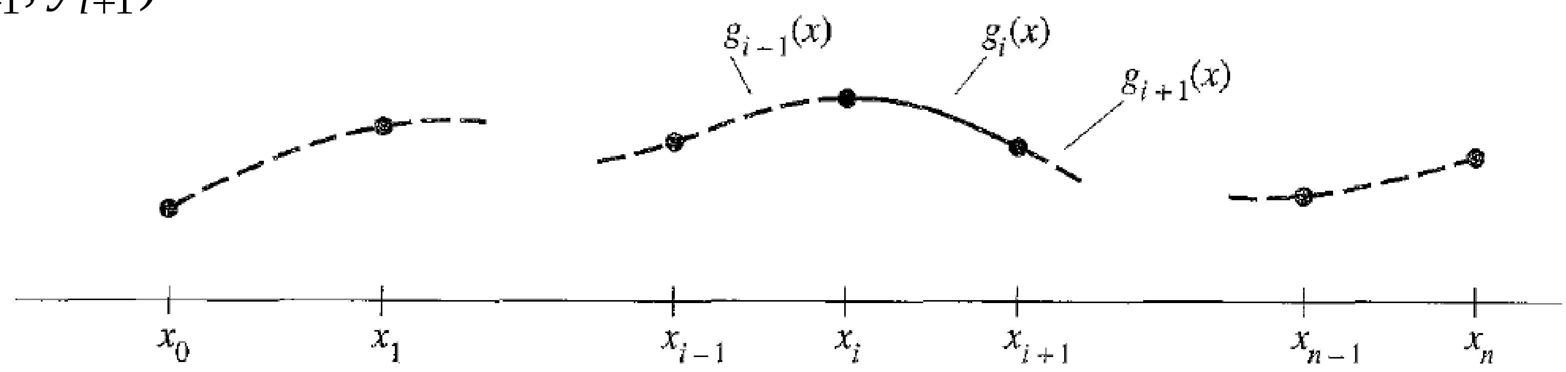
Interpolate w/ cubic spline

- **A piecewise interpolation**

For a set of $n+1$ data points:

$$(x_i, y_i),\ \ i = 0, 1, 2, \cdots, n.$$

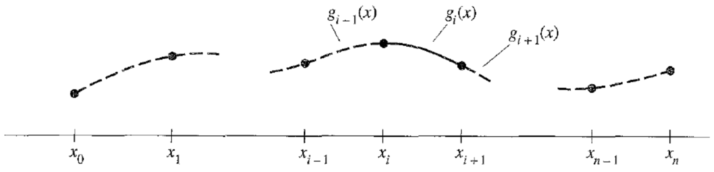
We fit with a set of k-th-degree polynomials $g_i(x)$ between each pair of adjacent points (x_i, y_i) and (x_{i+1}, y_{i+1}).



Interpolate w/ cubic spline

Denote the cubic spline $g_i(x)$ on $[x_i, x_{i+1}]$:

$$g_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i.$$



So the interpolating cubic spline function is

$$g(x) = g_i(x) \text{ on interval } [x_i, x_{i+1}], \quad \text{for } i = 0, 1, \cdots, n-1$$

and meets these conditions :

$$\begin{cases} 1.\ g_i(x_i) = y_i,\ i = 0, 1, \cdots, n-1, \\ \qquad \text{and } g_{n-1}(x_n) = y_n; \\ 2.\ g_i(x_{i+1}) = g_{i+1}(x_{i+1}),\ i = 0, 1, \cdots, n-2; \\ 3.\ g'_i(x_{i+1}) = g'_{i+1}(x_{i+1}),\ i = 0, 1, \cdots, n-2; \\ 4.\ g''_i(x_{i+1}) = g''_{i+1}(x_{i+1}),\ i = 0, 1, \cdots, n-2. \end{cases}$$

Note:

There are $4*n$ unknowns, but $4n-2$ conditions $(3n\text{-}3 + n + 1 = 4n\text{-}2)$.

We will transform the problem to the one with $n+1$ unknowns with $n-1$ conditions.

Interpolate w/ cubic spline

$\boxed{\text{Cond. 1:}}\ g_i(x_i) = y_i, i = 0,1,\cdots,n-1,$

and $g_{n-1}(x_n) = y_n$ implies that

$d_i = y_i, i = 0,1,\cdots,n-1.$

$g_i(x_{i+1}) = g_{i+1}(x_{i+1}), i = 0,1,\cdots,n-2$

implies that

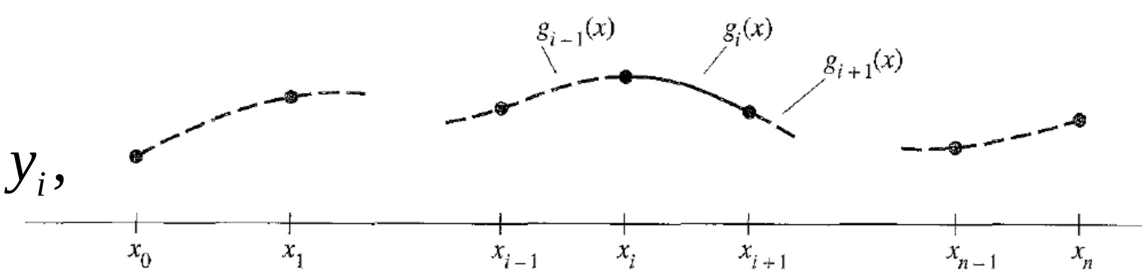
$$\begin{aligned} y_{i+1} &= g_{i+1}(x_{i+1}) = g_i(x_{i+1}) \\ &= a_i(x_{i+1}-x_i)^3 + b_i(x_{i+1}-x_i)^2 \\ &\quad + c_i(x_{i+1}-x_i) + y_i \\ &= a_i h_i^3 + b_i h_i^2 + c_i h_i + y_i, \\ &\qquad i = 0,1,\cdots,n-1. \end{aligned}$$

To relate the slopes and curvatures of the joint splines, we differentiate $g_i(x_i)$:

$$g_i'(x) = 3a_i(x-x_i)^2 + 2b_i(x-x_i) + c_i$$

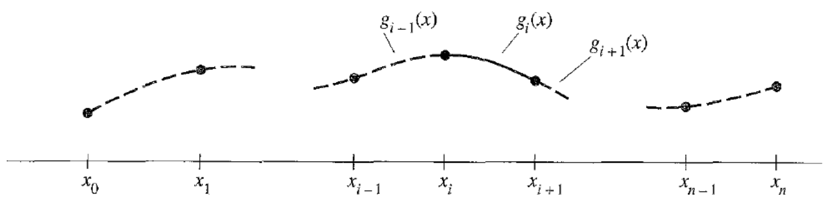
$$g_i''(x) = 6a_i(x-x_i) + 2b_i,$$

for $i = 0,1,\cdots,n-1.$



Interpolate w/ cubic spline

Second derivative of a cubic is linear, so
$g''(x)$ is piecewise linear within $[x_i, x_{i+1}]$.
Let $S_i = g_i''(x_i)$ for $i = 0, 1, \cdots, n-1$, and
$S_n = g_n''(x_n) = g_{n-1}''(x_n)$.



Consider $\boxed{\text{(Cond. 4)}}$
$g''(x) = g_i''(x)$ within $[x_i, x_{i+1}]$, we have

$$S_i = g_i''(x_i) = 6a_i(x_i - x_i) + 2b_i = 2b_i$$

$$S_{i+1} = g_{i+1}''(x_{i+1}) = g_i''(x_{i+1})$$

$$= 6a_i(x_{i+1} - x_i) + 2b_i = 6a_i h_i + 2b_i$$

So

$$b_i = S_i / 2$$

$$a_i = \frac{S_{i+1} - S_i}{6h_i}$$

$\boxed{\text{Cond.2:}}$

$$y_{i+1} = g_{i+1}(x_{i+1}) = g_i(x_{i+1})$$

$$= a_i h_i^3 + b_i h_i^2 + c_i h_i + y_i,$$

$$i = 0, 1, \cdots, n-1.$$

Substitute a_i, b_i, and d_i into $g_i(x)$, we have

$$y_{i+1} = \left(\frac{S_{i+1} - S_i}{6h_i} \right) h_i^3 + \frac{S_i}{2} h_i^2 + c_i h_i + y_i$$

$$\Rightarrow c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{2h_i S_i + h_i S_{i+1}}{6}$$

Interpolate w/ cubic spline

Consider condition of slope continuous at (x_i, y_i). With $x = x_i$, we have

$$y_i' = g_i'(x_i)$$
$$= 3a_i(x_i - x_i)^2 + 2b_i(x_i - x_i) + c_i = c_i.$$

In the previous interval $[x_{i-1}, x_i]$, the slpoe at right end:

$$y_i' = g_{i-1}'(x_i)$$
$$= 3a_{i-1}(x_i - x_{i-1})^2 + 2b_{i-1}(x_i - x_{i-1}) + c_{i-1}$$
$$= 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1}$$

Equating these and substitute for a, b, c, and d, their relationships in terms of S and y, we have

$\boxed{\text{Cond. 3:}}$

$$y_i' = g'(x_i) = c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{2h_i S_i + h_i S_{i+1}}{6}$$

$$y_i' = g_{i-1}'(x_i) = 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1}$$

$$\frac{y_{i+1} - y_i}{h_i} - \frac{2h_i S_i + h_i S_{i+1}}{6}$$
$$= 3\left(\frac{S_i - S_{i-1}}{6h_{i-1}}\right)h_{i-1}^2 + 2\left(\frac{S_{i-1}}{2}\right)h_{i-1}$$
$$+ \left(\frac{y_i - y_{i-1}}{h_{i-1}} - \frac{2h_{i-1}S_{i-1} + h_{i-1}S_i}{6}\right)$$

Interpolate w/ cubic spline

Last equation can be simplified to

$$h_{i-1}S_{i-1} + (2h_{i-1} + 2h_i)S_i + h_i S_{i+1}$$

$$= 6\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right)$$

$$= 6\,(f[x_i, x_{i+1}] - f[x_{i-1}, x_i]),$$

for $i = 1, 2, \cdots, n-1.$

Applying the equation at each internal points, from $i = 1$ to $n-1$, gives $\boxed{n-1}$ equations relating the $\boxed{n+1}$ values of S_i.

