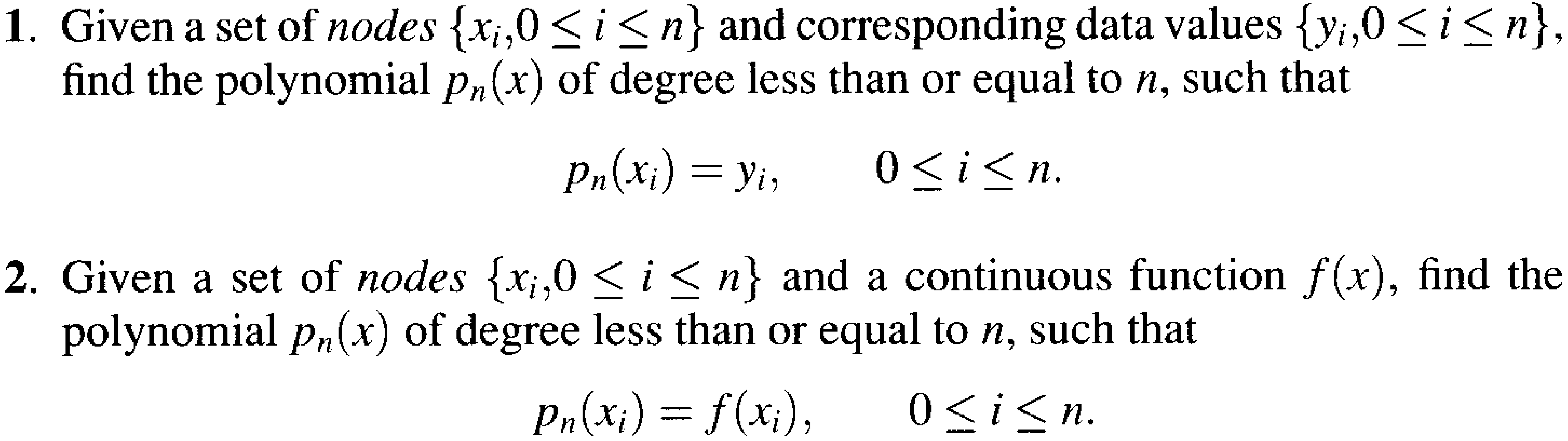
Analyse Numérique 1

Salem Nafiri

Ecole Hassania des Travaux Publics

Interpolation et Approximation

Interpolation de Lagrange

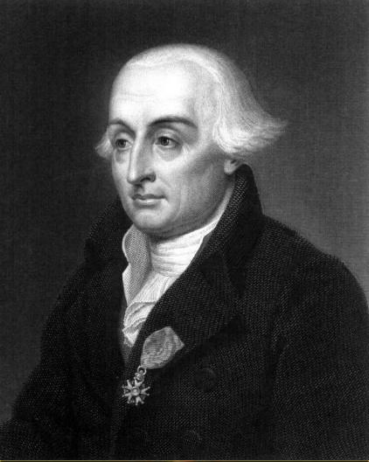


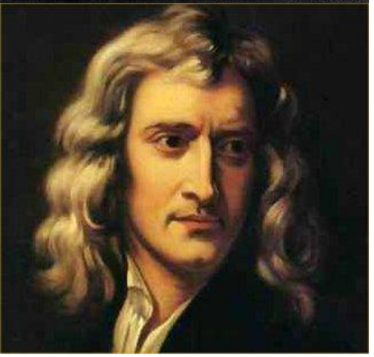
1. Given a set of *nodes* $\{x_i, 0 \le i \le n\}$ and corresponding data values $\{y_i, 0 \le i \le n\}$, find the polynomial $p_n(x)$ of degree less than or equal to n, such that

$$p_n(x_i) = y_i, \qquad 0 \le i \le n.$$

2. Given a set of *nodes* $\{x_i, 0 \le i \le n\}$ and a continuous function $f(x)$, find the polynomial $p_n(x)$ of degree less than or equal to n, such that

$$p_n(x_i) = f(x_i), \qquad 0 \le i \le n.$$



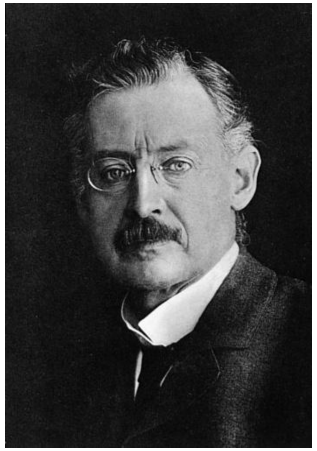


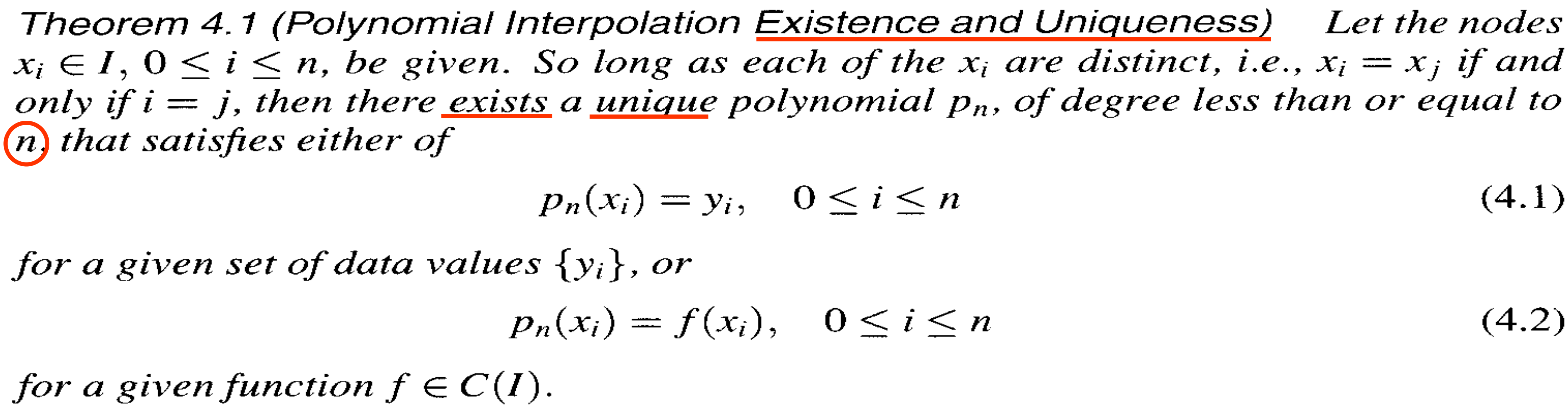
1. Given a set of *nodes* $\{x_i, 0 \leq i \leq n\}$ and corresponding data values $\{y_i, 0 \leq i \leq n\}$, find the polynomial $p_n(x)$ of degree less than or equal to n, such that

$$p_n(x_i) = y_i, \qquad 0 \leq i \leq n.$$

2. Given a set of *nodes* $\{x_i, 0 \leq i \leq n\}$ and a continuous function $f(x)$, find the polynomial $p_n(x)$ of degree less than or equal to n, such that

$$p_n(x_i) = f(x_i), \qquad 0 \leq i \leq n.$$





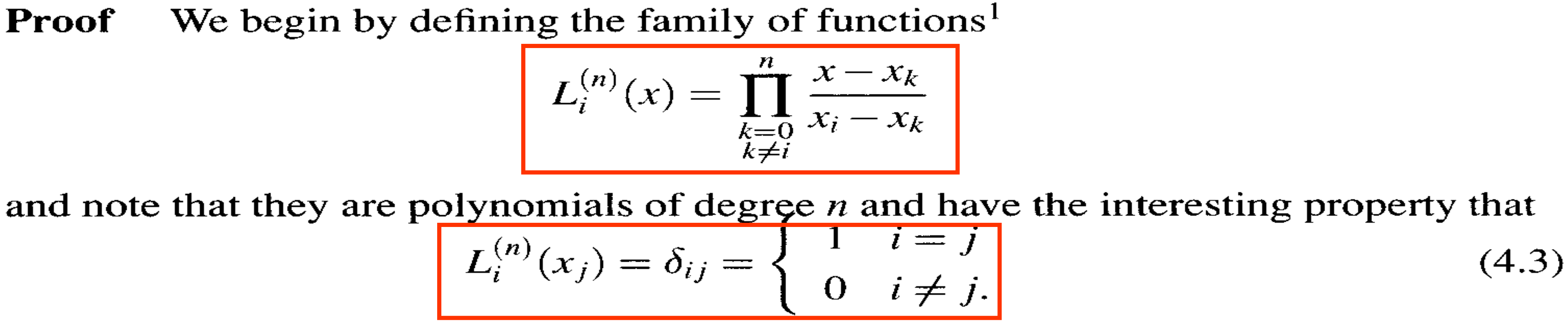
Theorem 4.1 (Polynomial Interpolation Existence and Uniqueness) *Let the nodes $x_i \in I$, $0 \le i \le n$, be given. So long as each of the x_i are distinct, i.e., $x_i = x_j$ if and only if $i = j$, then there exists a unique polynomial p_n, of degree less than or equal to n, that satisfies either of*

$$p_n(x_i) = y_i, \quad 0 \le i \le n \tag{4.1}$$

for a given set of data values $\{y_i\}$, or

$$p_n(x_i) = f(x_i), \quad 0 \le i \le n \tag{4.2}$$

for a given function $f \in C(I)$.

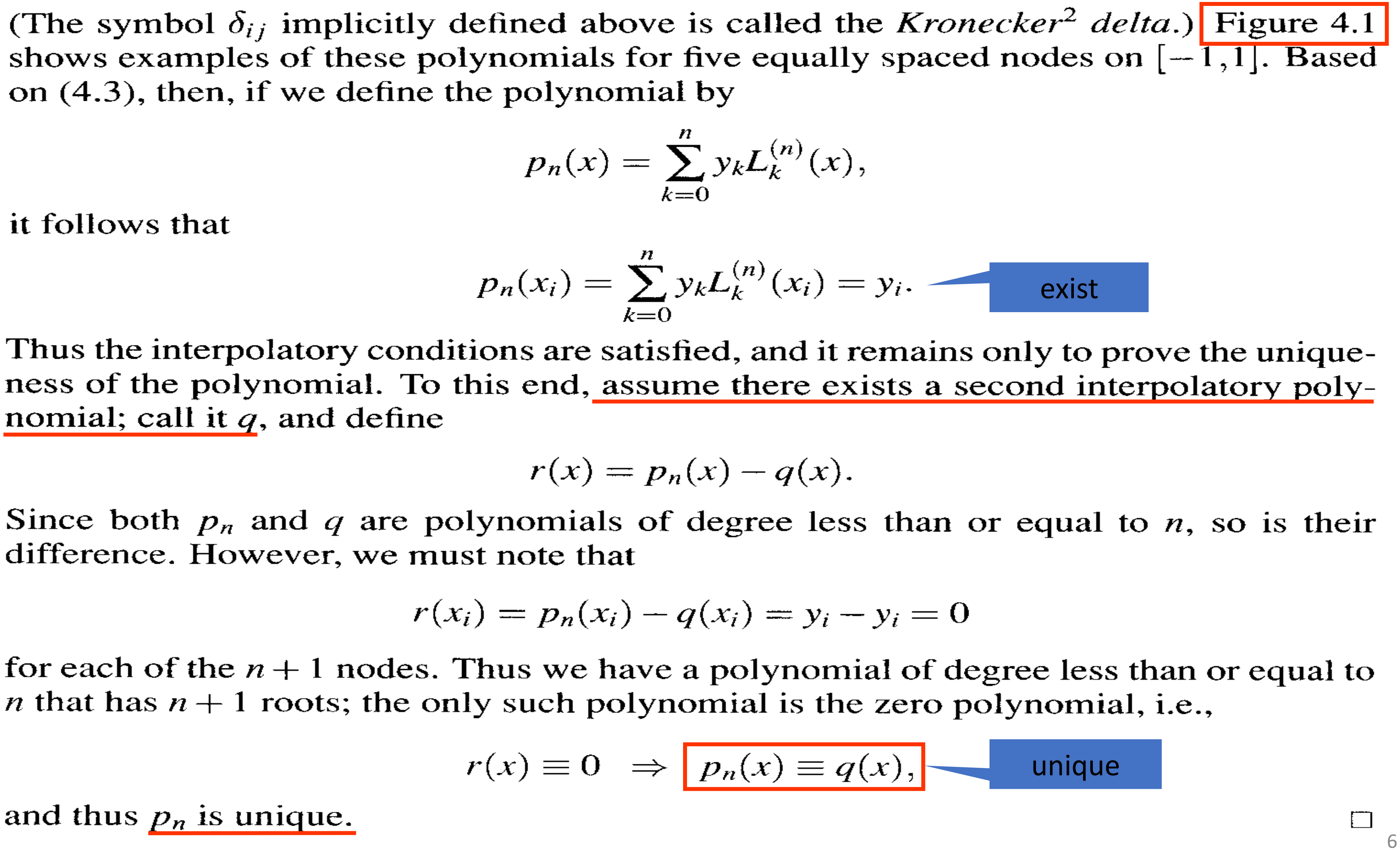


Proof We begin by defining the family of functions[1]

$$\boxed{L_i^{(n)}(x) = \prod_{\substack{k=0 \\ k \ne i}}^{n} \frac{x - x_k}{x_i - x_k}}$$

and note that they are polynomials of degree n and have the interesting property that

$$\boxed{L_i^{(n)}(x_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \ne j. \end{cases}} \tag{4.3}$$