We need two more conditions.

Interpolate w/ cubic spline

$n-1$ conditions : for $i = 1,2,\cdots,n-1.$

$$h_{i-1}S_{i-1} + (2h_{i-1} + 2h_i)S_i + h_i S_{i+1}$$
$$= 6(f[x_i, x_{i+1}] - f[x_{i-1}, x_i])$$

can be written in matrix form :

Constrain two unknowns using end conditions.

The matrix is always tridiagonal.

$$\begin{bmatrix} h_0 & 2(h_0+h_1) & h_1 & & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & & \\ & & h_2 & 2(h_2+h_3) & h_3 & & \\ & & & & \ddots & & \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \\ \vdots \\ S_{n-1} \\ S_n \end{bmatrix}$$

$$= 6 \begin{bmatrix} f[x_1, x_2] - f[x_0, x_1] \\ f[x_2, x_3] - f[x_1, x_2] \\ f[x_3, x_4] - f[x_2, x_3] \\ \vdots \\ f[x_{n-1}, x_n] - f[x_{n-2}, x_{n-1}] \end{bmatrix}.$$

Interpolate w/ cubic spline

Initial problem :

$4n$ unknowns :

Four unknown coefficients for each of n splines $g_i(x)$.

$4n-2$ conditions.

Transformed problem :

$n+1$ unknowns : $S_0, S_1, \cdots, S_n$

$n-1$ conditions :

$$h_{i-1}S_{i-1} + (2h_{i-1} + 2h_i)S_i + h_i S_{i+1} = 6(f[x_i, x_{i+1}] - f[x_{i-1}, x_i]),$$
$$i = 1, 2, \cdots, n-1.$$

Users specify two more conditions.

After $S_0, S_1, \cdots, S_n$ are derived, the coefficients of $g_i(x)$ can be computed：

$$a_i = \frac{S_{i+1} - S_i}{6h_i}$$
$$b_i = S_i / 2$$

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{2h_i S_i + h_i S_{i+1}}{6}, \quad d_i = y_i$$

Interpolate w/ cubic spline
End conditions

We can specify conditions pertaining to the end intervals.

To some extent, these end conditions are arbitrary.

Four possible strategies are often used :

1. $S_0 = 0$ and $S_n = 0$ May flatten the curve too much at the ends!!
2. Fix the slopes at x_0 and x_n Probably the best end condition if reasonable slop estimates are available!!
3. $S_0 = S_1$ and $S_n = S_{n-1}$
4. Use linear extrapolation from nearby 2 points

$$S_0 = \frac{(h_0 + h_1)S_1 - h_0 S_2}{h_1}$$

May give too much curvature in the end intervals!!

$$S_n = \frac{(h_{n-2} + h_{n-1})S_{n-1} - h_{n-1} S_{n-2}}{h_{n-2}}$$

Interpolate w/ cubic spline End conditions

1. $S_0 = 0$ and $S_n = 0$:
   Called a "natural spline", makes the end cubics approach linearity at their extremities. May faltten the curve too much at the ends. Matches preciously to the drafting device, and is used frequently.

2. Fix the slopes at x_0 and x_n to specified values.
   If $f'(x_0) = A$ and $f'(x_n) = B$,
   From the equation : $y' = c_i = A$
   At left end :
   $$2h_0 S_0 + h_0 S_1 = 6(f[x_0, x_1] - A).$$
   At right end :
   $$h_{n-1} S_{n-1} + 2h_{n-1} S_n = 6(B - f[x_{n-1}, x_n]).$$
   This is probably the best end condition to use provided reasonable estimate of the derivative are available (estimated from the data point)!!

Interpolate w/ cubic spline
End conditions

3. $S_0 = S_1$ and $S_n = S_{n-1}$.
Equvalent to assuming that the end cubics approach parabolas at their extremities.

4. Take S_0 as a linear extrapolation from S_1 and S_2 :

$$S_0 = \frac{(h_0 + h_1)S_1 - h_0 S_2}{h_1}$$

Take S_n as a linear extrapolation from S_{n-1} and S_{n-2} :

$$S_n = \frac{(h_{n-2} + h_{n-1})S_{n-1} - h_{n-1} S_{n-2}}{h_{n-2}}$$

Only this condition gives cubic spline curves that match exactly to $f(x)$ when $f(x)$ is itself a cubic. But frequently suffers from the other extreme, giving too much curvature in the end intervals.

Interpolate w/ cubic spline

$n-1$ conditions : for $i = 1,2,\cdots,n-1.$

$$h_{i-1}S_{i-1} + (2h_{i-1} + 2h_i)S_i + h_iS_{i+1}$$
$$= 6(f[x_i, x_{i+1}] - f[x_{i-1}, x_i])$$

can be written in matrix form :

If we write the equation of $S_1, S_2, \ldots, S_{n-1}$ [Eq. (3.17)] in matrix form, we get

$$\begin{bmatrix} h_0 & 2(h_0+h_1) & h_1 & & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & & \\ & & h_2 & 2(h_2+h_3) & h_3 & & \\ & & & & \ddots & & \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \\ \vdots \\ S_{n-1} \\ S_n \end{bmatrix}$$
$$= 6 \begin{bmatrix} f[x_1, x_2] - f[x_0, x_1] \\ f[x_2, x_3] - f[x_1, x_2] \\ f[x_3, x_4] - f[x_2, x_3] \\ \vdots \\ f[x_{n-1}, x_n] - f[x_{n-2}, x_{n-1}] \end{bmatrix}.$$

Constrain two unknowns using end conditions.

The matrix is always tridiagonal.

Interpolate w/ cubic spline

Condition 1 $\quad S_0 = 0, S_n = 0$:

$$\begin{bmatrix} 2(h_0 + h_1) & h_1 & & & \\ h_1 & 2(h_1 + h_2) & h_2 & & \\ & h_2 & 2(h_2 + h_3) & h_3 & \\ & & & \ddots & \\ & & & h_{n-2} & 2(h_{n-2} + h_{n-1}) \end{bmatrix}.$$

(n-1)x(n-1)

Condition 2 $\quad f'(x_0) = A$ and $f'(x_n) = B$:

$$\begin{bmatrix} 2h_0 & h_0 & & & \\ h_0 & 2(h_0 + h_1) & h_1 & & \\ & h_1 & 2(h_1 + h_2) & h_2 & \\ & & & \ddots & \\ & & & h_{n-1} & 2h_{n-1} \end{bmatrix}.$$

(n+1)x(n+1)

Interpolate w/ cubic spline

Condition 3 $\quad S_0 = S_1, S_n = S_{n-1}$:

$$\begin{bmatrix} (3h_0 + 2h_1) & h_1 & & & \\ h_1 & 2(h_1 + h_2) & h_2 & & \\ & h_2 & 2(h_2 + h_3) & h_3 & \\ & & & \ddots & \\ & & & h_{n-2} & (2h_{n-2} + 3h_{n-1}) \end{bmatrix}.$$

(n-1)x(n-1)

Condition 4 $\quad S_0$ and S_n are linear extrapolations:

(n+1)x(n+1)

$$\begin{bmatrix} \frac{(h_0 + h_1)(h_0 + 2h_1)}{h_1} & \frac{h_1^2 - h_0^2}{h_1} & & & \\ h_1 & 2(h_1 + h_2) & h_2 & & \\ & h_2 & 2(h_2 + h_3) & h_3 & \\ & & & \ddots & \\ & & & \frac{h_{n-2}^2 - h_{n-1}^2}{h_{n-2}} & \frac{(h_{n-1} + h_{n-1})(h_{n-1} + 2h_{n-2})}{h_{n-2}} \end{bmatrix}.$$

With condition 4, we need to compute

$$S_0 = \frac{(h_0 + h_1)S_1 - h_0 S_2}{h_1}$$

$$S_n = \frac{(h_{n-2} + h_{n-1})S_{n-1} - h_{n-1} S_{n-2}}{h_{n-2}}$$

Interpolate w/ cubic spline

After the S_i are obtained, coefficients a_i, b_i, c_i, d_i for $g_i(x)$ are computed:

$$\begin{cases} a_i = \dfrac{S_{i+1} - S_i}{6h_i} \\[2ex] b_i = \dfrac{S_i}{2} \\[2ex] c_i = \dfrac{y_{i+1} - y_i}{h_i} - \dfrac{2h_i S_i + h_i S_{i+1}}{6} \\[2ex] d_i = y_i \end{cases}$$

Interpolate w/ cubic spline Example 3.5

$$f(x) = 2e^x - x^2$$

Table 3.6

x	$f(x)$
0.0	2.0000
1.0	4.4366
1.5	6.7134
2.25	13.9130

Problem :

Fit the data with a cubic spline

and to interpolate $g(0.66)$ and $g(1.75)$:

Note that :

$h_0 = 1.0,\ h_1 = 0.5,\ h_2 = 0.75$

$f[0,1] = 2.4366,\ f[1,1.5] = 4.5536,$

$f[1.5, 2.25] = 9.5995$

Interpolate w/ cubic spline Example 3.5

Using condition $1\,(S_0 = S_3 = 0)$,

we solve

$$\begin{bmatrix} 3.0 & 0.5 \\ 0.5 & 2.5 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \begin{bmatrix} 12.7020 \\ 30.2754 \end{bmatrix}$$

$$\Rightarrow \begin{cases} S_1 = 2.2920 \\ S_2 = 11.6518 \end{cases}$$

Using the S's, we obtain $g_i(x)$:

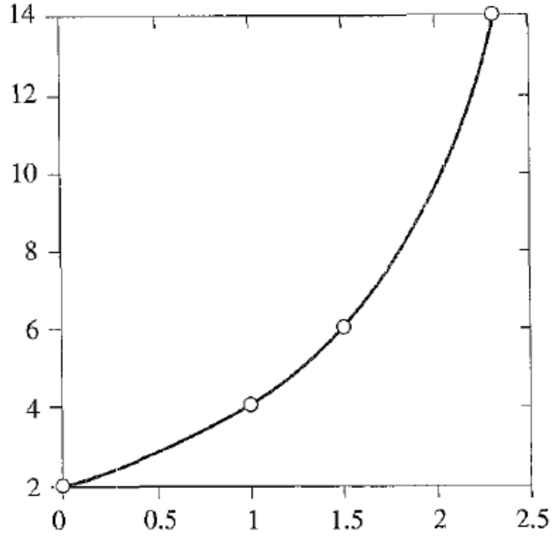


Figure 3.3

i	**Interval**	$g_i(x)$
0	[0.0, 1.0]	$0.3820(x-0)^3 + 0(x-0)^2 + 2.0546(x-0) + 2.0000$
1	[1.0, 1.5]	$3.1199(x-1)^3 + 1.146(x-1)^2 + 3.2005(x-1) + 4.4366$
2	[1.5, 2.25]	$-2.5893(x-1.5)^3 + 5.8259(x-1.5)^2 + 6.6866(x-1.5) + 6.7134$

Fig 3.3: The cubic spline curve

We use g_0 to find g(0.66)=3.4659 (True=3.4340)

We use g_2 to find g(1.75)=8.7087 (True=8.4467)

Interpolate w/ cubic spline Example 3.6

Fit cubic spline to $f(x) = \cos^{10}(x)$

with knots at -2, -1, -0.5, 0, 0.5, 1, 2.

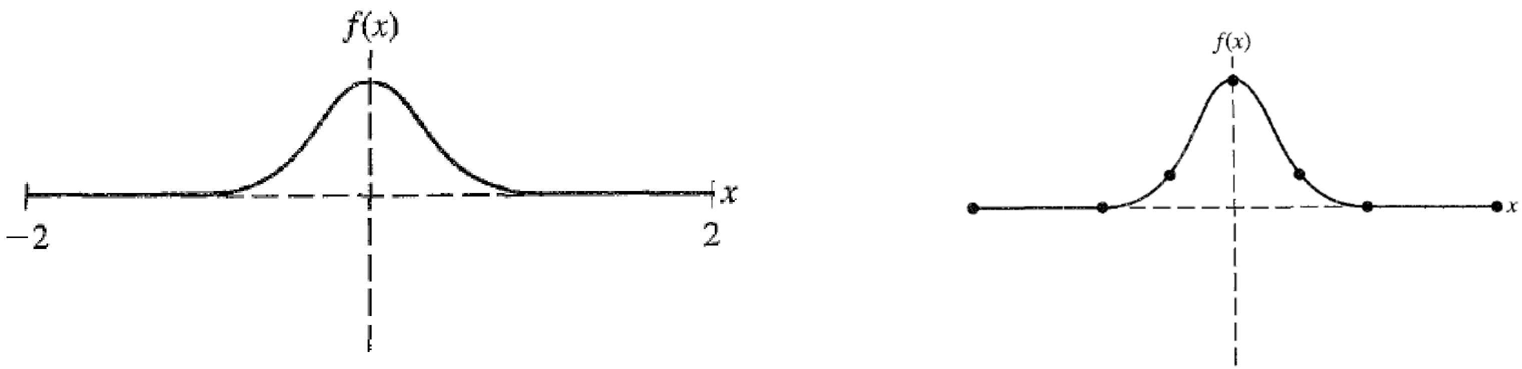


Figure 3.5

Interpolate w/ cubic spline Example 3.6

Table 3.9 A cubic spline fitted to the function $f(x) = \cos^{10}(x)$, end condition 1

x-value	Spline value	$f(x)$	Error
−2.00	0.0002	0.0002	0.0000
−1.75	−0.0046	0.0000	0.0046
−1.50	−0.0073	0.0000	0.0073
−1.25	−0.0058	0.0000	0.0058
−1.00	0.0021	0.0021	−0.0000
−0.75	0.0467	0.0440	−0.0027
−0.50	0.2709	0.2709	−0.0000
−0.25	0.7283	0.7292	0.0009
0.00	1.0000	1.0000	0.0000
0.25	0.7283	0.7292	0.0009
0.50	0.2709	0.2709	−0.0000
0.75	0.0467	0.0440	−0.0027
1.00	0.0021	0.0021	−0.0000
1.25	−0.0058	0.0000	0.0058
1.50	−0.0073	0.0000	0.0073
1.75	−0.0046	0.0000	0.0046
2.00	0.0002	0.0002	−0.0000

Bezier and B-spline curves

- **Widely used in computer graphics and CAD**
- **Not really interpolating splines, they don't pass all points**
- **Good features:**
  - **Convex-Hull property**
  - **Local effect of moving a point**
    - **The points are called *control points***
- **They are parametric curves:**

$$P(u) = \begin{pmatrix} P_x(u) \\ P_y(u) \end{pmatrix}$$

where u is called parameter, which normally ranges from 0 to 1.

Bezier curve

- **Named after the French engineer P. Bezier, who developed Bezier curve and surface in early 1960s.**

Given $n+1$ control points

$$p_i = (x_{i,} y_i), i = 0, \cdots, n.$$

The n-th degree Bezier curve

defined by $n+1$ points is

$$P(u) = \sum_{i=0}^{n} \binom{n}{i} (1-u)^{n-i} u^i p_i$$

where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

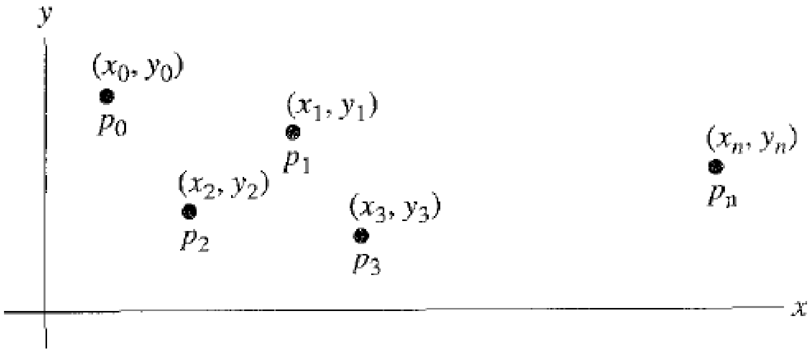


Figure 3.6

Bezier curve Example

For $n = 2$

$$P(u) = \begin{bmatrix} P_x(u) \\ P_y(u) \end{bmatrix} = (1-u)^2 p_0 + 2(1-u)u\ p_1 + u^2 p_2$$

$$= \begin{bmatrix} (1-u)^2 x_0 + 2(1-u)ux_1 + u^2 x_2 \\ (1-u)^2 y_0 + 2(1-u)uy_1 + u^2 y_2 \end{bmatrix}$$

Note:

1. The Bezier curve passes through two end points.
2. Bezier equations are weighted sums of three polynomials in u, where the weighting factors are coordinates of the three points.

Cubic Bezier curve

For $n = 3$

$$P(u) = \begin{bmatrix} P_x(u) \\ P_y(u) \end{bmatrix} = (1-u)^3 p_0 + 3(1-u)^2 u\ p_1 + 3(1-u)u^2 p_2 + u^3 p_3$$

$$= \begin{bmatrix} (1-u)^3 x_0 + 3(1-u)^2 u x_1 + 3(1-u)u^2 x_2 + u^3 x_3 \\ (1-u)^3 y_0 + 3(1-u)^2 u y_1 + 3(1-u)u^2 y_2 + u^3 y_3 \end{bmatrix}$$

Properties:

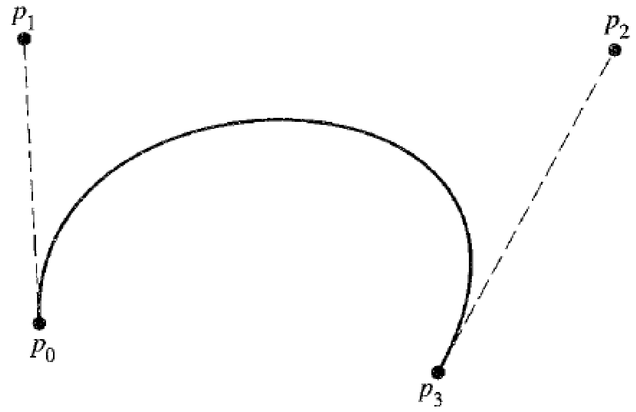
1. $P(0) = p_0, P(1) = p_3$.
2. Slope of the curve at $u = 0$ is

   $dy / dx = (x_1 - x_0) / (y_1 - y_0)$

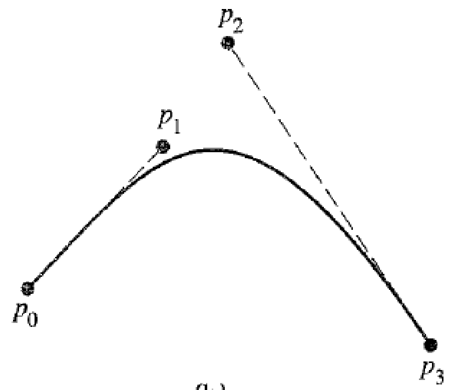
   which is the slop of the secant line between p_0 and p_1.

   Similar for the other end.
3. Convex hull property: the whole curve is contained inside the convex hull determined by the four points.
4. Moving the control point changes the shape of the whole curve.