(The symbol δ_{ij} implicitly defined above is called the *Kronecker*^{2} *delta*.) Figure 4.1 shows examples of these polynomials for five equally spaced nodes on $[-1,1]$. Based on (4.3), then, if we define the polynomial by

$$p_n(x) = \sum_{k=0}^{n} y_k L_k^{(n)}(x),$$

it follows that

$$p_n(x_i) = \sum_{k=0}^{n} y_k L_k^{(n)}(x_i) = y_i.$$

exist

Thus the interpolatory conditions are satisfied, and it remains only to prove the uniqueness of the polynomial. To this end, assume there exists a second interpolatory polynomial; call it q, and define

$$r(x) = p_n(x) - q(x).$$

Since both p_n and q are polynomials of degree less than or equal to n, so is their difference. However, we must note that

$$r(x_i) = p_n(x_i) - q(x_i) = y_i - y_i = 0$$

for each of the $n+1$ nodes. Thus we have a polynomial of degree less than or equal to n that has $n+1$ roots; the only such polynomial is the zero polynomial, i.e.,

$$r(x) \equiv 0 \quad \Rightarrow \quad \boxed{p_n(x) \equiv q(x),}$$

unique

and thus p_n is unique. □

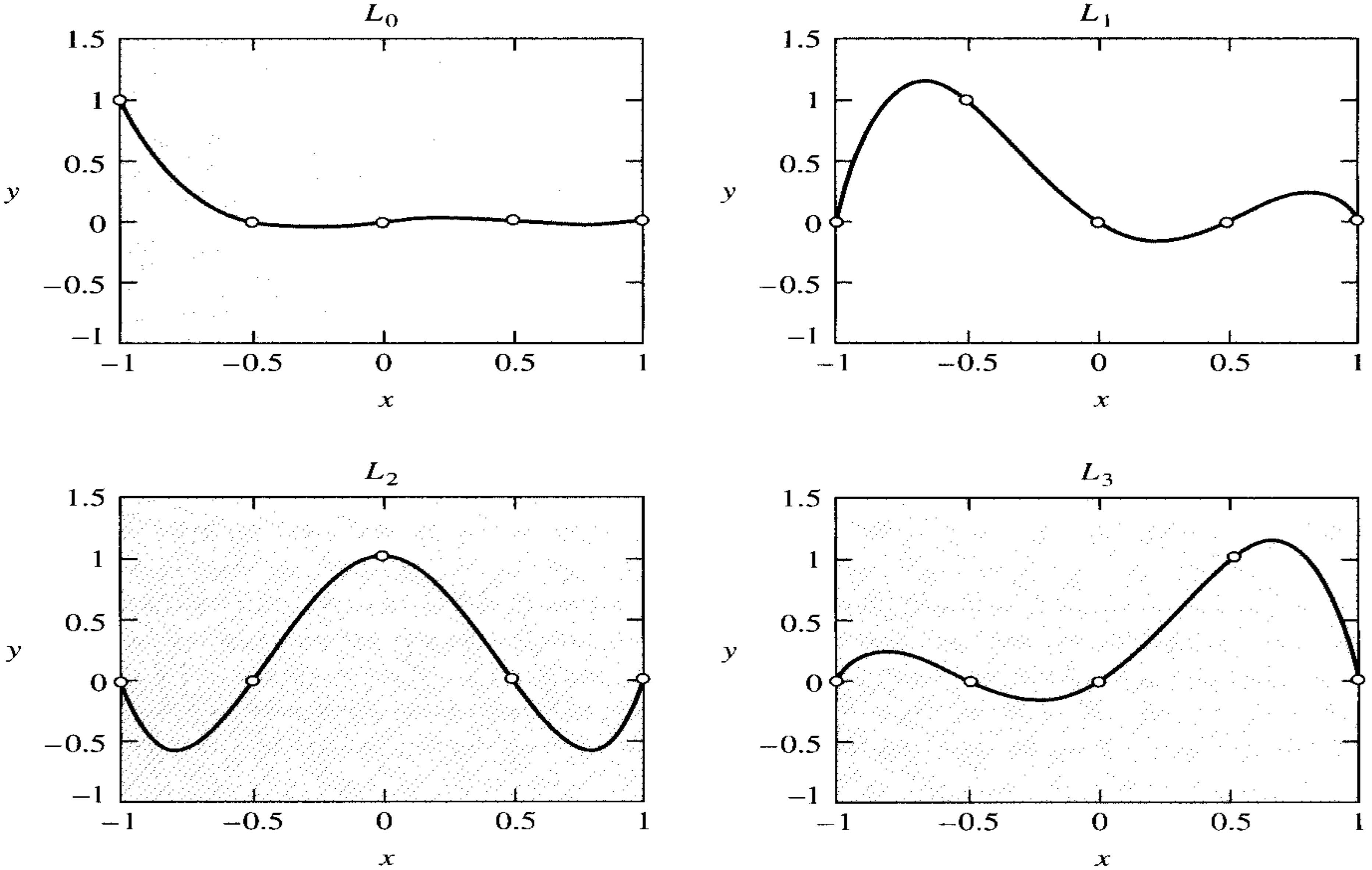
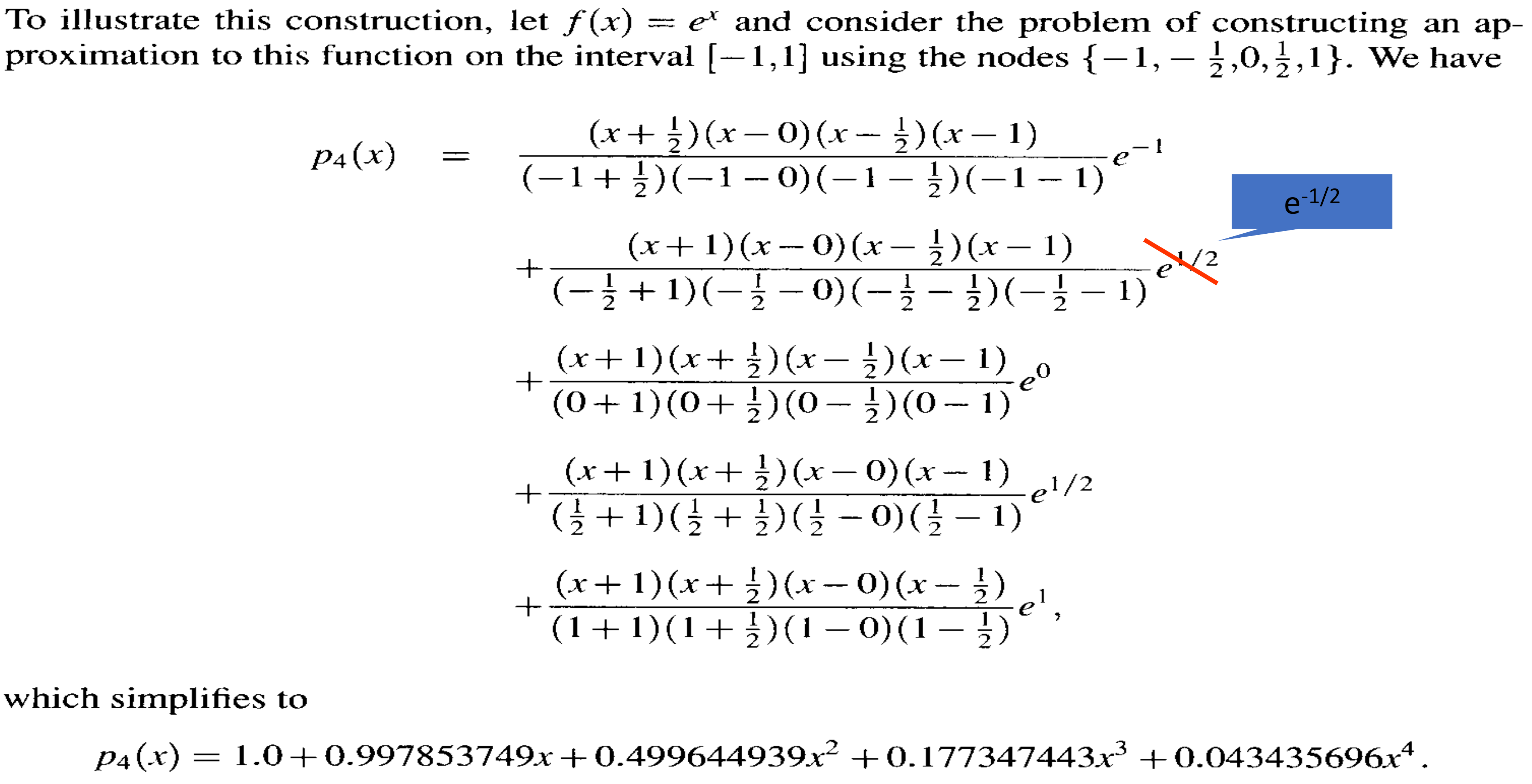


FIGURE 4.1 Lagrange polynomials for five equally spaced nodes on $[-1,1]$.

Example 4.1



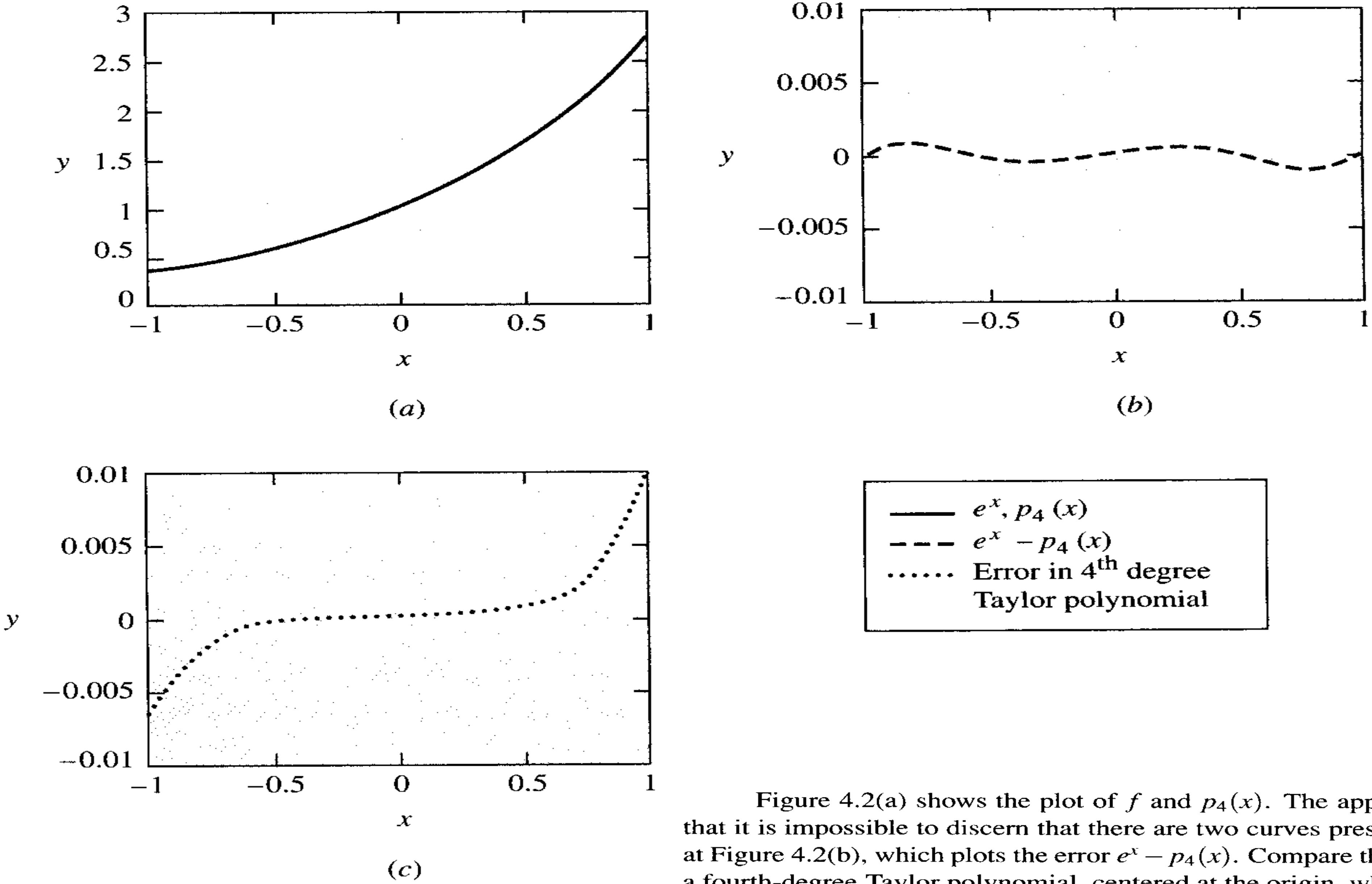
To illustrate this construction, let $f(x) = e^x$ and consider the problem of constructing an approximation to this function on the interval $[-1,1]$ using the nodes $\{-1, -\frac{1}{2}, 0, \frac{1}{2}, 1\}$. We have

$$
\begin{aligned}
p_4(x) \quad = \quad & \frac{(x+\frac{1}{2})(x-0)(x-\frac{1}{2})(x-1)}{(-1+\frac{1}{2})(-1-0)(-1-\frac{1}{2})(-1-1)}e^{-1} \\
& + \frac{(x+1)(x-0)(x-\frac{1}{2})(x-1)}{(-\frac{1}{2}+1)(-\frac{1}{2}-0)(-\frac{1}{2}-\frac{1}{2})(-\frac{1}{2}-1)}e^{\cancel{1/2}} \\
& + \frac{(x+1)(x+\frac{1}{2})(x-\frac{1}{2})(x-1)}{(0+1)(0+\frac{1}{2})(0-\frac{1}{2})(0-1)}e^{0} \\
& + \frac{(x+1)(x+\frac{1}{2})(x-0)(x-1)}{(\frac{1}{2}+1)(\frac{1}{2}+\frac{1}{2})(\frac{1}{2}-0)(\frac{1}{2}-1)}e^{1/2} \\
& + \frac{(x+1)(x+\frac{1}{2})(x-0)(x-\frac{1}{2})}{(1+1)(1+\frac{1}{2})(1-0)(1-\frac{1}{2})}e^{1},
\end{aligned}
$$

$e^{-1/2}$

which simplifies to

$$p_4(x) = 1.0 + 0.997853749x + 0.499644939x^2 + 0.177347443x^3 + 0.043435696x^4.$$



(*a*)

(*b*)

(*c*)

FIGURE 4.2 Interpolation to $f(x) = e^x$.

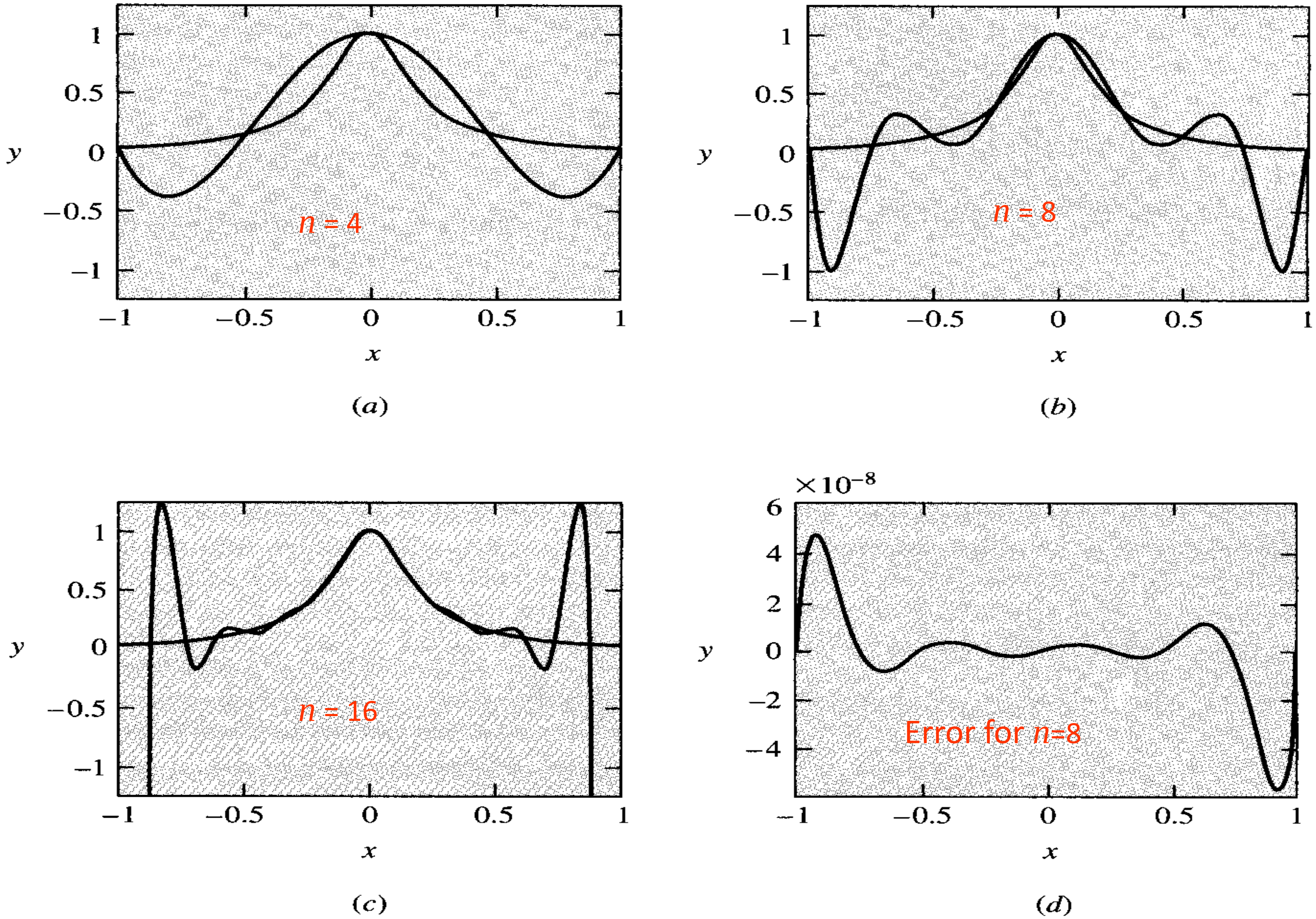
Figure 4.2(a) shows the plot of f and $p_4(x)$. The approximation is sufficiently accurate that it is impossible to discern that there are two curves present, so it is probably better to look at Figure 4.2(b), which plots the error $e^x - p_4(x)$. Compare this with the error obtained by doing a fourth-degree Taylor polynomial, centered at the origin, which is given in Figure 4.2(c). Note that while the Taylor polynomial is *very* accurate near the middle of the interval, and much less accurate near the ends, the interpolating polynomial has less variation in its error, and a much smaller worst-case error over the same interval. ■

Discussion

- The construction presented in this section is called Lagrange interpolation.
- How good is interpolation at approximating a function? (Sections 4.3, 4.11)
- Consider another example:

$$f(x) = (1 + 25x^2)^{-1}.$$

  - If we use a fourth-degree interpolating polynomial to approximate this function, the results are as shown in Figure 4.3 (a).