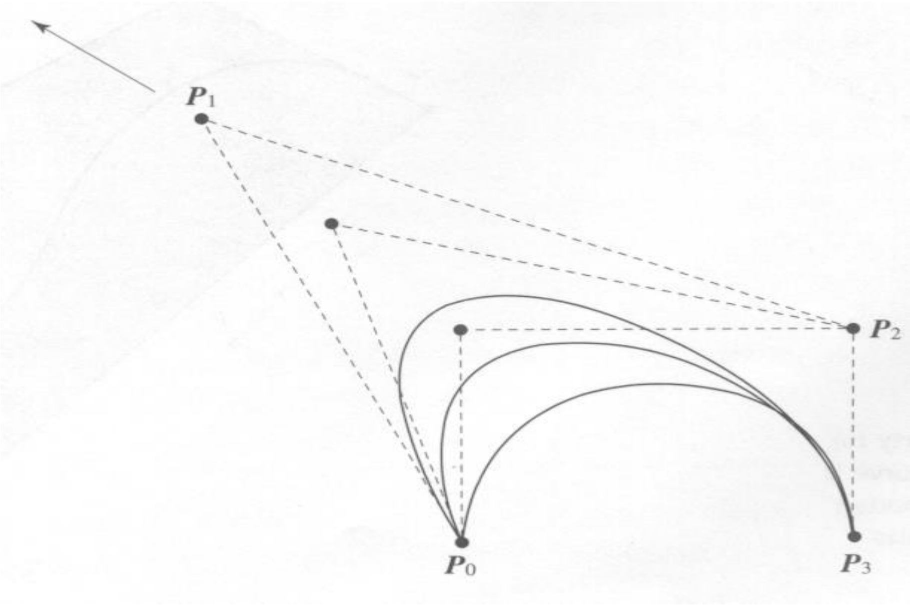
Cubic Bezier curve

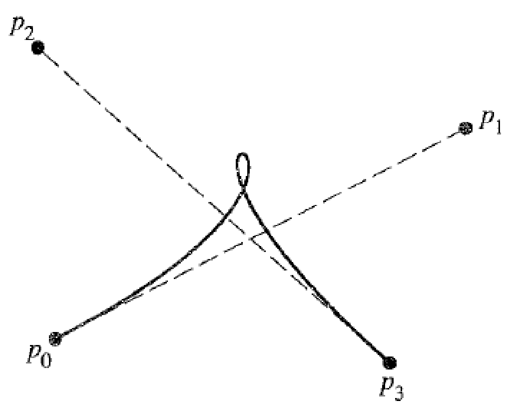


(a)



(b)





(c)

Joining cubic Bezier curves

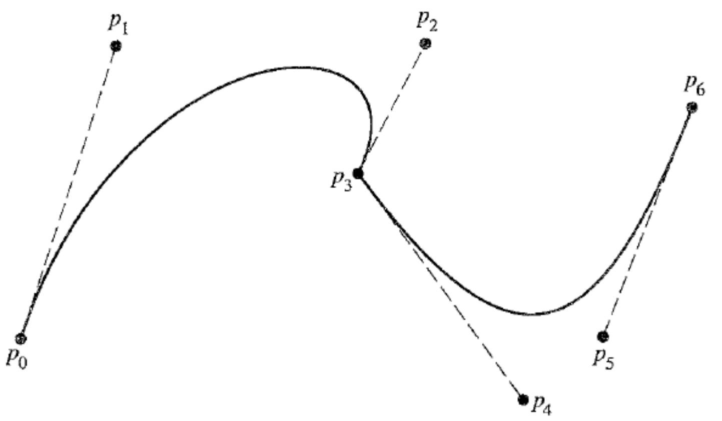
Two cubic Bezier curves defined by

p_0, p_1, p_2, p_3 and $p_3, p_4, p_5, p_6,$

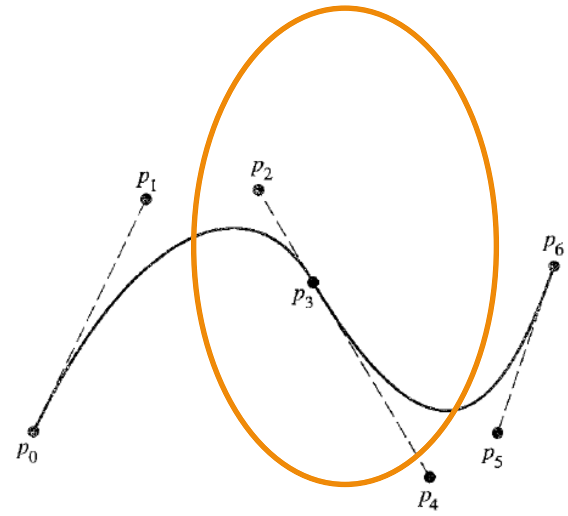
respectively.

To smoothly join these two curve at p_3,

p_2, p_3 and p_4 must be collinear.



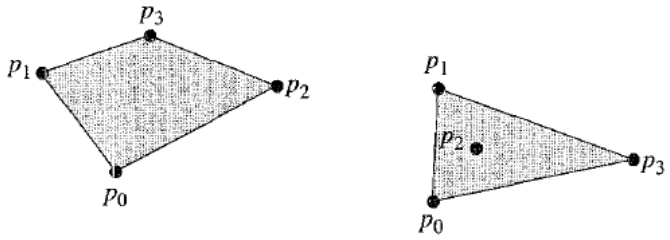
(d)



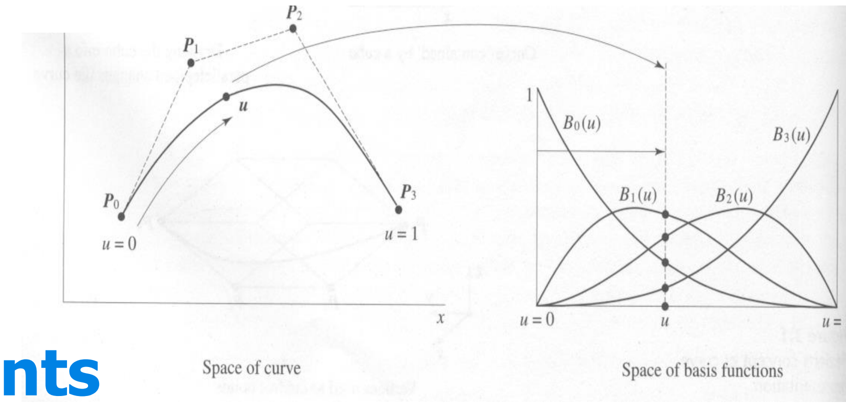
(e)

Bezier curve
Convex Hull property

- **Convex hull of a set of points is the smallest convex set that contains the points**
  - A set C is convex iff the line segment between any two points in the set lies entirely in set C



- **Convex Hull property**
  - The whole Bezier curve is inside the convex hull defined by the control points



Bezier curve
Matrix form

- **It is often convenient to represent the Bezier curve in matrix form.**
- **For cubic Bezier cubics:**

$$P(u)=\begin{bmatrix} x(u) \\ y(u) \end{bmatrix} = (1-u)^3 p_0 + 3(1-u)^2 u p_1$$

$$+3(1-u)u^2 p_2 + u^3 p_3$$

$$=\begin{bmatrix} (1-u)^3 x_0 + 3(1-u)^2 u x_1 + 3(1-u)u^2 x_2 + u^3 x_3 \\ (1-u)^3 y_0 + 3(1-u)^2 u y_1 + 3(1-u)u^2 y_2 + u^3 y_3 \end{bmatrix}$$

$$P(u)=[u^3, u^2, u, 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$=u^T M p$$

B-spline curves

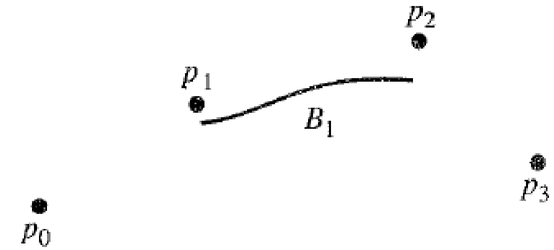
- **Like Bezier curve, but differ in**
  - Do not pass through all points
  - Better convex hull property (smaller convex hull)
  - Slopes at end points have any relation with control points
  - Local control vs. global control in Bezier curves
  - Bezier curve is a special case of B-spline curve

B-spline curves

- **Cubic B-spline resemble the cubic spline discussed previously in Sec. 3.3:**
  - **A separate cubic is derived for each pair of points**
  - **But, B-spline need not pass through any point**

Given points $p_i = (x_i, y_i), i = 0,1,\cdots,n,$ the cubic B-spline for the interval $(p_i, p_{i+1}), i = 1,2,\cdots,n-1,$ is

$$B_i(u) = \sum_{k=-1}^{2} b_k(u)\, p_{i+k},$$



(a)

where $b_{-1}(u) = \dfrac{(1-u)^3}{6},\ b_0(u) = \dfrac{u^3}{2} - u^2 + \dfrac{2}{3}$

$$b_1(u) = -\frac{u^3}{2} + \frac{u^2}{2} + \frac{u}{2} + \frac{1}{6},\ b_2(u) = \frac{u^3}{6},\ 0 \le u \le 1.$$

B-spline curves

Equivalently,

$$B_i(u) = \begin{bmatrix} \dfrac{(1-u)^3}{6} x_{i-1} + (\dfrac{u^3}{2} - u^2 + \dfrac{2}{3}) x_i + (-\dfrac{u^3}{2} + \dfrac{u^2}{2} + \dfrac{u}{2} + \dfrac{1}{6}) x_{i+1} + \dfrac{u^3}{6} x_{i+2} \\ \dfrac{(1-u)^3}{6} y_{i-1} + (\dfrac{u^3}{2} - u^2 + \dfrac{2}{3}) y_i + (-\dfrac{u^3}{2} + \dfrac{u^2}{2} + \dfrac{u}{2} + \dfrac{1}{6}) y_{i+1} + \dfrac{u^3}{6} y_{i+2} \end{bmatrix}$$