(a)

(b)

(c)

(d)

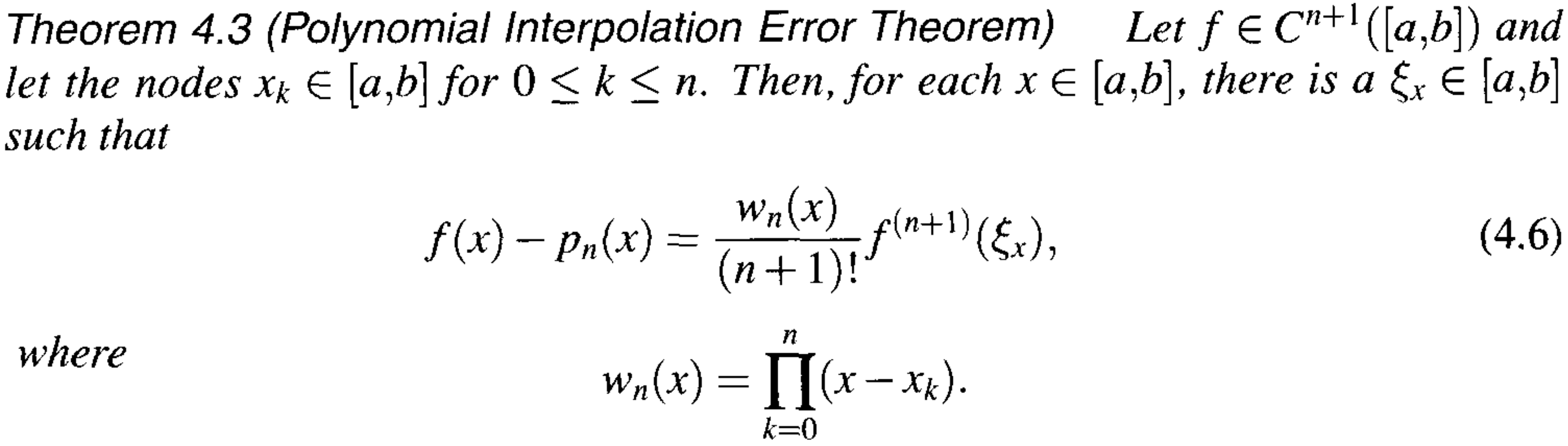
FIGURE 4.3 Polynomial interpolation to $f(x) = (1 + 25x^2)^{-1}$ with $n = 4$ (A), $n = 8$ (B), and $n = 16$ (C), and the error in interpolation for $n = 8$ to $f(x) = e^x$ (D).

Discussion

- There are circumstances in which polynomial interpolation as approximation will work very well, and other circumstances in which it will not.

- The Lagrange form of the interpolating polynomial is not well suited for actual computations, and there is an alternative construction that is far superior to it.

L’erreur d’interpolation



Theorem 4.3 (Polynomial Interpolation Error Theorem) Let $f \in C^{n+1}([a,b])$ and let the nodes $x_k \in [a,b]$ for $0 \leq k \leq n$. Then, for each $x \in [a,b]$, there is a $\xi_x \in [a,b]$ such that

$$f(x) - p_n(x) = \frac{w_n(x)}{(n+1)!} f^{(n+1)}(\xi_x), \tag{4.6}$$

where

$$w_n(x) = \prod_{k=0}^{n} (x - x_k).$$

Inconvenient de la formule de Lagrange

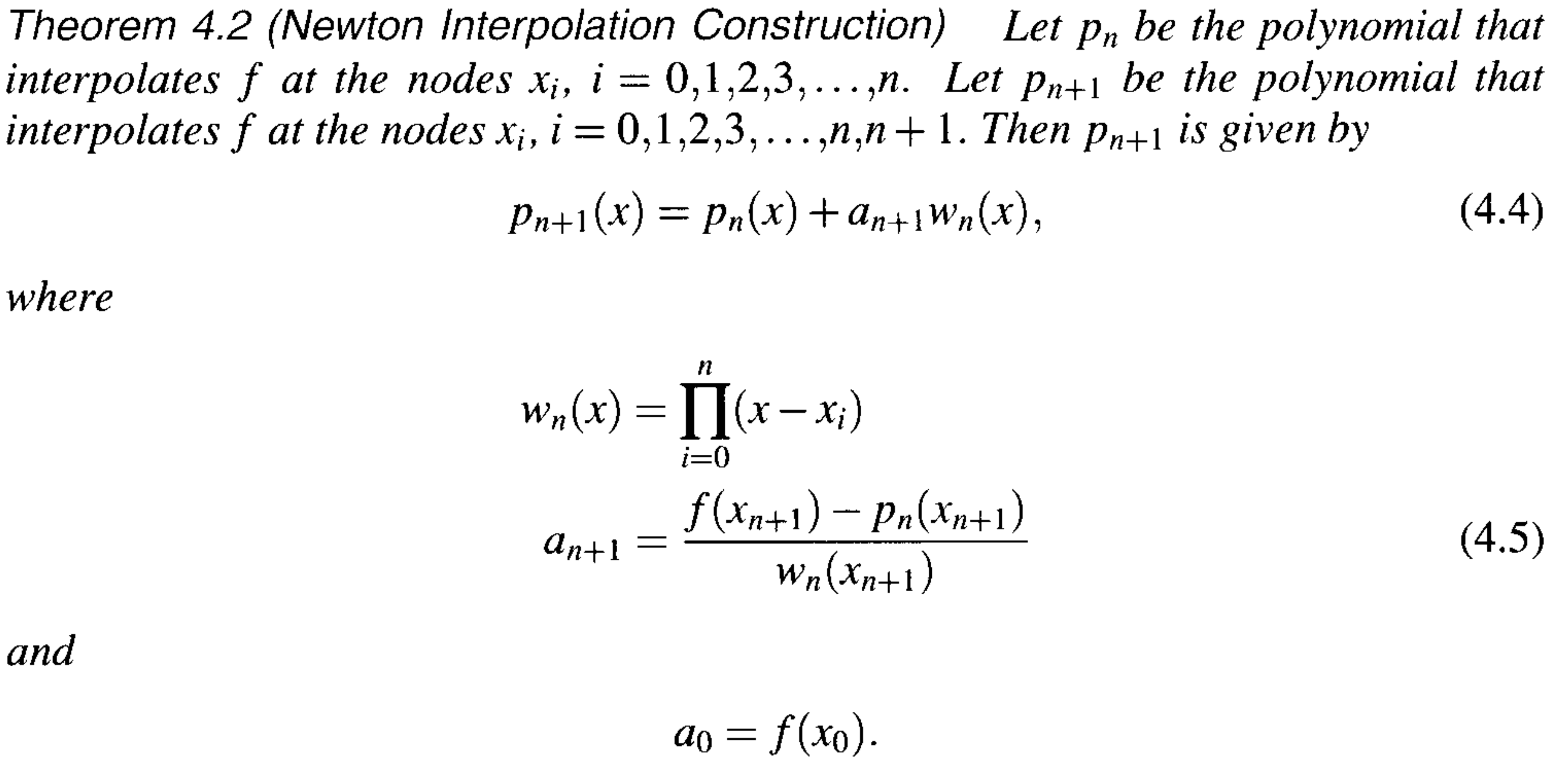
Si on décide d’ajouter un point à l’ensemble des points, on doit recalculer toutes les fonctions de base $L_i^{(n)}$

we cannot (easily) write p_{n+1} in terms of p_n,

Existe il une autre alternative ?

C’est l’interpolation de Newton. Il permet d’avoir une relation de recurrence entre p_{n+1} et p_n.

Interpolation de Newton et différences divisées



Theorem 4.2 (Newton Interpolation Construction) *Let p_n be the polynomial that interpolates f at the nodes x_i, $i = 0,1,2,3,\ldots,n$. Let p_{n+1} be the polynomial that interpolates f at the nodes x_i, $i = 0,1,2,3,\ldots,n,n+1$. Then p_{n+1} is given by*

$$p_{n+1}(x) = p_n(x) + a_{n+1}w_n(x), \tag{4.4}$$

where

$$w_n(x) = \prod_{i=0}^{n}(x - x_i)$$

$$a_{n+1} = \frac{f(x_{n+1}) - p_n(x_{n+1})}{w_n(x_{n+1})} \tag{4.5}$$

and

$$a_0 = f(x_0).$$

Proof Since we know that the interpolation polynomial is unique, all we have to do is show that p_{n+1}, as given in (4.4), satisfies the interpolation conditions. For $k \le n$, we have $w_n(x_k) = 0$, so

$$p_{n+1}(x_k) = p_n(x_k) + a_{n+1}\underbrace{w_n(x_k)}_{=0} = p_n(x_k) = f(x_k);$$

hence p_{n+1} interpolates all but the last point. To check x_{n+1}, we directly compute

$$\begin{aligned} p_{n+1}(x_{n+1}) &= p_n(x_{n+1}) + a_{n+1}w_n(x_{n+1}) \\ &= p_n(x_{n+1}) + f(x_{n+1}) - p_n(x_{n+1}) \\ &= f(x_{n+1}). \end{aligned}$$

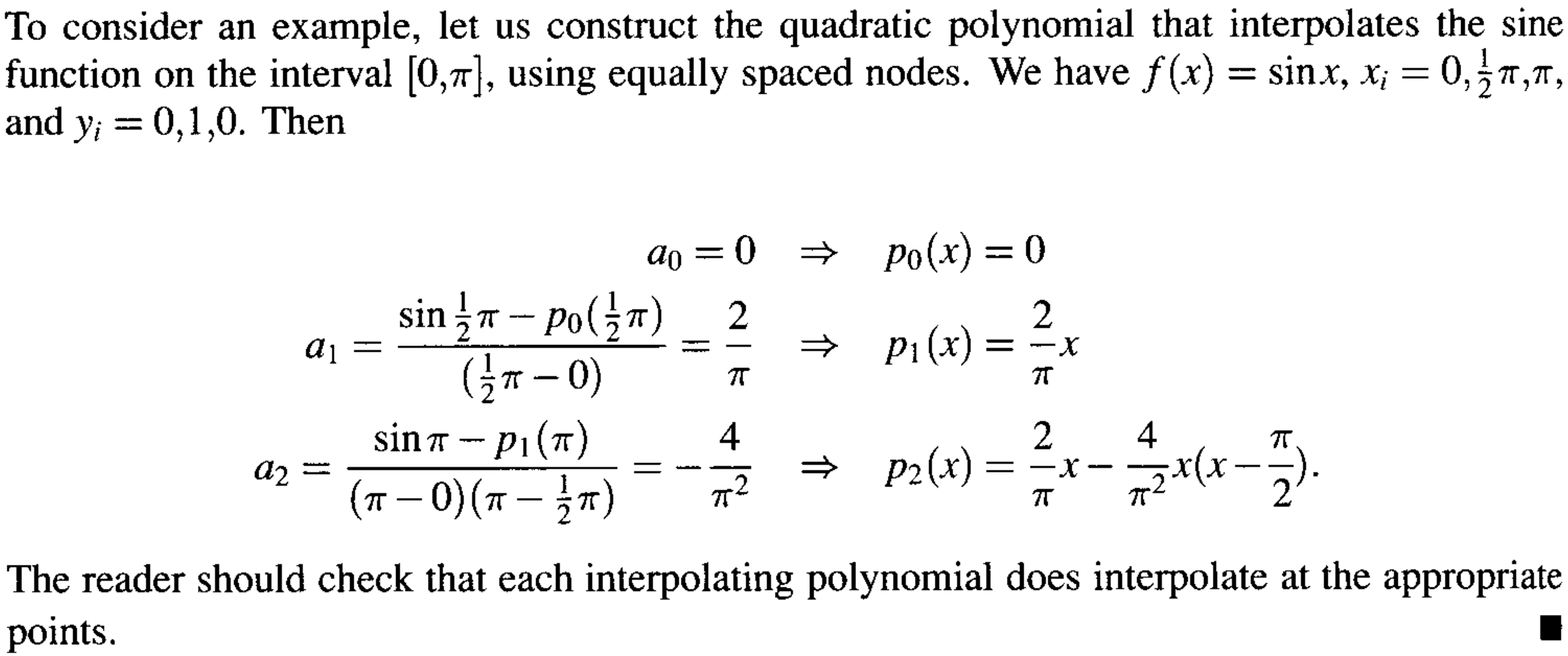
Thus p_{n+1} interpolates at all the nodes; moreover, it clearly is a polynomial of degree less than or equal to $n+1$, and so we are done. □

Corollary 4.1 *For $\{a_k\}$ and w_n as defined in Theorem 4.2, we have*

$$\begin{aligned} p_n(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1}) \\ &= \sum_{k=0}^{n} a_k w_{k-1}(x) \end{aligned}$$

where we have used $w_{-1}(x) \equiv 1$.