B-spline curves

- $b_k(u)$ serves as basis and can be considered weighting factors applied to the 4 points
  - At u=0, weights are 1/6, 2/3, 1/6, 0
  - At u=1, weights are 0, 1/6, 2/3, 1/6
- Local control
  - Moving a point, affects 4 curve segments
    - Ex, Moving P_2 affects B_0, B_1, B_2, B_3

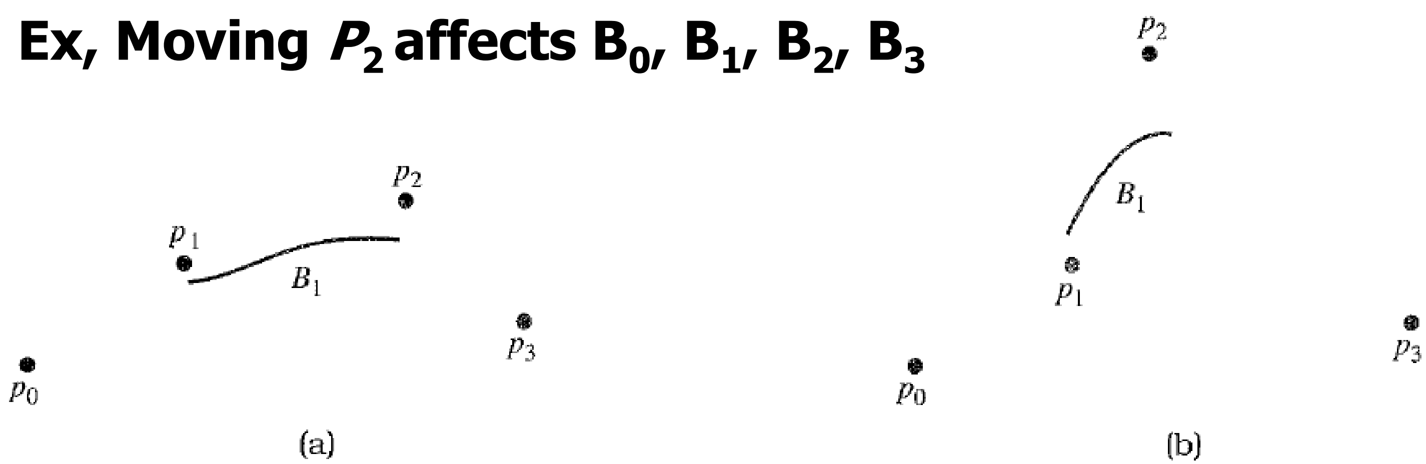


Figure 3.8

B-spline curves

- **A set of 4 points is required to generate only a portion of the B-spline**
- **Globally, $B_i(u)$ and $B_{i+1}(u)$ can be pieced together, sharing 3 points**

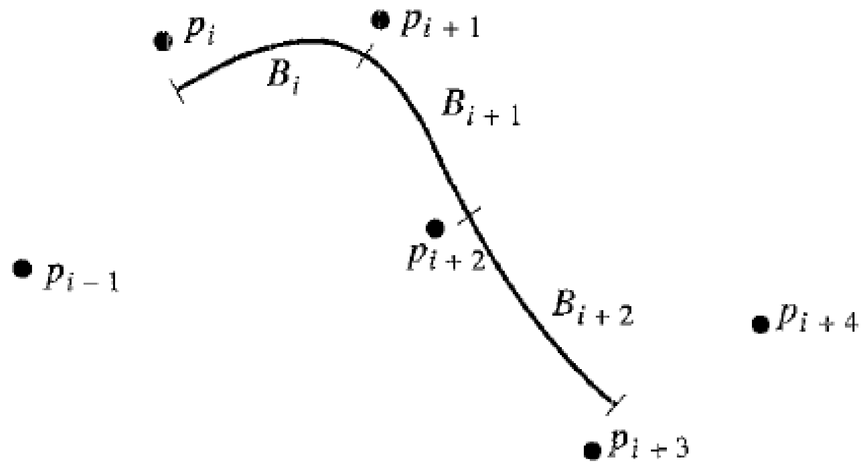


Figure 3.9
Successive B-splines joined together

B-spline curves

- **Given n+1 points $p_0, p_1, \ldots, p_n$, we want to form a piecewise B-spline.**
- **$b_k(u)$ are such that continuity requirement of the first and second derivatives met**

1. $B_i(u)$ and $B_{i+1}(u)$ are pieced together so they agree at their joint in three ways:

   a. $B_i(1) = B_{i+1}(0) = \dfrac{p_i + 4p_{i+1} + p_{i+2}}{6}$

   b. $B_i'(1) = B_{i+1}'(0) = \dfrac{-p_i + p_{i+2}}{2}$

   c. $B_i''(1) = B_i''(0) = p_i - 2p_{i+1} + p_{i+2}$

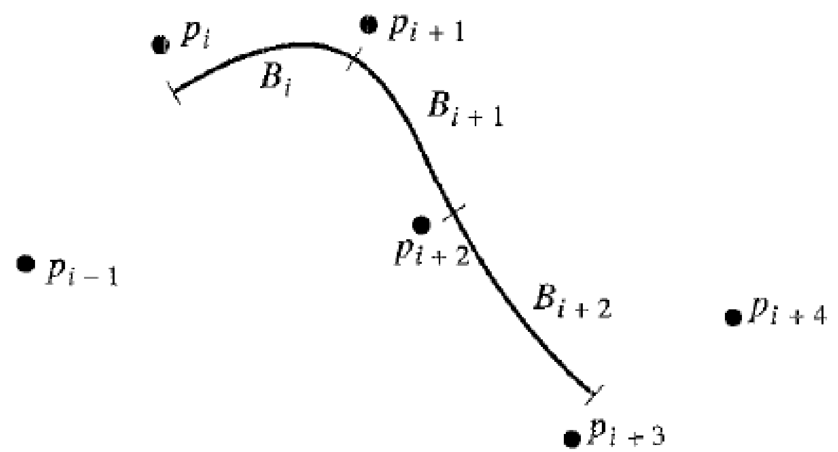


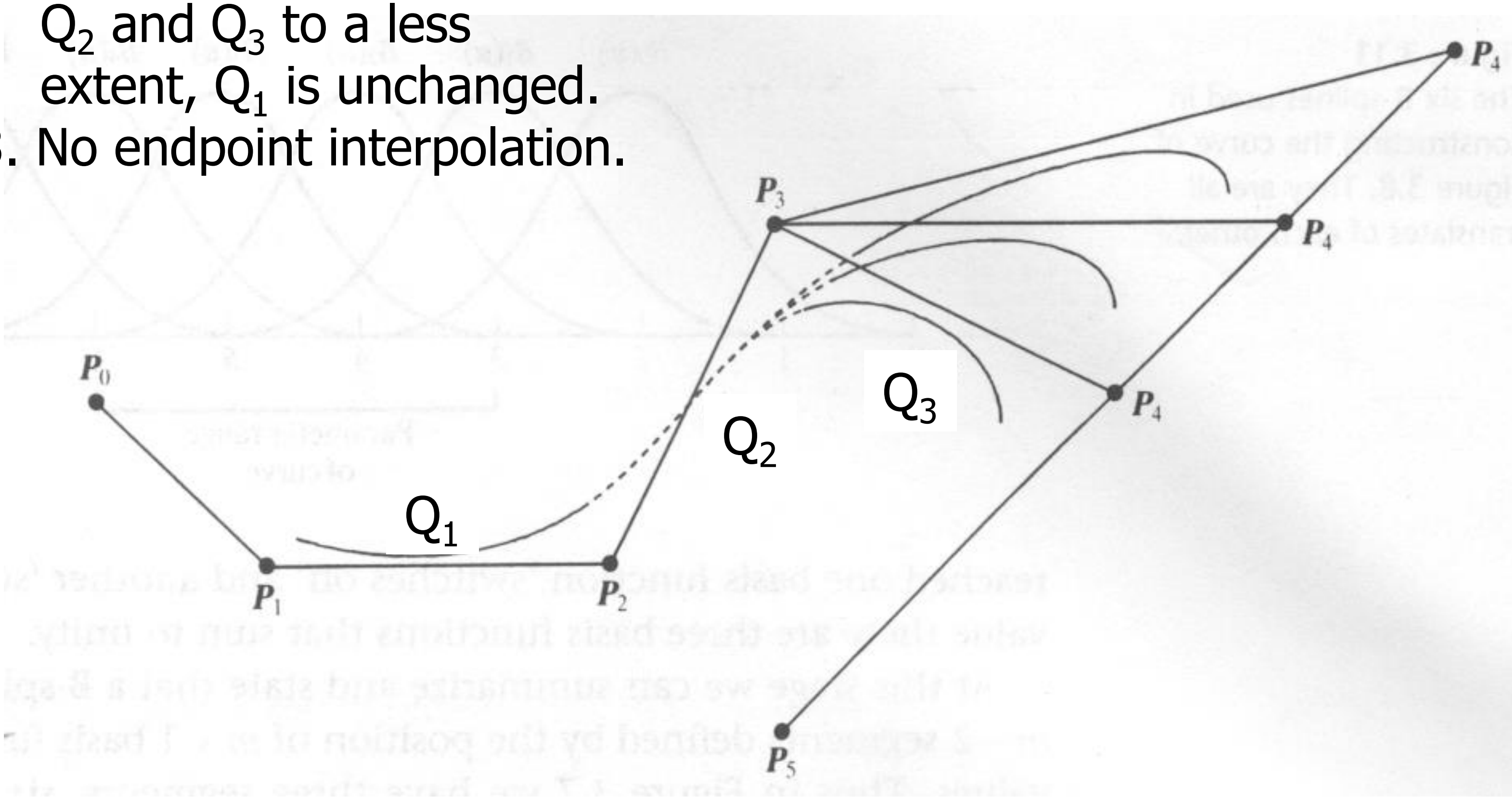
Figure 3.9
Successive B-splines joined together

B-spline curves

2. Cubic B-spline is C^2-continuous within each segment and at the joint. This is automatically satisfied due to its definition.
3. $p_0,\ldots,p_n$ specify a series of n-2 curve segments $B_1(u), B_2(u), \cdots, B_{n-2}(u)$. We need two more segments.
4. $B_i(u)$ is within the concex hull of the four points p_{i-1}, p_i, p_{i+1}, and p_{i+2}. This is a better convex-hull property than Bezier curves.
5. Local control: Moving p_i affects four segments B_{i-2}, B_{i-1}, B_i, and B_{i+1}.