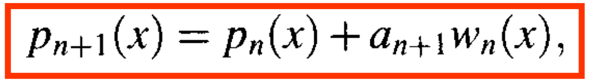
Exemple



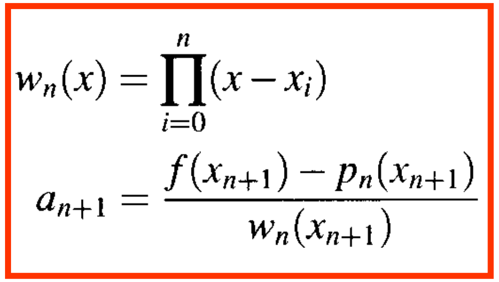
To consider an example, let us construct the quadratic polynomial that interpolates the sine function on the interval $[0,\pi]$, using equally spaced nodes. We have $f(x) = \sin x$, $x_i = 0, \frac{1}{2}\pi, \pi$, and $y_i = 0,1,0$. Then

$$\begin{aligned}
a_0 &= 0 & &\Rightarrow & p_0(x) &= 0 \\
a_1 &= \frac{\sin\frac{1}{2}\pi - p_0(\frac{1}{2}\pi)}{(\frac{1}{2}\pi - 0)} = \frac{2}{\pi} & &\Rightarrow & p_1(x) &= \frac{2}{\pi}x \\
a_2 &= \frac{\sin\pi - p_1(\pi)}{(\pi - 0)(\pi - \frac{1}{2}\pi)} = -\frac{4}{\pi^2} & &\Rightarrow & p_2(x) &= \frac{2}{\pi}x - \frac{4}{\pi^2}x(x - \frac{\pi}{2}).
\end{aligned}$$

The reader should check that each interpolating polynomial does interpolate at the appropriate points. ■



$$p_{n+1}(x) = p_n(x) + a_{n+1}w_n(x),$$

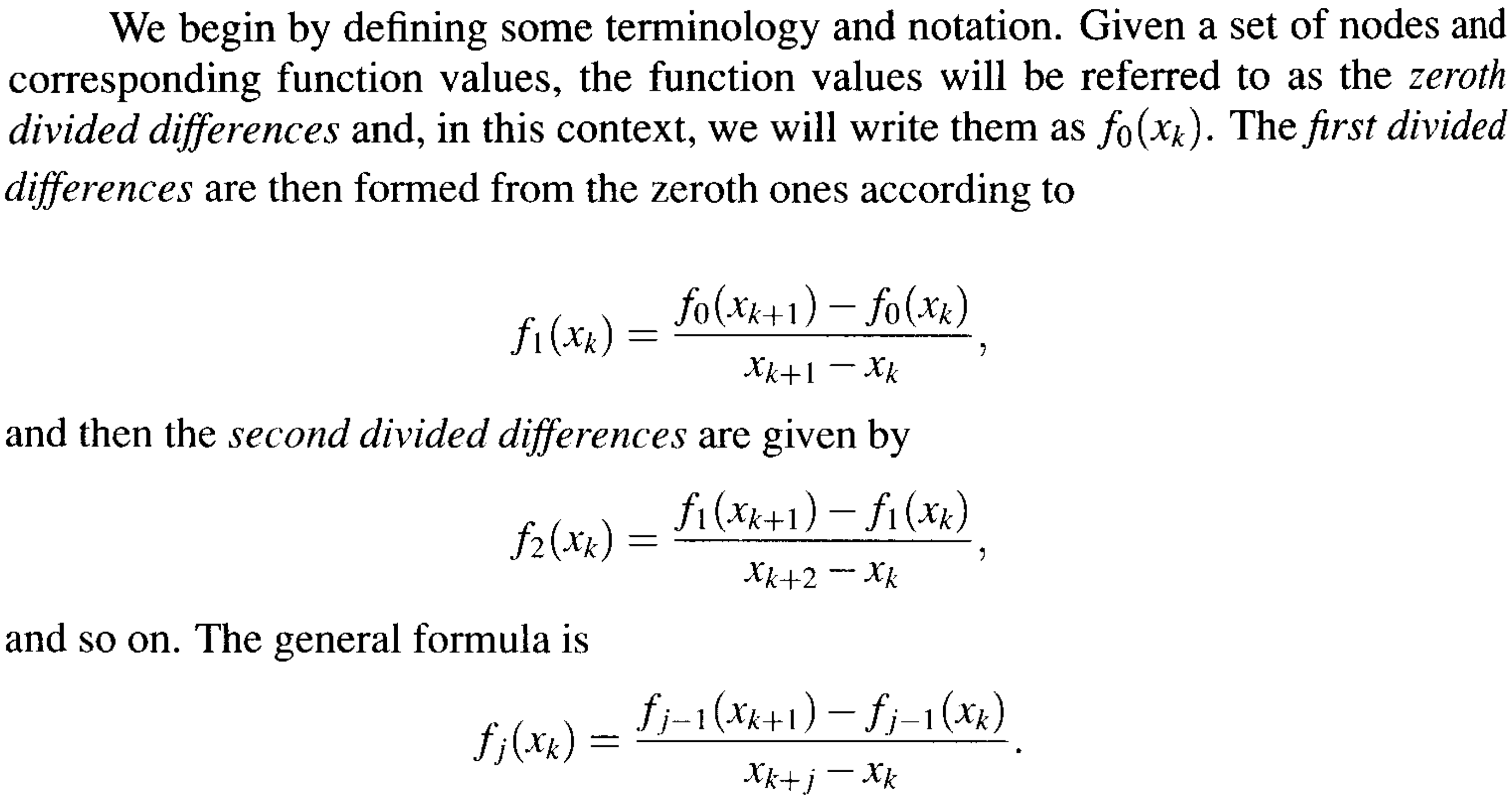


$$w_n(x) = \prod_{i=0}^{n}(x - x_i)$$

$$a_{n+1} = \frac{f(x_{n+1}) - p_n(x_{n+1})}{w_n(x_{n+1})}$$

Discussion

- Les coefficients a_k s'appellant differences divisées.
- On peut utiliser le tableau des differences divisées pour les trouver.



We begin by defining some terminology and notation. Given a set of nodes and corresponding function values, the function values will be referred to as the *zeroth divided differences* and, in this context, we will write them as $f_0(x_k)$. The *first divided differences* are then formed from the zeroth ones according to

$$f_1(x_k) = \frac{f_0(x_{k+1}) - f_0(x_k)}{x_{k+1} - x_k},$$

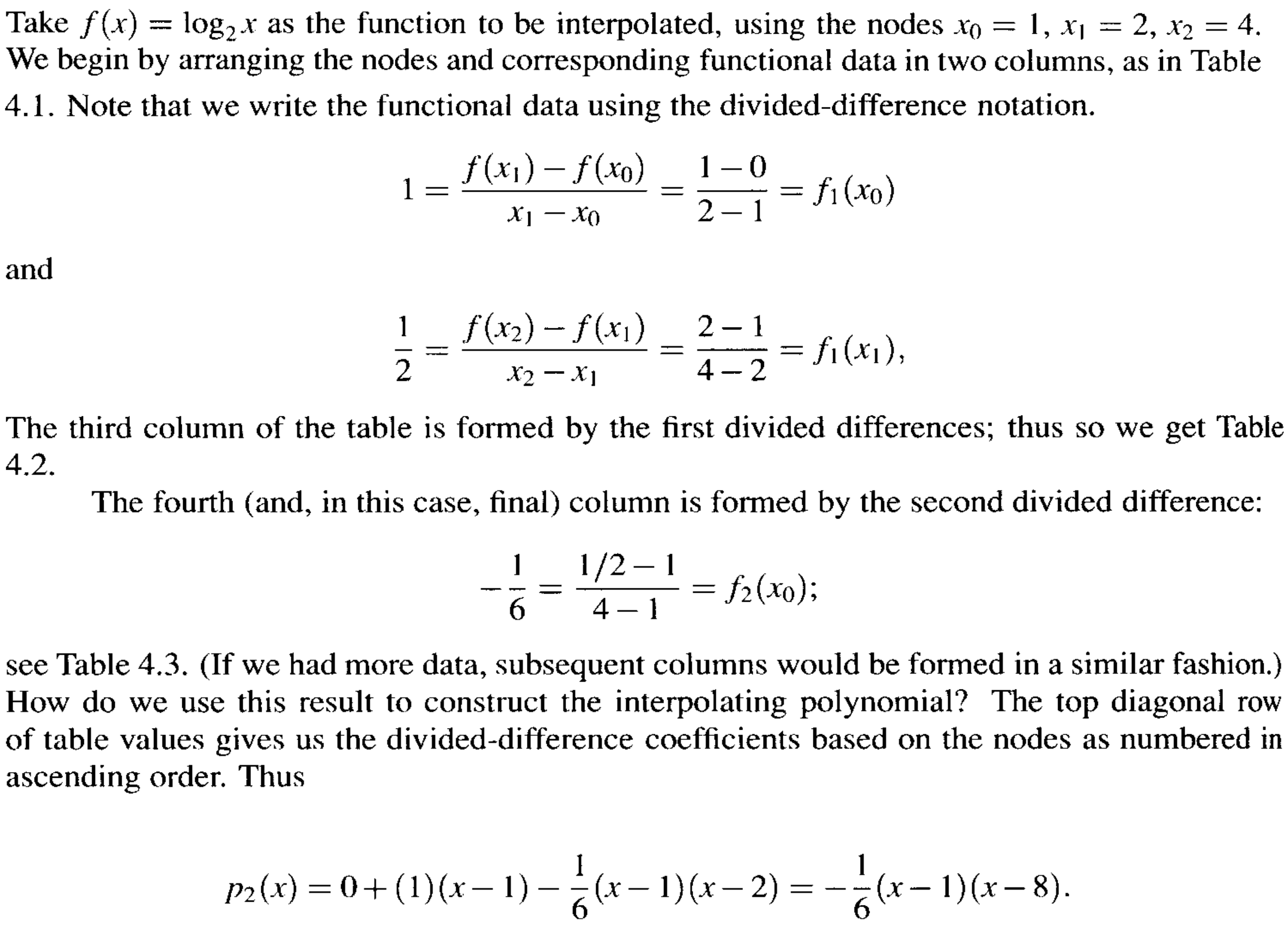
and then the *second divided differences* are given by

$$f_2(x_k) = \frac{f_1(x_{k+1}) - f_1(x_k)}{x_{k+2} - x_k},$$

and so on. The general formula is

$$f_j(x_k) = \frac{f_{j-1}(x_{k+1}) - f_{j-1}(x_k)}{x_{k+j} - x_k}.$$

Exemple



Take $f(x) = \log_2 x$ as the function to be interpolated, using the nodes $x_0 = 1$, $x_1 = 2$, $x_2 = 4$. We begin by arranging the nodes and corresponding functional data in two columns, as in Table 4.1. Note that we write the functional data using the divided-difference notation.

$$1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{1-0}{2-1} = f_1(x_0)$$

and

$$\frac{1}{2} = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{2-1}{4-2} = f_1(x_1),$$

The third column of the table is formed by the first divided differences; thus so we get Table 4.2.

The fourth (and, in this case, final) column is formed by the second divided difference:

$$-\frac{1}{6} = \frac{1/2 - 1}{4-1} = f_2(x_0);$$

see Table 4.3. (If we had more data, subsequent columns would be formed in a similar fashion.) How do we use this result to construct the interpolating polynomial? The top diagonal row of table values gives us the divided-difference coefficients based on the nodes as numbered in ascending order. Thus

$$p_2(x) = 0 + (1)(x-1) - \frac{1}{6}(x-1)(x-2) = -\frac{1}{6}(x-1)(x-8).$$

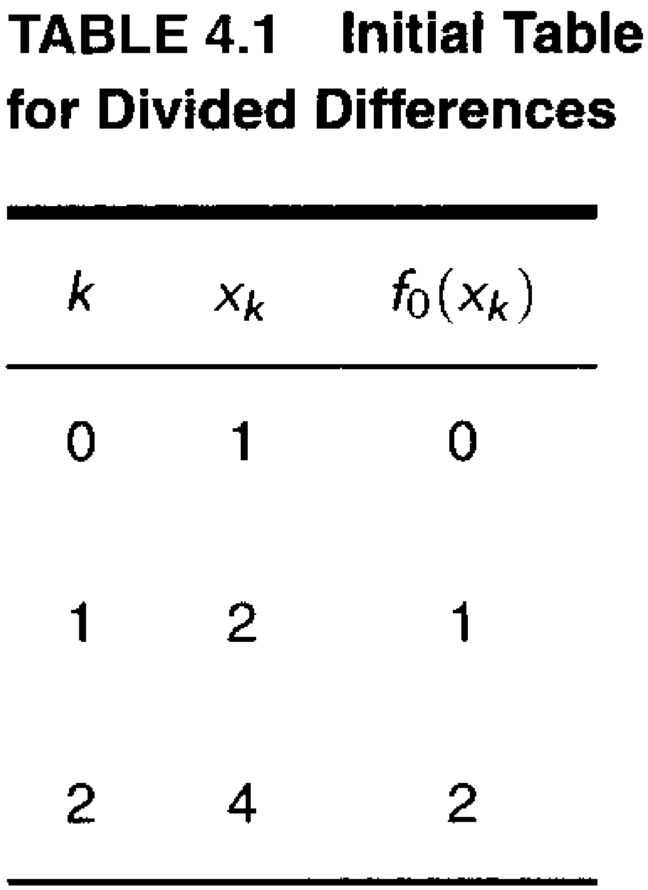


TABLE 4.1 Initial Table for Divided Differences

k	x_k	$f_0(x_k)$
0	1	0
1	2	1
2	4	2

TABLE 4.1 Initial Table for Divided Differences

k	x_k	$f_0(x_k)$
0	1	0
1	2	1
2	4	2

TABLE 4.2 Table for Divided Differences after Computation of First Differences

k	x_k	$f_0(x_k)$	$f_1(x_k)$
0	1	0	1
1	2	1	$\frac{1}{2}$
2	4	2	

TABLE 4.3 Table for Divided Differences after Computation of Second Differences

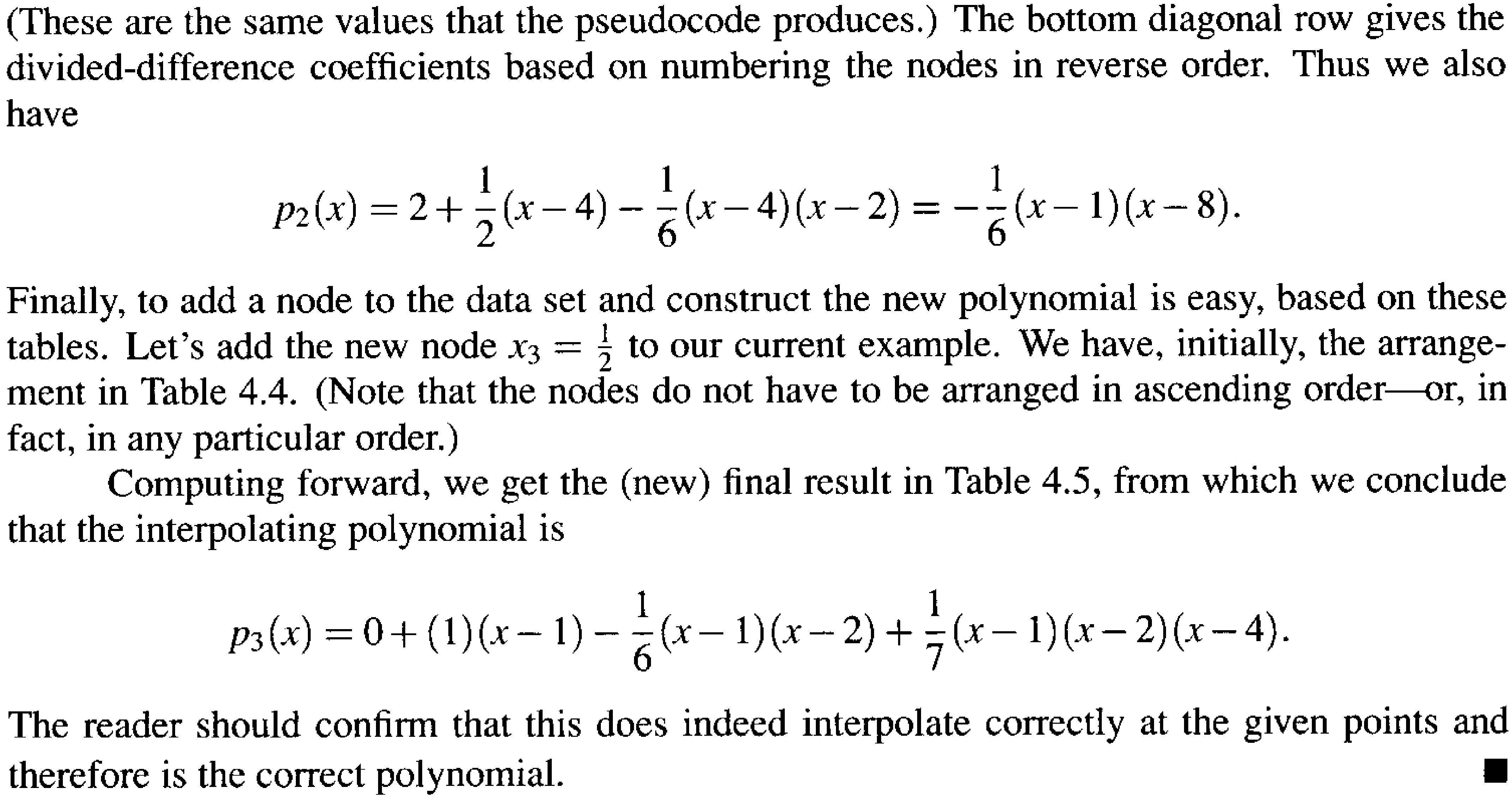
k	x_k	$f_0(x_k)$	$f_1(x_k)$	$f_2(x_k)$
0	1	0	1	$-\frac{1}{6}$
1	2	1	$\frac{1}{2}$	
2	4	2		

$$\boxed{f_j(x_k) = \frac{f_{j-1}(x_{k+1}) - f_{j-1}(x_k)}{x_{k+j} - x_k}.}$$

$$p_2(x) = 0 + (1)(x-1) - \frac{1}{6}(x-1)(x-2) = -\frac{1}{6}(x-1)(x-8)$$

$$p_2(x) = 2 + \frac{1}{2}(x-4) - \frac{1}{6}(x-4)(x-2) = -\frac{1}{6}(x-1)(x-8).$$

Exemple: (suite)



(These are the same values that the pseudocode produces.) The bottom diagonal row gives the divided-difference coefficients based on numbering the nodes in reverse order. Thus we also have

$$p_2(x) = 2 + \frac{1}{2}(x-4) - \frac{1}{6}(x-4)(x-2) = -\frac{1}{6}(x-1)(x-8).$$

Finally, to add a node to the data set and construct the new polynomial is easy, based on these tables. Let's add the new node $x_3 = \frac{1}{2}$ to our current example. We have, initially, the arrangement in Table 4.4. (Note that the nodes do not have to be arranged in ascending order—or, in fact, in any particular order.)

Computing forward, we get the (new) final result in Table 4.5, from which we conclude that the interpolating polynomial is

$$p_3(x) = 0 + (1)(x-1) - \frac{1}{6}(x-1)(x-2) + \frac{1}{7}(x-1)(x-2)(x-4).$$

The reader should confirm that this does indeed interpolate correctly at the given points and therefore is the correct polynomial. ■

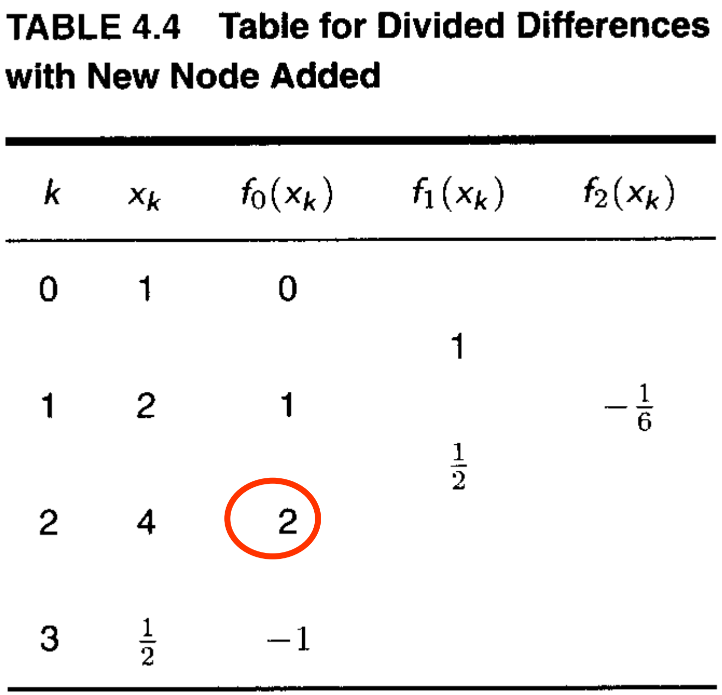


TABLE 4.4 Table for Divided Differences with New Node Added

k	x_k	$f_0(x_k)$	$f_1(x_k)$	$f_2(x_k)$
0	1	0		
			1	
1	2	1		$-\frac{1}{6}$
			$\frac{1}{2}$	
2	4	2		
3	$\frac{1}{2}$	-1		

Table 4.5

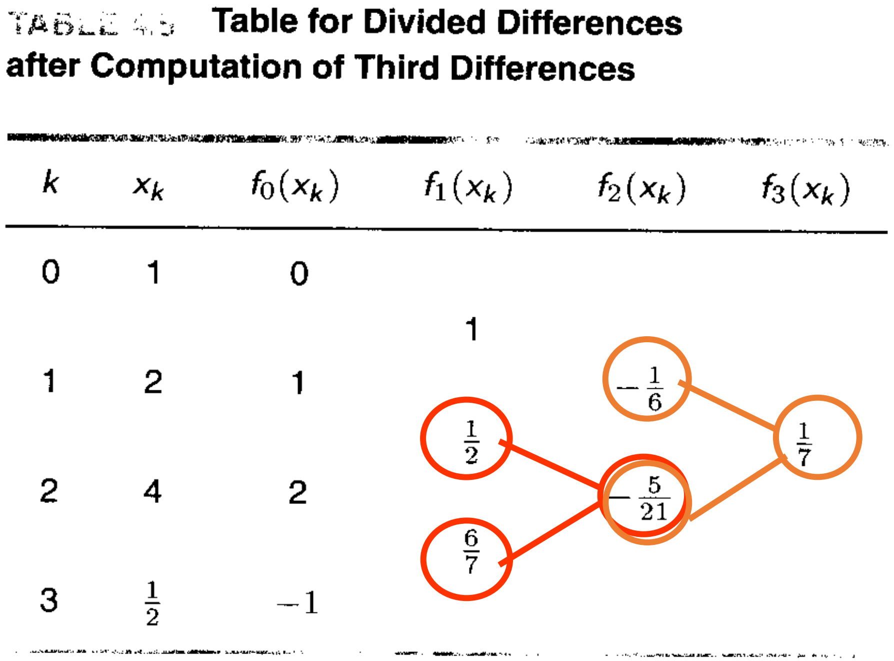


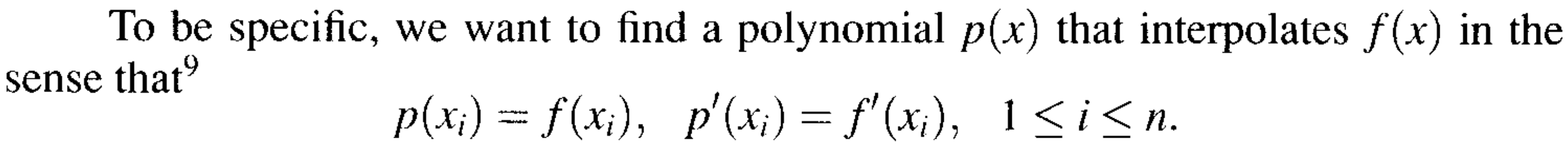
TABLE 4.5 Table for Divided Differences after Computation of Third Differences

k	x_k	$f_0(x_k)$	$f_1(x_k)$	$f_2(x_k)$	$f_3(x_k)$
0	1	0			
			1		
1	2	1		$-\frac{1}{6}$	
			$\frac{1}{2}$		$\frac{1}{7}$
2	4	2		$-\frac{5}{21}$	
			$\frac{6}{7}$		
3	$\frac{1}{2}$	-1			

$$p_3(x) = 0 + (1)(x-1) - \frac{1}{6}(x-1)(x-2) + \frac{1}{7}(x-1)(x-2)(x-4).$$

Interpolation d'Hermite

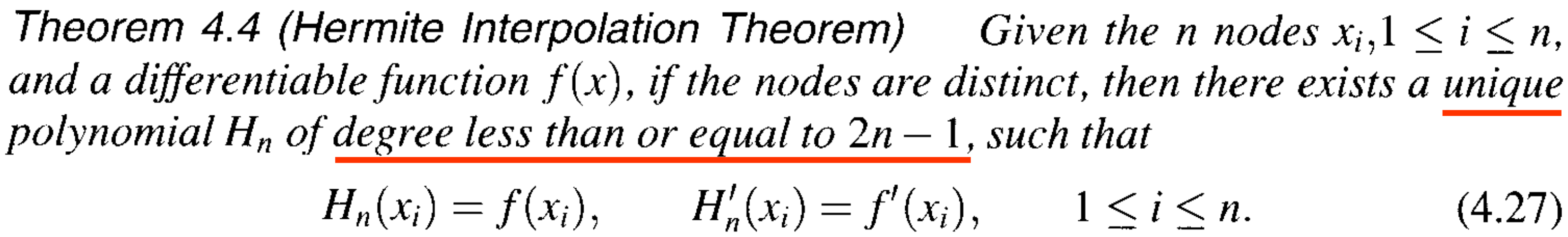
- Problème d'interpolation d'Hermite



To be specific, we want to find a polynomial $p(x)$ that interpolates $f(x)$ in the sense that[9]

$$p(x_i) = f(x_i), \quad p'(x_i) = f'(x_i), \quad 1 \leq i \leq n.$$

- Est ce possible ? oui.



Theorem 4.4 (Hermite Interpolation Theorem) *Given the n nodes $x_i, 1 \leq i \leq n$, and a differentiable function $f(x)$, if the nodes are distinct, then there exists a unique polynomial H_n of degree less than or equal to $2n - 1$, such that*

$$H_n(x_i) = f(x_i), \qquad H_n'(x_i) = f'(x_i), \qquad 1 \leq i \leq n. \qquad (4.27)$$

Proof This is basically the same as the proof of Theorem 4.1. We define the two families of polynomials

$$h_k(x) = [1 - 2[L_k^{(n)}]'(x_k)(x - x_k)][L_k^{(n)}(x)]^2$$

and

$$\tilde{h}_k(x) = (x - x_k)[L_k^{(n)}(x)]^2;$$

now observe that

$$h_k(x_j) = \delta_{kj}, \qquad h_k'(x_j) = 0,$$

and

$$\tilde{h}_k(x_j) = 0, \qquad \tilde{h}_k'(x_j) = \delta_{kj}.$$

Therefore, the polynomial

$$H_n(x) = \sum_{k=1}^{n} \left(f(x_k)h_k(x) + f'(x_k)\tilde{h}_k(x)\right)$$

satisfies the interpolating conditions (4.27) and is clearly of degree less than or equal to $2n - 1$. Uniqueness follows by the same argument used before. □

Tableau des diff. divisées

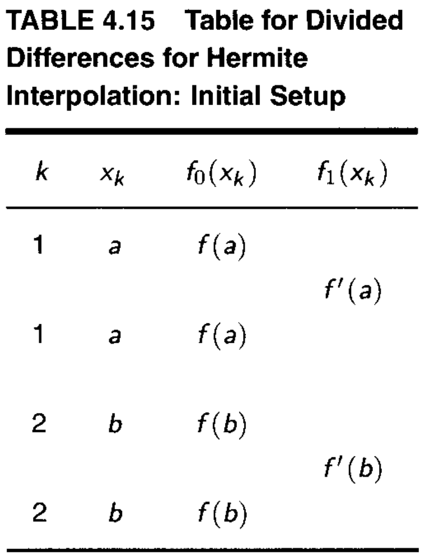


TABLE 4.15 Table for Divided Differences for Hermite Interpolation: Initial Setup

k	x_k	$f_0(x_k)$	$f_1(x_k)$
1	a	$f(a)$	
			$f'(a)$
1	a	$f(a)$	
2	b	$f(b)$	
			$f'(b)$
2	b	$f(b)$	

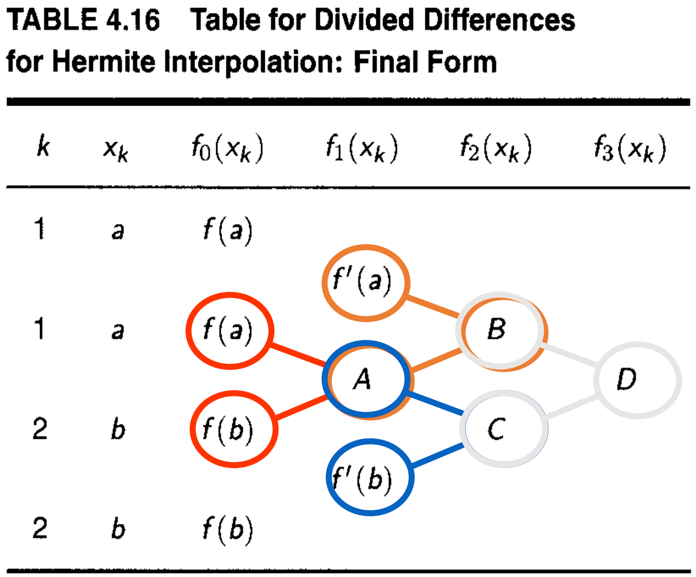


TABLE 4.16 Table for Divided Differences for Hermite Interpolation: Final Form

k	x_k	$f_0(x_k)$	$f_1(x_k)$	$f_2(x_k)$	$f_3(x_k)$
1	a	$f(a)$			
			$f'(a)$		
1	a	$f(a)$		B	
			A		D
2	b	$f(b)$		C	
			$f'(b)$		
2	b	$f(b)$			

$$A = \frac{f(b) - f(a)}{b - a}$$

$$B = \frac{A - f'(a)}{b - a}$$

$$C = \frac{f'(b) - A}{b - a}$$

$$D = \frac{C - B}{b - a}.$$

$$H_2(x) = \underline{f(a)} + \underline{f'(a)}(x-a) + \underline{B}(x-a)^2 + \underline{D}(x-a)^2(x-b).$$

Exemple

$$x_1 = -1, x_2 = 1,$$

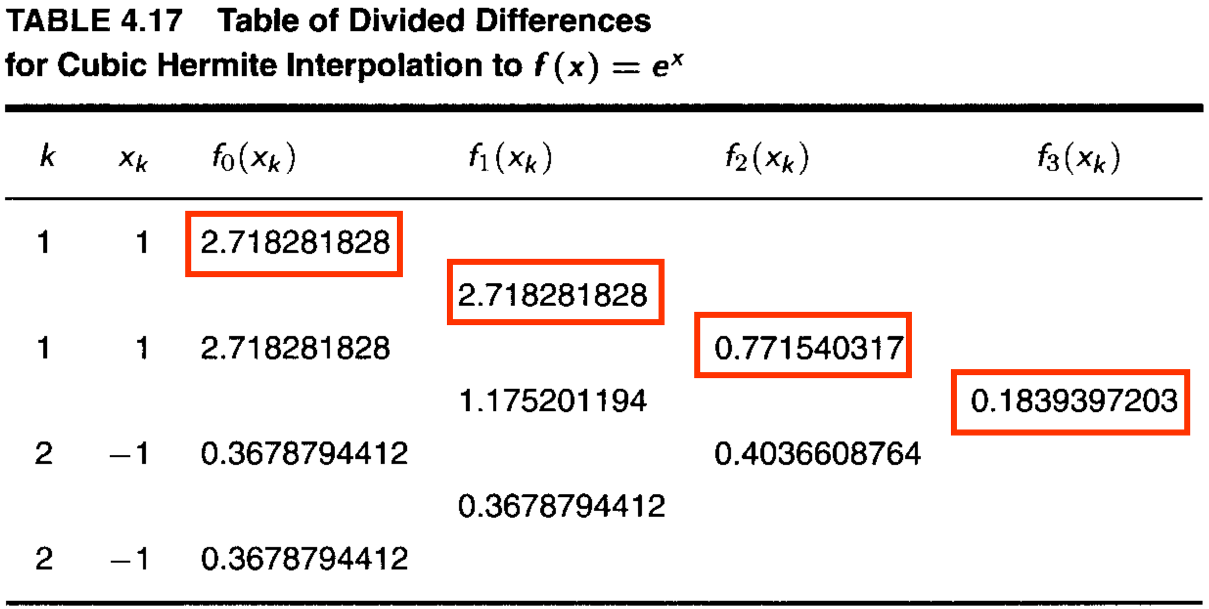


TABLE 4.17 Table of Divided Differences for Cubic Hermite Interpolation to $f(x) = e^x$

k	x_k	$f_0(x_k)$	$f_1(x_k)$	$f_2(x_k)$	$f_3(x_k)$
1	1	2.718281828			
			2.718281828		
1	1	2.718281828		0.771540317	
			1.175201194		0.1839397203
2	−1	0.3678794412		0.4036608764	
			0.3678794412		
2	−1	0.3678794412			