B-spline curves

1. Moving a control point influences 4 segments.
2. Moving P_4 changes Q_2 and Q_3 to a less extent, Q_1 is unchanged.
3. No endpoint interpolation.



B-spline curves

- If we have points $p_0, p_1, \ldots, p_n$, we can construct $B_1, B_2, \ldots, B_{n-2}$.
  - We need B_0 and B_{n-1}
    - Add one point coincide with the end point?
      - B_0 and B_{n-1} don't fit the end points!
    - Add two points coincide with the end points
      - Add $p_{-2}, p_{-1}, p_{n+1}, p_{n+2}$, with $p_{-2}=p_{-1}=p_0$ and $p_n=p_{n+1}=p_{n+2}$
      - The new curves not only join properly with the portions already made, but start and end at the end points

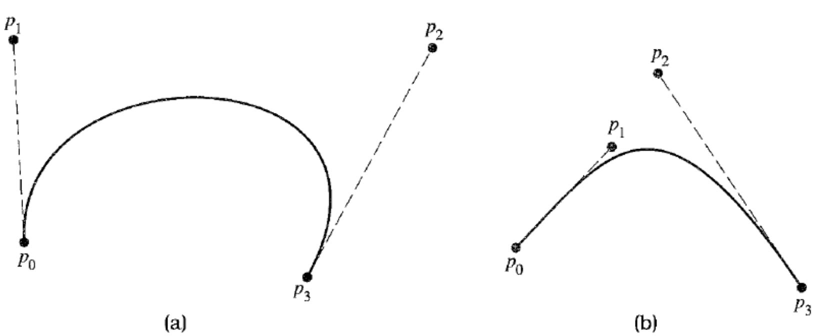
$$B_0(0) = p_0 \text{ and } B_{n-1}(1) = p_n$$

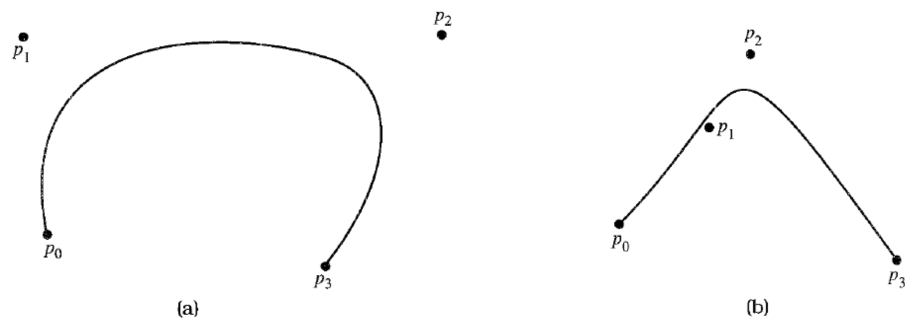
$$B_0'(1) = B_1'(0) \text{ and } B_{n-2}'(1) = B_{n-1}'(0)$$

$$B_0''(1) = B_1''(0) \text{ and } B_{n-2}''(1) = B_{n-1}''(0)$$

B-spline curves Examples

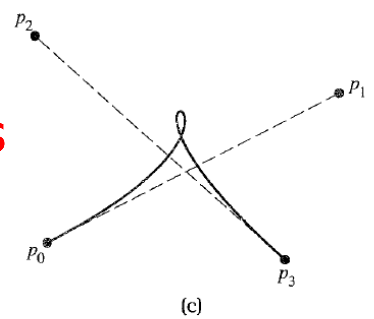
- **Defined by the same sets of points as the Bezier curves (on the left)**



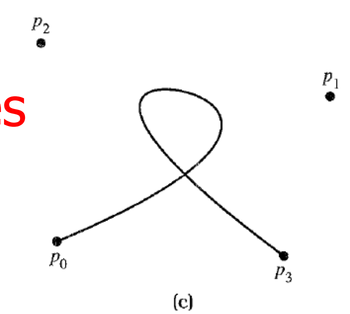


Fictitious points have been added!

Bezier curves

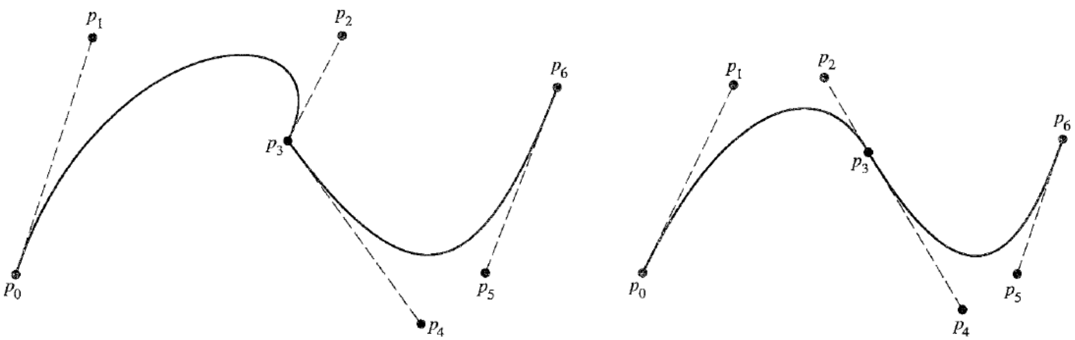


B-spline curves



B-spline curves Examples

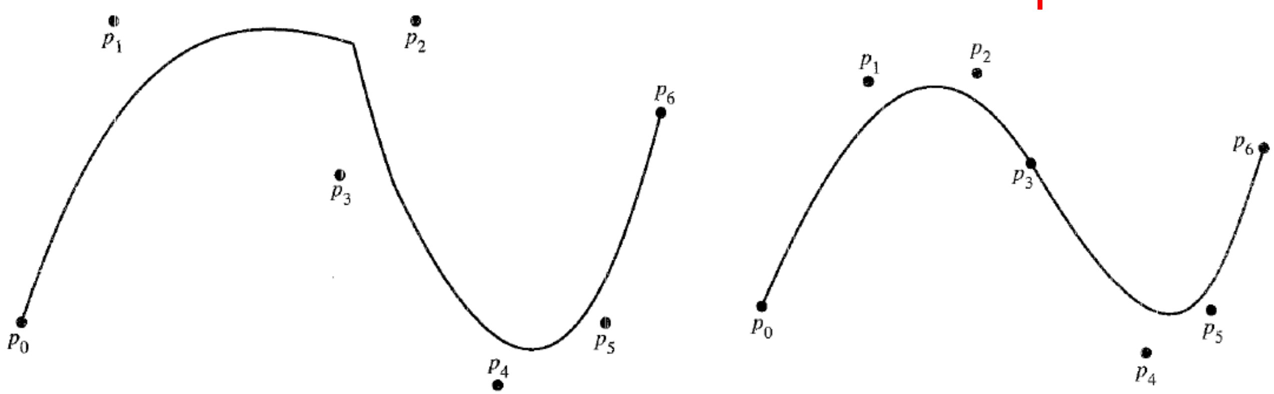
- **Defined by the same sets of points as the previous Bezier curves**



(d) (e)

Bezier curves

B-spline curves

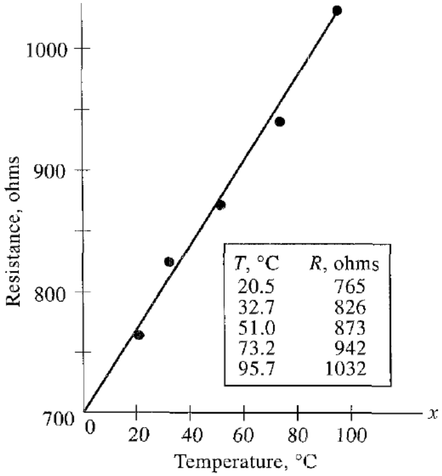


(d) (e)

Least-square approximations

- Until now, we have assumed that the data are accurate.
- But for measurement data, they have errors. For example,
  - The graph suggest a linear relationship:

    y=ax+b
  - Fitting a line that is
    - Unambiguous
    - Minimizes the deviation of the points from the line
      - Deviation: distance between line and points
        - » Depends on whether there are errors in both variables or in just one of them



T, °C	R, ohms
20.5	765
32.7	826
51.0	873
73.2	942
95.7	1032

Figure 3.13

Least-square approximations

- **Linear case**
  - Given a set of points, find a line y=ax+b that achieving some criterion:
    - Minimizing sum of absolute error (See next page)
    - Minimizing maximum error (rarely done!)
    - Minimizing the square sum: widely used since the minimization turns out to be easy
- **Nonlinear case**
  - Popular forms
    - $y=ax^b$
    - $y=ae^{bx}$