$$H_2(x) = 2.718281828 + 2.718281828(x-1) + 0.771540317(x-1)^2 \\ + 0.1839397203(x-1)^2(x+1),$$

$$H_2(x) = 0.1839397206x^3 + 0.5876005967x^2 + 0.9912614728x + 0.9554800379.$$

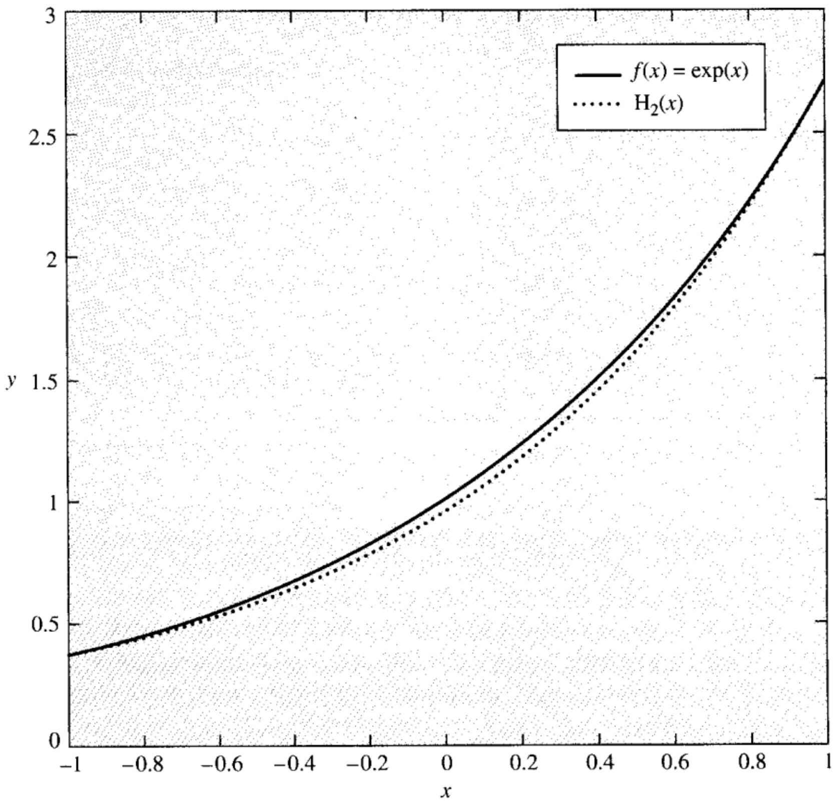


FIGURE 4.6 Hermite interpolation to $f(x) = e^x$.

Erreur

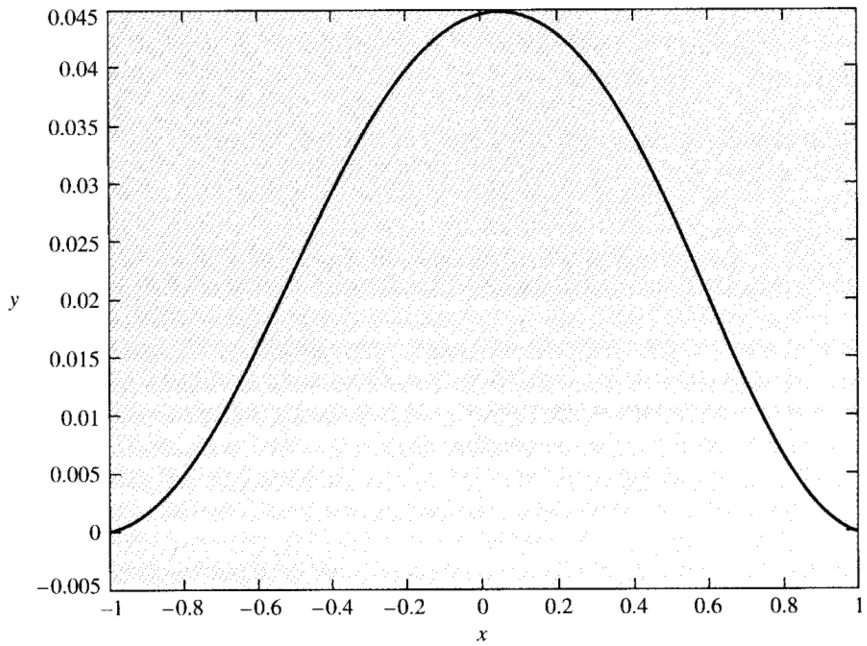
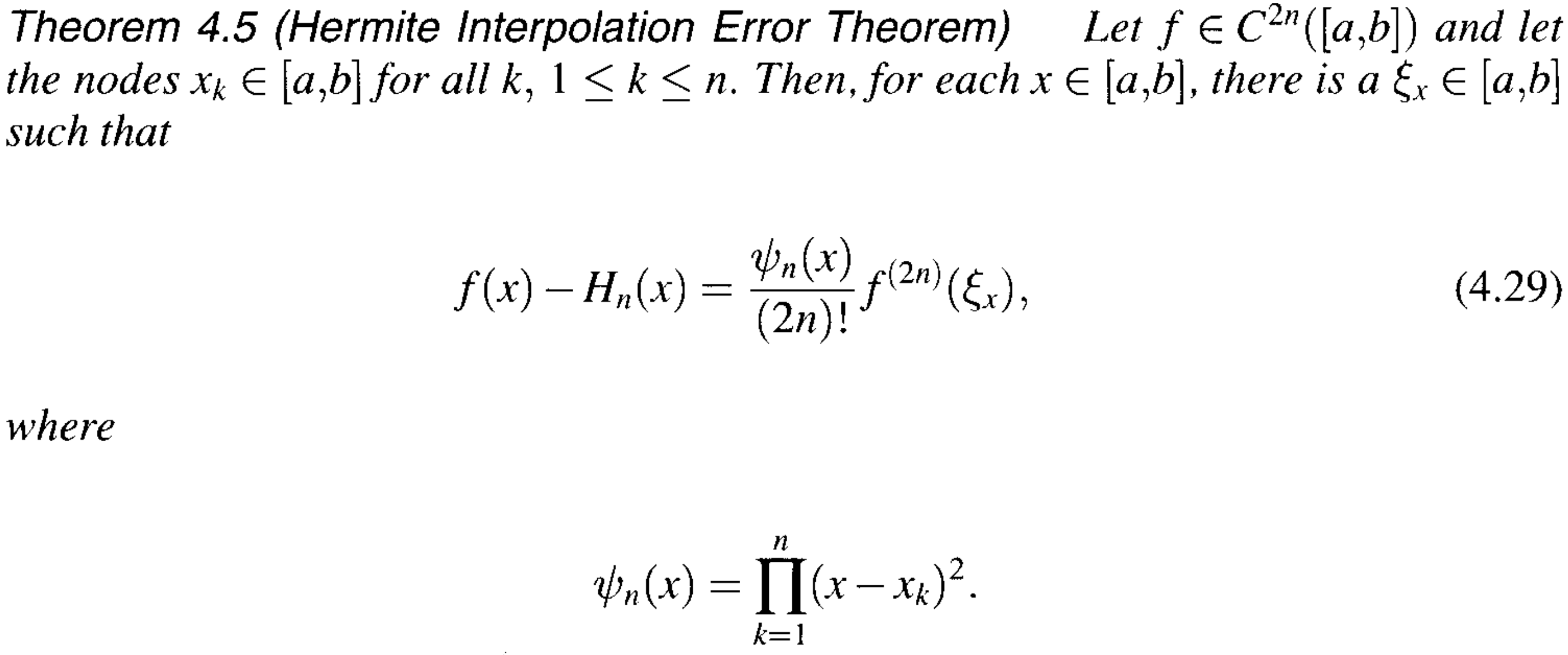


FIGURE 4.7 Error in Hermite interpolation to $f(x) = e^x$.

Thm. d’erreur d’interpolation d’Hermite



Theorem 4.5 (Hermite Interpolation Error Theorem) *Let $f \in C^{2n}([a,b])$ and let the nodes $x_k \in [a,b]$ for all k, $1 \leq k \leq n$. Then, for each $x \in [a,b]$, there is a $\xi_x \in [a,b]$ such that*

$$f(x) - H_n(x) = \frac{\psi_n(x)}{(2n)!} f^{(2n)}(\xi_x), \tag{4.29}$$

where

$$\psi_n(x) = \prod_{k=1}^{n} (x - x_k)^2.$$

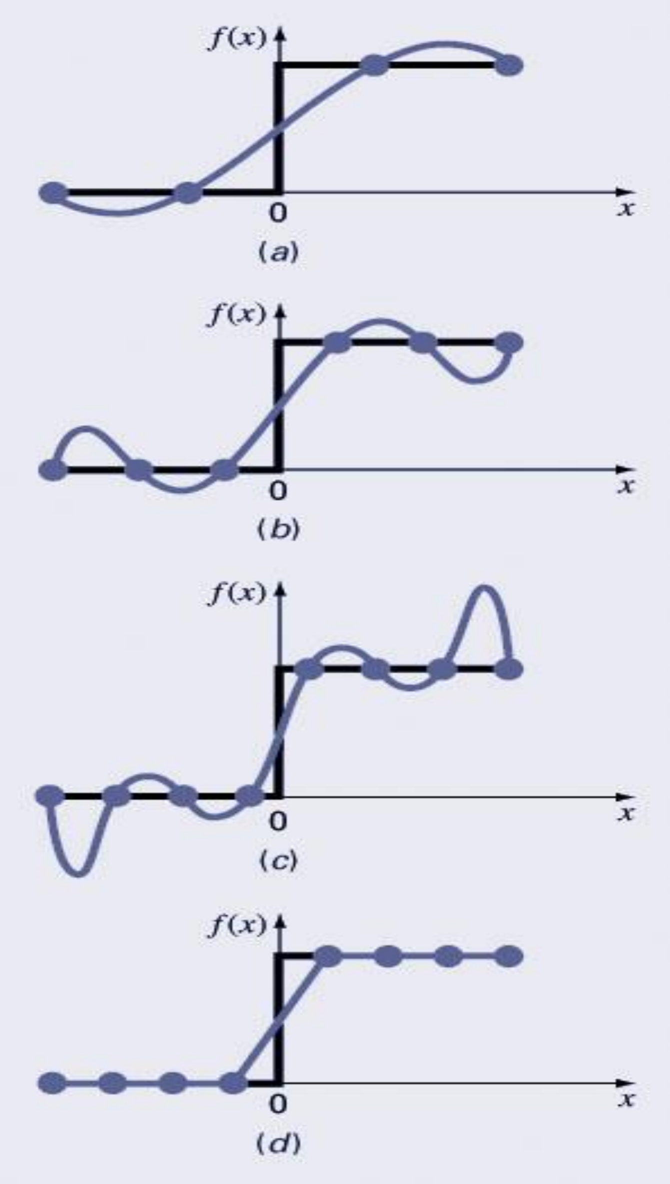
Cubic Spline

Introduction to Splines

- An alternative approach to using a single (n-1)th order polynomial to interpolate between n points is to apply lower-order polynomials in a piecewise fashion to subsets of data points.
- These connecting polynomials are called *spline functions*.
- Splines minimize oscillations and reduce round-off error due to their lower-order nature.

Higher Order vs. Splines

- Splines eliminate oscillations by using small subsets of points for each interval rather than every point. This is especially useful when there are jumps in the data:
  a) 3rd order polynomial
  b) 5th order polynomial
  c) 7th order polynomial
  d) Linear spline
    - seven 1st order polynomials generated by using pairs of points at a time



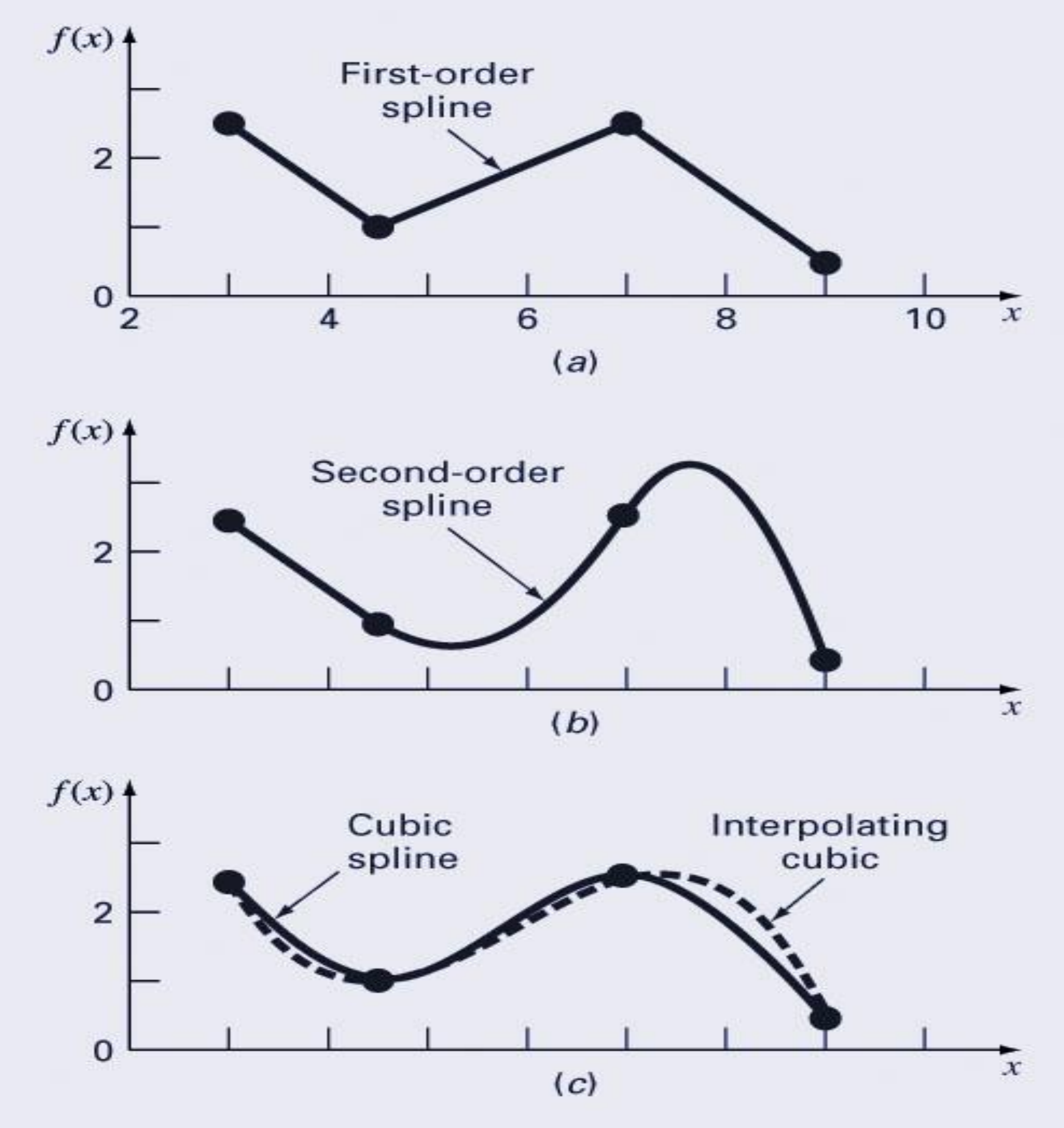
Spline Development

a) First-order splines find straight-line equations between each pair of points that
   - Go through the points

b) Second-order splines find quadratic equations between each pair of points that
   - Go through the points
   - Match first derivatives at the interior points

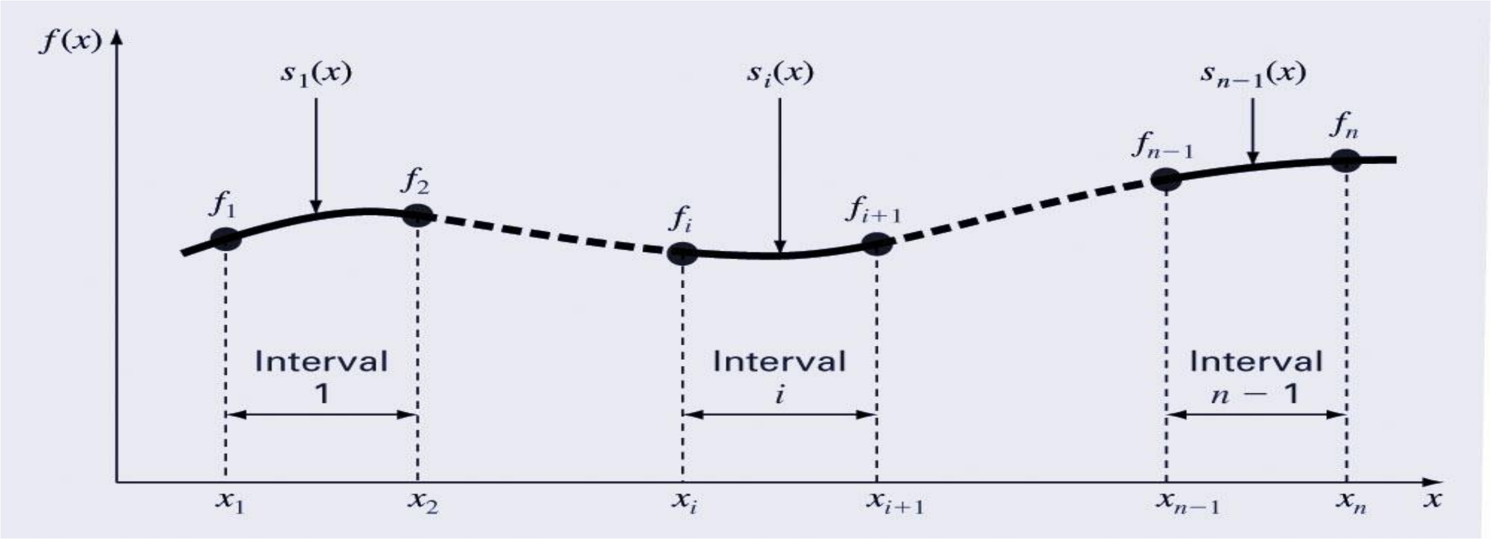
c) Third-order splines find cubic equations between each pair of points that
   - Go through the points
   - Match first and second derivatives at the interior points

     Note that the results of cubic spline interpolation are different from the results of an interpolating cubic.



Spline Development

- Spline function ($s_i(x)$)coefficients are calculated for each interval of a data set.
- The number of data points (f_i) used for each spline function depends on the order of the spline function.



Cubic Splines

- While data of a particular size presents many options for the order of spline functions, cubic splines are preferred because they provide the simplest representation that exhibits the desired appearance of smoothness.
- In general, the *i*th spline function for a cubic spline can be written as:

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

- For *n* data points, there are *n*-1 intervals and thus 4(*n*-1) unknowns to evaluate to solve all the spline function coefficients.
- There is no ‘one equation’ that can represent the whole spline function on the domain

Why Splines ?

$$f(x) = \frac{1}{1+25x^2}$$

Table : Six equidistantly spaced points in [-1, 1]

x	$y = \frac{1}{1+25x^2}$
-1.0	0.038461
-0.6	0.1
-0.2	0.5
0.2	0.5
0.6	0.1
1.0	0.038461

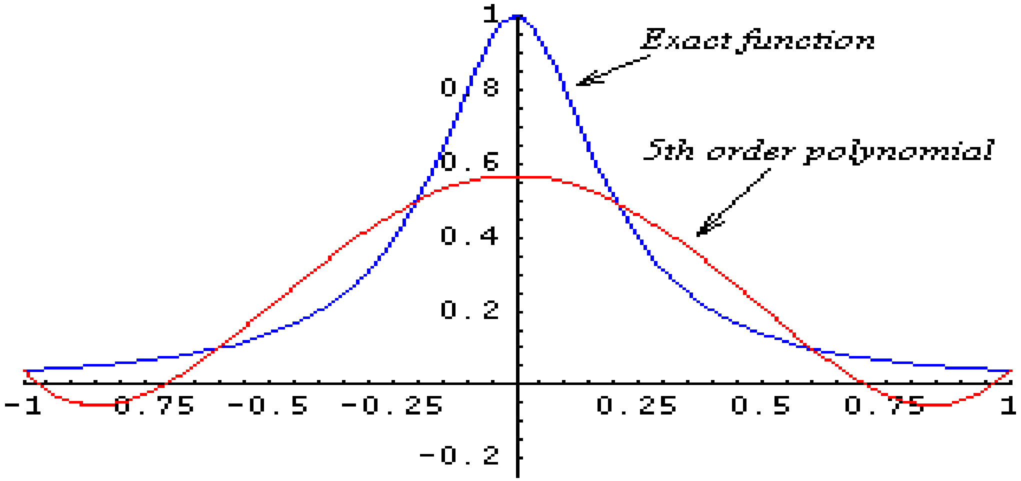


Figure : 5th order polynomial vs. exact function

Why Splines ?

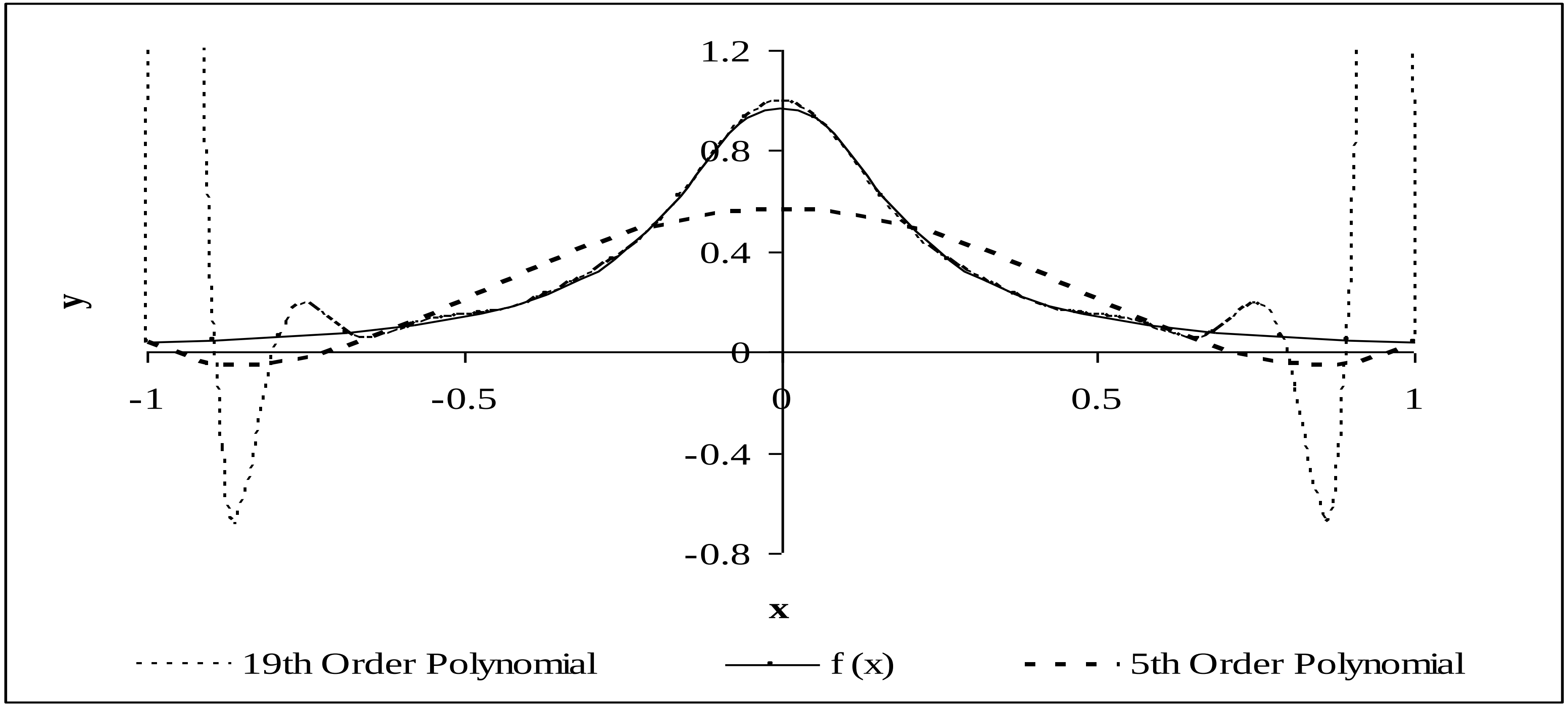
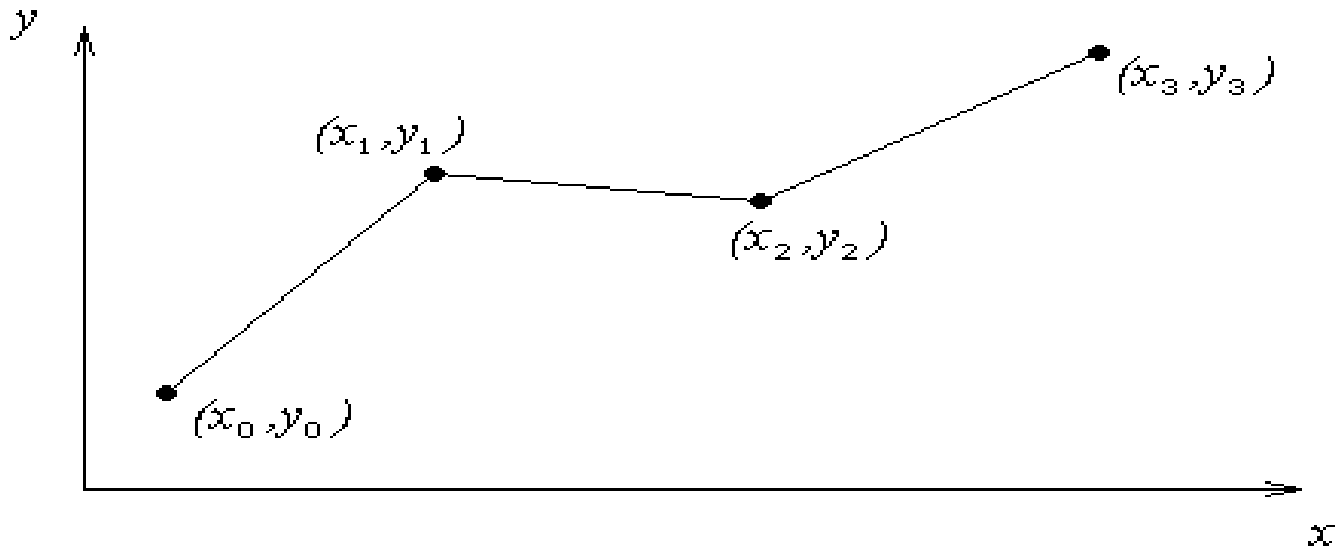


Figure : Higher order polynomial interpolation is a bad idea

Linear Interpolation

Given $(x_0, y_0),(x_1, y_1),......,(x_{n-1}, y_{n-1})(x_n, y_n)$, fit linear splines to the data. This simply involves forming the consecutive data through straight lines. So if the above data is given in an ascending order, the linear splines are given by $(y_i = f(x_i))$

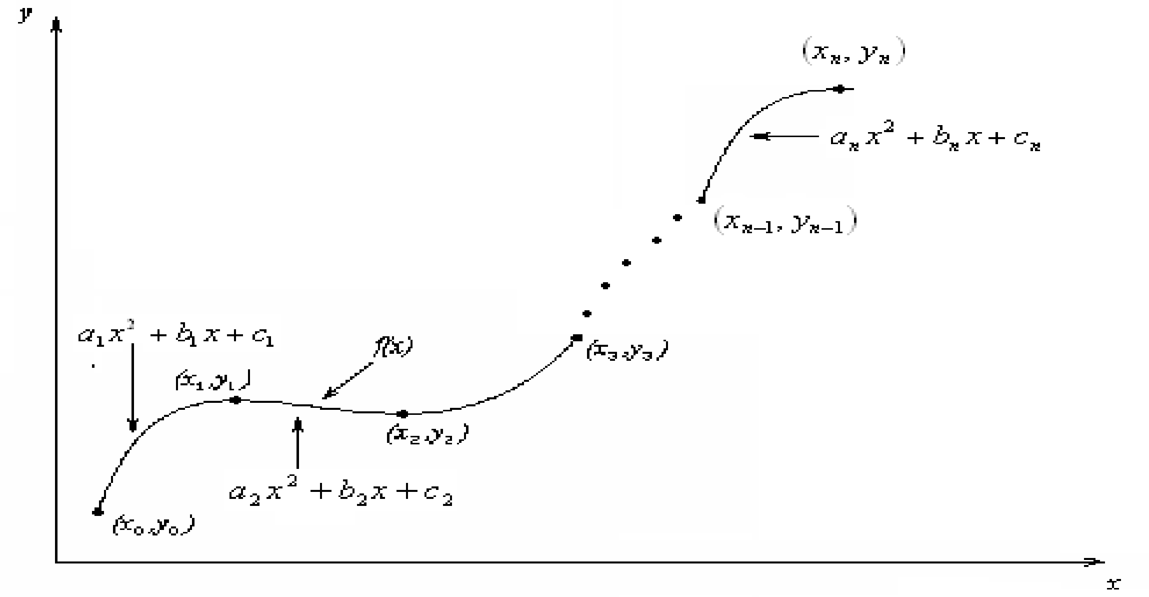
Figure : Linear splines



Quadratic Interpolation

Given $(x_0, y_0), (x_1, y_1), \ldots\ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)$, fit quadratic splines through the data. The splines are given by

$$
\begin{aligned}
f(x) &= a_1 x^2 + b_1 x + c_1, && x_0 \le x \le x_1 \\
&= a_2 x^2 + b_2 x + c_2, && x_1 \le x \le x_2 \\
&\;. \\
&\;. \\
&\;. \\
&= a_n x^2 + b_n x + c_n, && x_{n-1} \le x \le x_n
\end{aligned}
$$

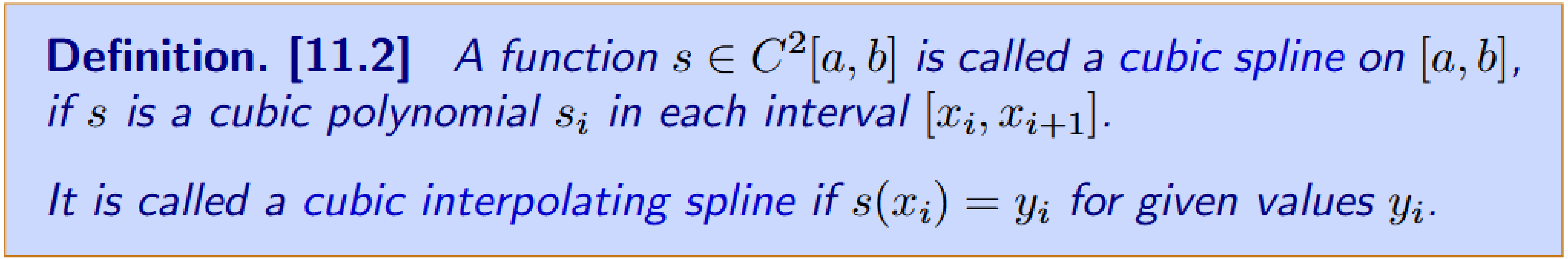


Find a_i, b_i, c_i, $i = 1, 2, \ldots, \mathrm{n}$

Cubic Splines

Let $K = \{x_0, \dots, x_m\}$ be a set of given knots with

$$a = x_0 < x_1 < \cdots < x_m = b$$

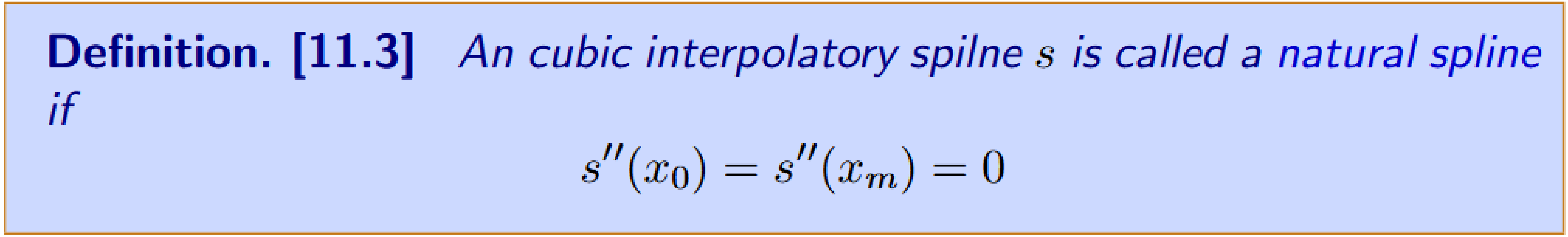


Definition. [11.2] *A function $s \in C^2[a,b]$ is called a cubic spline on $[a,b]$, if s is a cubic polynomial s_i in each interval $[x_i, x_{i+1}]$.*

It is called a cubic interpolating spline if $s(x_i) = y_i$ for given values y_i.