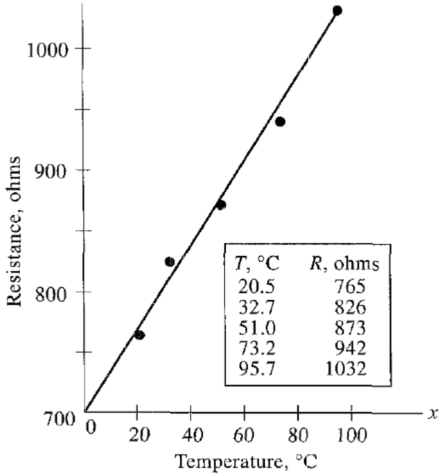


Figure 3.13

Least-square approximations

- Minimize sum of deviations (errors)
  - Not an adequate criterior
  - Fig 3.14:
    - Best line passes two points
    - Line passing the midpoint has also a sum of errors 0
  - Fig 3.15:
    - Best line passes the averag of the duplicated points
    - Any line falling between the dotted lines has the same error sum

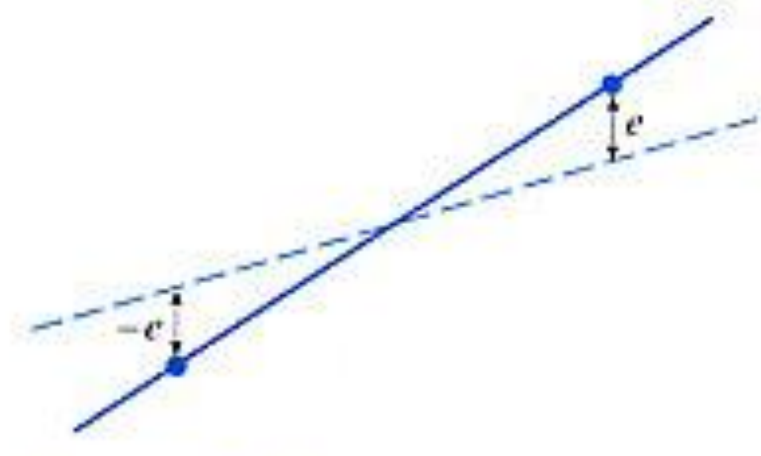


Figure 3.14

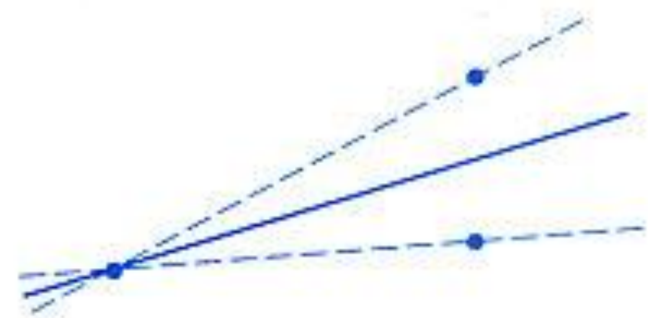


Figure 3.15

Least square approximations

Given sample data $(x_i, Y_i), i = 1, 2, \cdots, N$.

Let Y_i represent an experimental value,

y_i be a value from the equation

$$y_i = ax_i + b$$

where x_i is a particular value of the variable assumed to be free of error.

Goal :

To determine the best values for a and b so that the sum of squares of the error is minimized.

Let $e_i = Y_i - y_i$.

Square of errors :

$$S = e_1^2 + e_2^2 + \cdots + e_N^2$$

$$= \sum_{i=1}^{N} e_i^2$$

$$= \sum_{i=1}^{N} (Y_i - ax_i - b)^2$$

Least square approximations

At a minimum for S, the two partials $\partial S / \partial a$ and $\partial S / \partial b$ will both be zero.

$$\frac{\partial S}{\partial a} = 0 = \sum_{i=1}^{N} 2(Y_i - a x_i - b)(-x_i)$$

$$\frac{\partial S}{\partial b} = 0 = \sum_{i=1}^{N} 2(Y_i - a x_i - b)(-1)$$

Normal equations :

$$a \sum_{i=1}^{N} x_i^2 + b \sum_{i=1}^{N} x_i = \sum_{i=1}^{N} x_i Y_i$$

$$a \sum_{i=1}^{N} x_i + bN = \sum_{i=1}^{N} Y_i$$

Solving the equations gives a and b.

Least square approximations Example

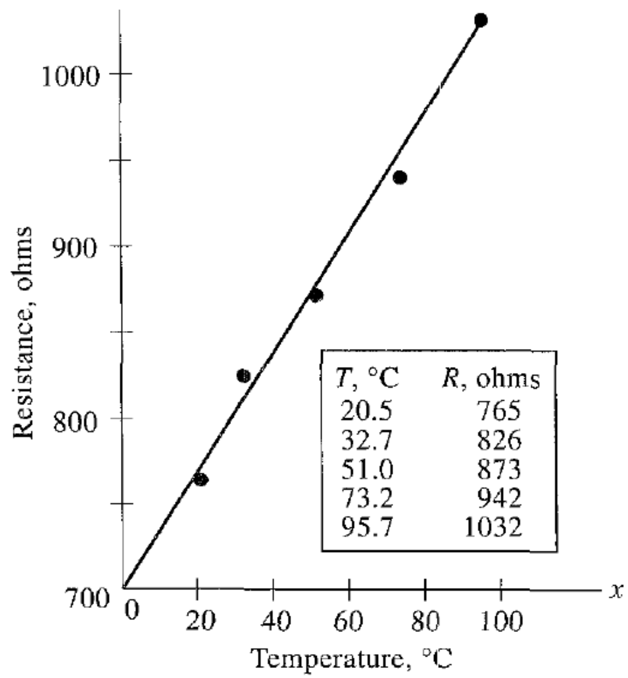


Figure 3.13

$$\sum T_i = 273.1,\ \sum T_i^{\ 2} = 18607.27$$
$$\sum R_i = 4438,\ \sum T_i R_i = 254932.5$$

Normal equation :

$$18607.27a + 273.1b = 254932.5$$
$$273.1a + 5b = 4438$$

Solution : $a = 3.395, b = 702.2$

$$R = 702 + 3.39T$$

Least square approximations Nonlinear case

- **Approximation by**

  $y=ax^b$, or

  $y=ae^{bx}$

- **Normal equations are nonlinear, which is much more difficult to solve**
- **Linearize by taking logarithms before determining the parameters by least square**

  $\ln y= \ln a+b \ln x$, or

  $\ln y=\ln a +bx$

So we fit $z=\ln y$ as a linear function of $\ln x$ or x

Least-square polynomials

- Least square polynomials
  - Polynomials can be readily manipulated, fitting polynomials to data that do not plot linearly is common
  - Its normal equations are linear!
- N: # of data pairs, n: polynomial degree
  - If $N=n+1$, the polynomial passes exactly through each point, so it is the interpolating polynomial
  - Here we have $N > n+1$

Least-square polynomials

Given N points, (x_i, Y_i), $i = 1,2,\cdots N$. Find a polynomial of degree n to approximate the date in least-square sense.

Here, $N > n+1$.

Assume the function relationship

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

with errors defined by

$$e_i = Y_i - y_i$$
$$= Y_i - a_0 - a_1 x_i - \cdots - a_n x_i^{\ n}.$$

Sum of squares

$$S = \sum_{i=1}^{N} e_i^2$$
$$= \sum_{i=1}^{N} (Y_i - a_0 - a_1 x_i - a_2 x_i^{\ 2} - \cdots - a_n x_i^{\ n})^2$$

Normal equations :

$$\frac{\partial S}{\partial a_0} = 0 = \sum_{i=1}^{N} 2(Y_i - a_0 - a_1 x_i - \cdots - a_n x_i^{\ n})(-1)$$

$$\frac{\partial S}{\partial a_1} = 0 = \sum_{i=1}^{N} 2(Y_i - a_0 - a_1 x_i - \cdots - a_n x_i^{\ n})(-x_i)$$

$$\vdots$$

$$\frac{\partial S}{\partial a_n} = 0 = \sum_{i=1}^{N} 2(Y_i - a_0 - a_1 x_i - \cdots - a_n x_i^{\ n})(-x_i^{\ n})$$

Least-square polynomials Normal equations

$$
\begin{aligned}
a_0 N + a_1 \sum x_i + a_2 \sum x_i^2 + \cdots + a_n \sum x_i^n &= \sum Y_i, \\
a_0 \sum x_i + a_1 \sum x_i^2 + a_2 \sum x_i^3 + \cdots + a_n \sum x_i^{n+1} &= \sum x_i Y_i, \\
a_0 \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4 + \cdots + a_n \sum x_i^{n+2} &= \sum x_i^2 Y_i, \\
\vdots \qquad\qquad & \qquad \vdots \\
a_0 \sum x_i^n + a_1 \sum x_i^{n+1} + a_2 \sum x_i^{n+2} + \cdots + a_n \sum x_i^{2n} &= \sum x_i^n Y_i.
\end{aligned}
\tag{3.27}
$$

$$
B\,[a] = \begin{bmatrix}
N & \sum x_i & \sum x_i^2 & \sum x_i^3 & \ldots & \sum x_i^n \\
\sum x_i & \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \ldots & \sum x_i^{n+1} \\
\sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \sum x_i^5 & \ldots & \sum x_i^{n+2} \\
 & & \vdots & & & \vdots \\
\sum x_i^n & \sum x_i^{n+1} & \sum x_i^{n+2} & \sum x_i^{n+3} & \ldots & \sum x_i^{2n}
\end{bmatrix} [a] = \begin{bmatrix}
\sum Y_i \\ \sum x_i Y_i \\ \sum x_i^2 Y_i \\ \vdots \\ \sum x_i^n Y_i
\end{bmatrix}. \tag{3.28}
$$

(n+1)x(n+1)

Least-square polynomials Problems

- Solving large set of normal equations is not a simple task. Moreover, it is ill-conditioned when the degree is high
  - Accumulated round-off in the summation
  - The system often becomes ill conditioned quite rapidly as *n* increases
    - Up to *n*=3 or 4, the problem is not too great.
      - Special methods tat use orthogonal polynomials are a remedy
    - Beyond that, are rarely needed, and can be handled by fitting a series of polynomials to subsets of the data
- There is polynomial wiggle if data do not exhibit a polynomial nature

Least-square polynomials
Use of orthogonal polynomials

Six data points generated by $f(x) = \frac{1.44}{x^2} + 0.24x$

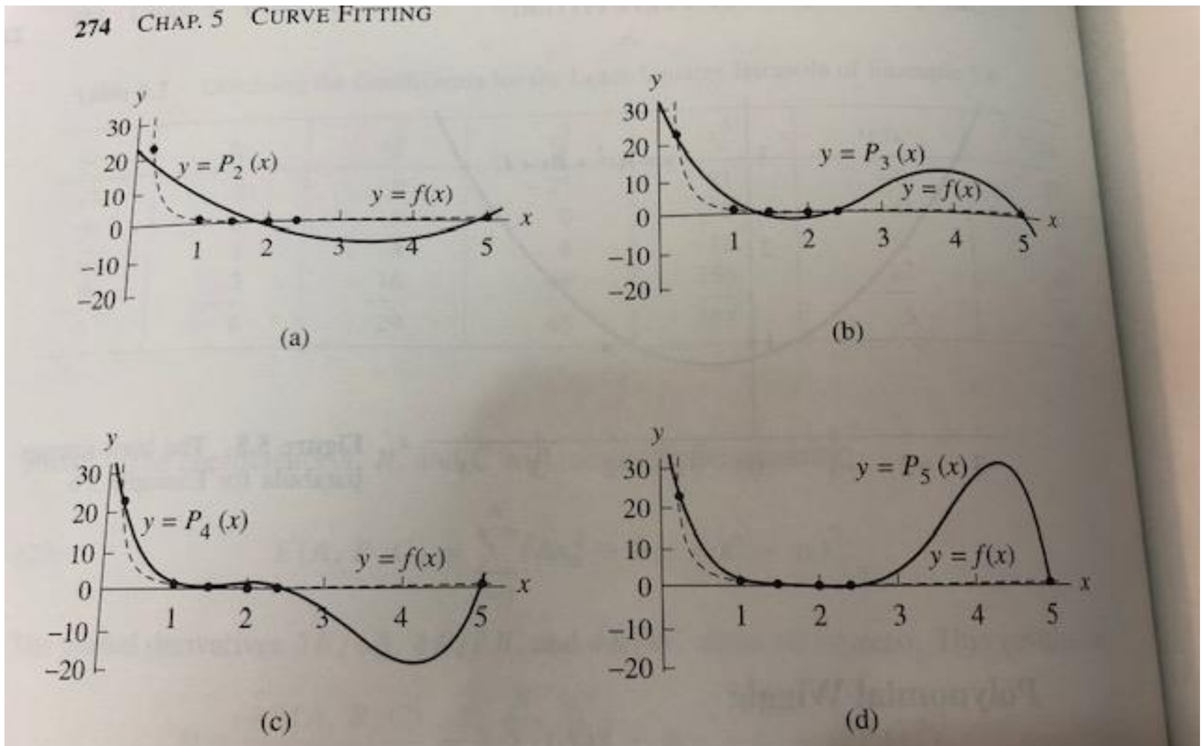


Figure 5.9 (a) Using $P_2(x)$ to fit data. (b) Using $P_3(x)$ to fit data. (c) Using $P_4(x)$ to fit data. (d) Using $P_5(x)$ to fit data.

Least-square polynomials
Use of orthogonal polynomials

- The normal equation system is ill-conditioned when the degree is high
  - Even for a cubic least-square polynomial, the condition number of the coefficient matrix can be large
  - Example:
    - Fitting a cubic polynomial to 21 data points, the condition number was found to be 22000
  - If we fit the data with orthogonal polynomials such as Chebyshev polynomial (in Chap 4) the condition number was reduced to about 5.

Least-square polynomials Solve normal eq.

For low-degree polynomial:

Design matrix: $(n+1)\times N$

$$A = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_N \\ \vdots & & & \vdots \\ x_1^n & x_2^n & \cdots & x_N^n \end{bmatrix}$$

Elements could vary Significantly!

We can show that

$B = AA^T$ and cond(AA^T)=cond(A)^2 !!

Ay = right-hand side of Eq 3.28

Rewrite Eq. 3.28 as

$$AA^T a = Ba = Ay$$

We can use Gaussian elimination to solve the system. However, because B has special properties, another method can be used to avoid the problem of ill-conditioning.

Least-square polynomials Solve normal eq. by SVD

1. $B = AA^T$ is symmetric and positive definite.

2. B can be diagonized by an orthogonal matrix P:

$$PBP^T = PAA^TP^T = D,$$

where the diagonal elements of D are the eigenvaules of B. Note that $PP^T = I$.

3. B is positive definite, so all of its eigenvalues are nonnegative. Thus, there is a S such that

$$S = \sqrt{D}$$

The diagonal elements of S are called the singular values of A.

4. Since $PBP^T = D$, $B = P^T DP$. Normal equation can be rewritten as

$$AA^T a = P^T DPa$$

$$= (SP)^T (SP)a = Ay$$

and

$$a = P^T D^{-1} PAy$$

Least-square polynomials Solve normal eq. by SVD

Linear Algebra, by Leon, Page 344 - 348

1. A symmetric $n \times n$ matrix B is said positive definite if $x^T Bx > 0$ for all nonzero x.
2. B is symmetric positive definite iff all its eigenvalue s are positive.
3. If B is symmetric positive definite, then B is nonsingula r.
4. If B is symmetric positive definite, then B can be factored into $B = LDL^T$, where L is lower triangular wit h 1's along the diagonal and D is a diagonal matrix whose diagonal elements are all positive.
5. If B is symmetric positive definite, B can be diagonized by an orthogonal matrix P:

$$PBP^T = D, \text{ i.e., } B = P^T DP$$

where the diagonal elements of D are the eigenvaule s of B.

Note that $PP^T = I$.

Least-square polynomials Example

$$11a_0 + \quad 6.01a_1 + 4.6545a_2 = 5.905,$$
$$6.01a_0 + 4.6545a_1 + 4.1150a_2 = 2.1839,$$
$$4.6545a_0 + 4.1150a_1 + 3.9161a_2 = 1.3357.$$

The result is $a_0 = 0.998$, $a_1 = -1.018$, $a_2 = 0.225$.

Lease square solution: $y = 0.998 - 1.018x + 0.225x^2$.

True: $y = 1 - x + 0.2x^2$.

Table 3.14 Data to illustrate curve fitting

x_i	0.05	0.11	0.15	0.31	0.46	0.52	0.70	0.74	0.82	0.98	1.171
Y_i	0.956	0.890	0.832	0.717	0.571	0.539	0.378	0.370	0.306	0.242	0.104

$\Sigma x_i = 6.01$	$N = 11$
$\Sigma x_i^2 = 4.6545$	$\Sigma Y_i = 5.905$
$\Sigma x_i^3 = 4.1150$	$\Sigma x_i Y_i = 2.1839$
$\Sigma x_i^4 = 3.9161$	$\Sigma x_i^2 Y_i = 1.3357$

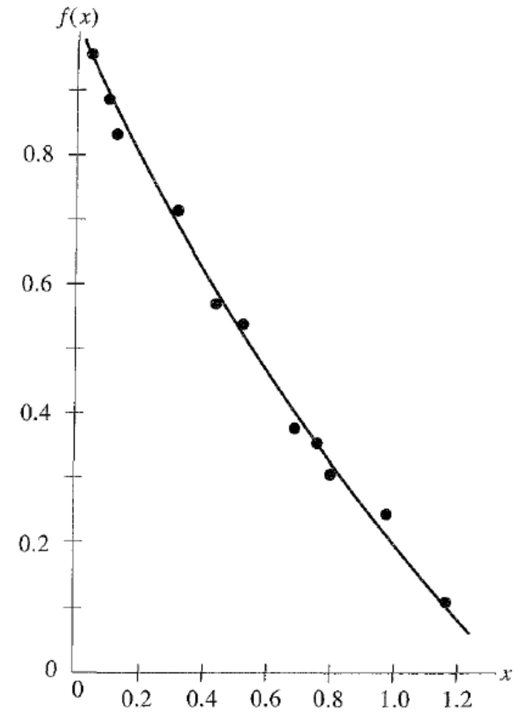


Figure 3.16

Least-square polynomials Degree of polynomial

What degree polynomial should be used?

- **For given N points, higher-degree polynomial reduces the error, but leading to wiggle problem**
  - **If the data points lie on a curve that is not polynomial-like, high-degree polynomial curves will oscillation between successive points when forced to go near them**
    - **The remedy for poor polynomial fit is a more suitable smooth function, not a polynomial of high degree**
- **When $n=N-1$, the least-square solution is an interpolating polynomial**

Least-square polynomials What degree ?

- **One increases the degree of approximating polynomial as long as there is a statistically significant decrease in the variance:**

$$\sigma^2 = \frac{\sum e_i^2}{N - n - 1}$$

- **Example**

Based on the criterion, we choose the optimum degree as 2.

Table 3.15

Degree	Equation	σ^2 (Eq. 3.27)	$\sum e^2$
1	$y = 0.95228 - 0.76041x$	0.00106	0.00915
2	$y = 0.99800 - 1.0180x + 0.22468x^2$	0.00023	0.00187
3	$y = 1.0037 - 1.0794x + 0.35137x^2 - 0.06894x^3$	0.00026	0.00181
4	$y = 0.98810 - 0.83690x - 0.52680x^2 + 1.0461x^3 - 0.45635x^4$	0.00027	0.00165
5	$y = 1.0369 - 1.8241x + 4.8953x^2 - 10.753x^3 + 10.537x^4 - 3.6594x^5$	0.00013	0.00067