Cubic Splines

Interpolating cubic splines need two additional conditions to be uniquely defined



Definition. [11.3] *An cubic interpolatory spilne s is called a natural spline if*

$$s''(x_0) = s''(x_m) = 0$$

Cubic Splines - Construction

We construct an interpolating in a different but equivalent way than in the textbook:

Ansatz for m the piecewise polynomials

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

By fixing the $4m$ free coefficients $a_i, b_i, c_i, d_i, i = 0 : m - 1$ the entire spline is fixed.

Cubic Splines - Construction

We need $4m$ conditions to fix the coefficients

(1) $s_i(x_i) = y_i,$ for $i = 0 : m - 1,$

(2) $s_{m-1} = y_m,$ 1 condition

(3) $s_i(x_{i+1}) = s_{i+1}(x_{i+1}),$ for $i = 0 : m - 2,$

(4) $s_i'(x_{i+1}) = s_{i+1}'(x_{i+1}),$ for $i = 0 : m - 2,$

(5) $s_i''(x_{i+1}) = s_{i+1}''(x_{i+1}),$ for $i = 0 : m - 2,$

These are $4m - 2$ conditions. We need two extra.

Cubic Splines – Boundary conditions

We can define two extra boundary conditions. One has several alternatives:

Natural Spline $s_0''(x_0) = 0$ and $s_{m-1}''(x_m) = 0$

End Slope Spline $s_0'(x_0) = y_0'$ and $s_{m-1}'(x_m) = y_m'$

Periodic Spline $s_0'(x_0) = s_{m-1}'(x_m)$ and $s_0''(x_0) = s_{m-1}''(x_m)$

Not-a-Knot Spline $s_0'''(x_1) = s_1'''(x_1)$ and $s_{m-2}'''(x_{m-1}) = s_{m-1}'''(x_{m-1})$

We consider here natural splines.

Natrul Splines – Construction

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

Let $h = x_{i+1} - x_i$ equidistant spacing

$$\begin{aligned} s_i'(t_{i+1}) &= 3a_ih^2 + 2b_ih + c_i \\ s_i''(t_{i+1}) &= 6a_ih + 2b_i \end{aligned}$$

From Condition (1) we get $\boxed{d_i = y_i}$.

We introduce new variables for the second derivatives at x_i, i.e.

$$\sigma_i := s''(x_i) = 6a_i(x_i - x_i) + 2b_i = 2b_i \quad i = 0 : m$$

Natrul Splines – Construction

Thus $\boxed{b_i = \frac{\sigma_i}{2}}$.

From

$$\sigma_{i+1} = 6a_i h + 2b_i.$$

and Condition (5) we get

$$\boxed{a_i = \frac{\sigma_{i+1} - \sigma_i}{6h}}.$$

By Condition (3) we get

$$y_{i+1} = a_i h^3 + b_i h^2 + c_i h + y_i.$$

Natrul Splines – Construction

… and after inserting the hight-lighted expressions for a_i and b_i we get

$$y_{i+1} = (\frac{\sigma_{i+1} - \sigma_i}{6h})h^3 + \frac{\sigma_i}{2}h^2 + c_i h + y_i.$$

From that we get c_i: $\boxed{c_i = \frac{y_{i+1}-y_i}{h} - h\frac{2\sigma_i+\sigma_{i+1}}{6}}$.

Using now Condition (4) gives a relation between c_i and c_{i+1}

$$c_{i+1} = 3a_i h^2 + 2b_i h + c_i.$$

Natrul Splines – Construction

Inserting now the expressions for a_i, b_i and c_i, using Condition (2) and simplifying finally gives the central recursion formula:

$$\sigma_{i-1} + 4\sigma_i + \sigma_{i+1} = 6(\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2})$$

with $i = 1, \ldots, m-1.$

We consider now natural boundary conditions

$$\sigma_0 = \sigma_m = 0.$$

Natrul Splines – Construction

Finally we rewrite this all a system of linear equations

$$\begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & \ddots & 1 & \\ & & & \ddots & 1 \\ & & & 1 & 4 \end{pmatrix} \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \vdots \\ \sigma_{m-1} \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_2 - 2y_1 + y_0 \\ y_3 - 2y_2 + y_1 \\ \vdots \\ \vdots \\ y_n - 2y_{n-1} + y_{n-2} \end{pmatrix}$$

First, this system is solved and then the coefficients a_i, b_i, c_i, d_i are determined by the high-lighted equations.

Least square method

Ajustement des données par la méthode des moindres carrées

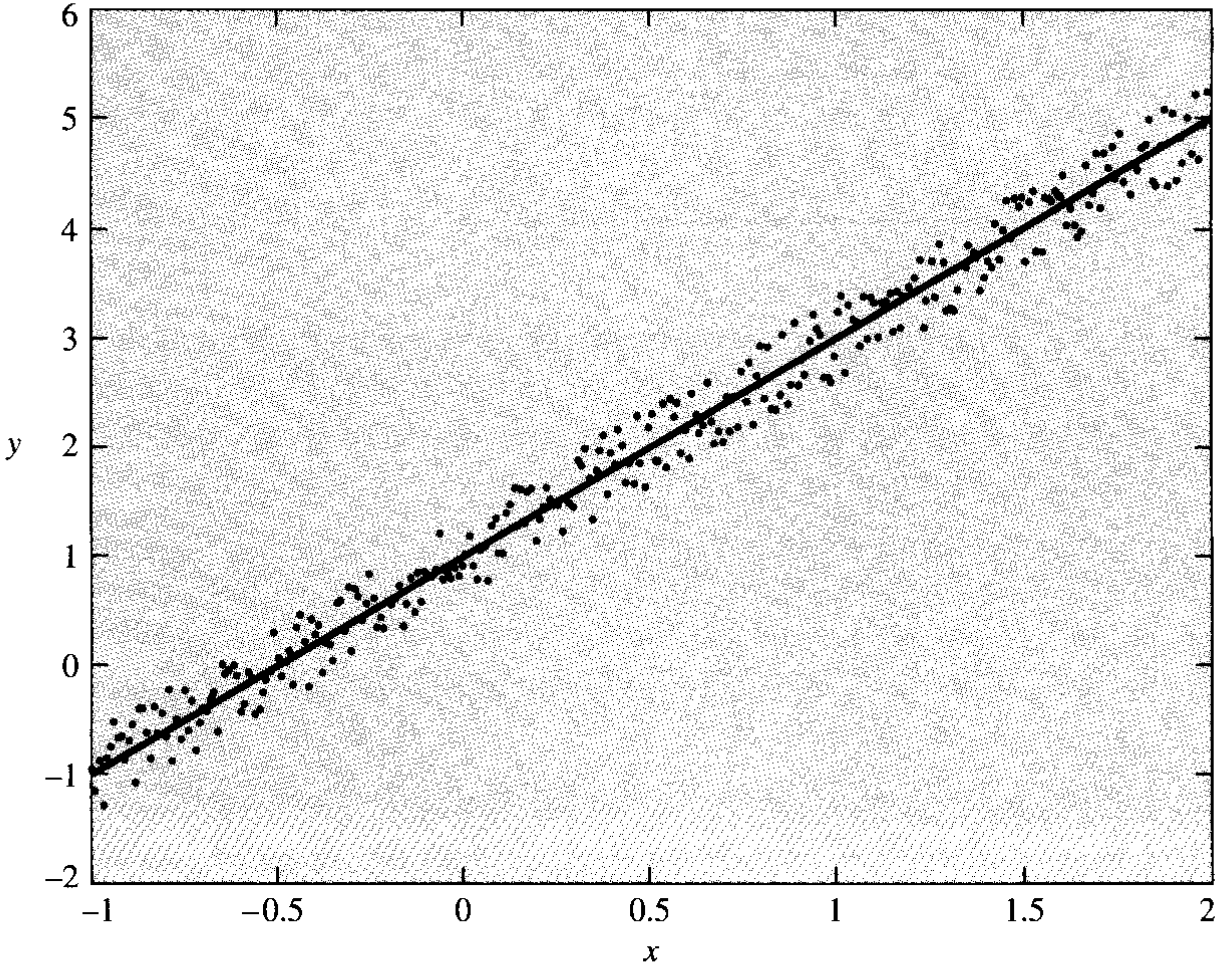
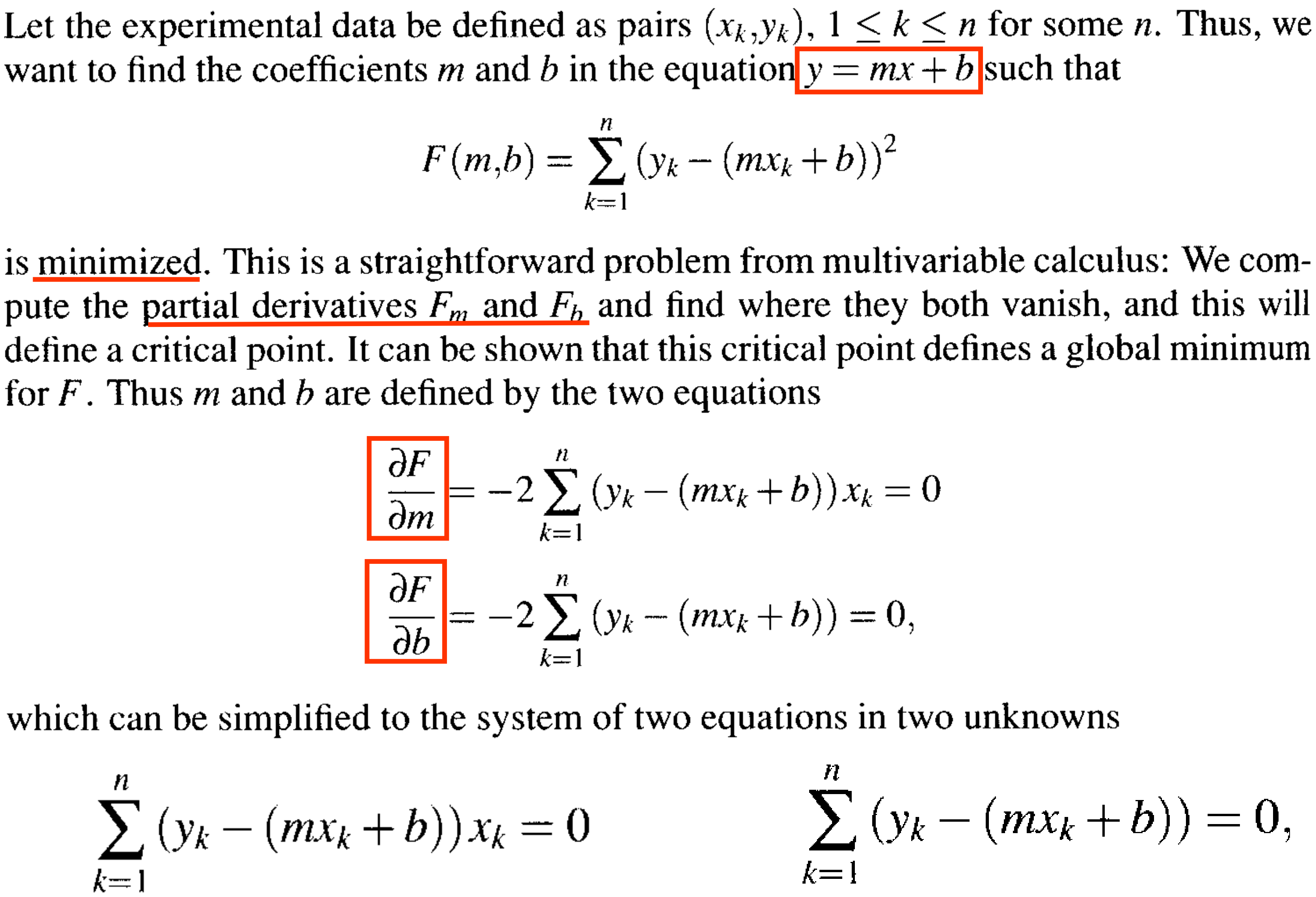


FIGURE 4.24 Straight-line fit to the data.



Let the experimental data be defined as pairs (x_k, y_k), $1 \leq k \leq n$ for some n. Thus, we want to find the coefficients m and b in the equation $\boxed{y = mx + b}$ such that

$$F(m,b) = \sum_{k=1}^{n} (y_k - (mx_k + b))^2$$

is minimized. This is a straightforward problem from multivariable calculus: We compute the partial derivatives F_m and F_b and find where they both vanish, and this will define a critical point. It can be shown that this critical point defines a global minimum for F. Thus m and b are defined by the two equations

$$\boxed{\frac{\partial F}{\partial m}} = -2 \sum_{k=1}^{n} (y_k - (mx_k + b))x_k = 0$$

$$\boxed{\frac{\partial F}{\partial b}} = -2 \sum_{k=1}^{n} (y_k - (mx_k + b)) = 0,$$

which can be simplified to the system of two equations in two unknowns

$$\sum_{k=1}^{n} (y_k - (mx_k + b))x_k = 0 \qquad \sum_{k=1}^{n} (y_k - (mx_k + b)) = 0,$$

$$\sum_{k=1}^{n} (y_k - (mx_k + b))x_k = 0$$

$$\sum_{k=1}^{n} (y_k - (mx_k + b)) = 0,$$

⇒

$$m\left(\sum_{k=1}^{n} x_k^2\right) + b\left(\sum_{k=1}^{n} x_k\right) = \sum_{k=1}^{n} x_k y_k$$

$$m\left(\sum_{k=1}^{n} x_k\right) + b\left(\sum_{k=1}^{n} 1\right) = \sum_{k=1}^{n} y_k.$$

The solution is then

⇒

$$m = \frac{n\sum_{i=1}^{n} x_k y_k - \left(\sum_{i=1}^{n} x_k\right)\left(\sum_{i=1}^{n} y_k\right)}{n\sum_{k=1}^{n} x_k^2 - \left(\sum_{i=1}^{n} x_k\right)^2}$$

$$b = \frac{\left(\sum_{i=1}^{n} x_k^2\right)\left(\sum_{i=1}^{n} y_k\right) - \left(\sum_{i=1}^{n} x_k\right)\left(\sum_{i=1}^{n} x_k y_k\right)}{n\left(\sum_{k=1}^{n} x_k^2\right) - \left(\sum_{i=1}^{n} x_k\right)^2},$$

If the data matrix X contains only two variables, a constant and a scalar regressor x_i, then this is called the "simple regression model".[12] This case is often considered in the beginner statistics classes, as it provides much simpler formulas even suitable for manual calculation. The parameters are commonly denoted as (α, β):

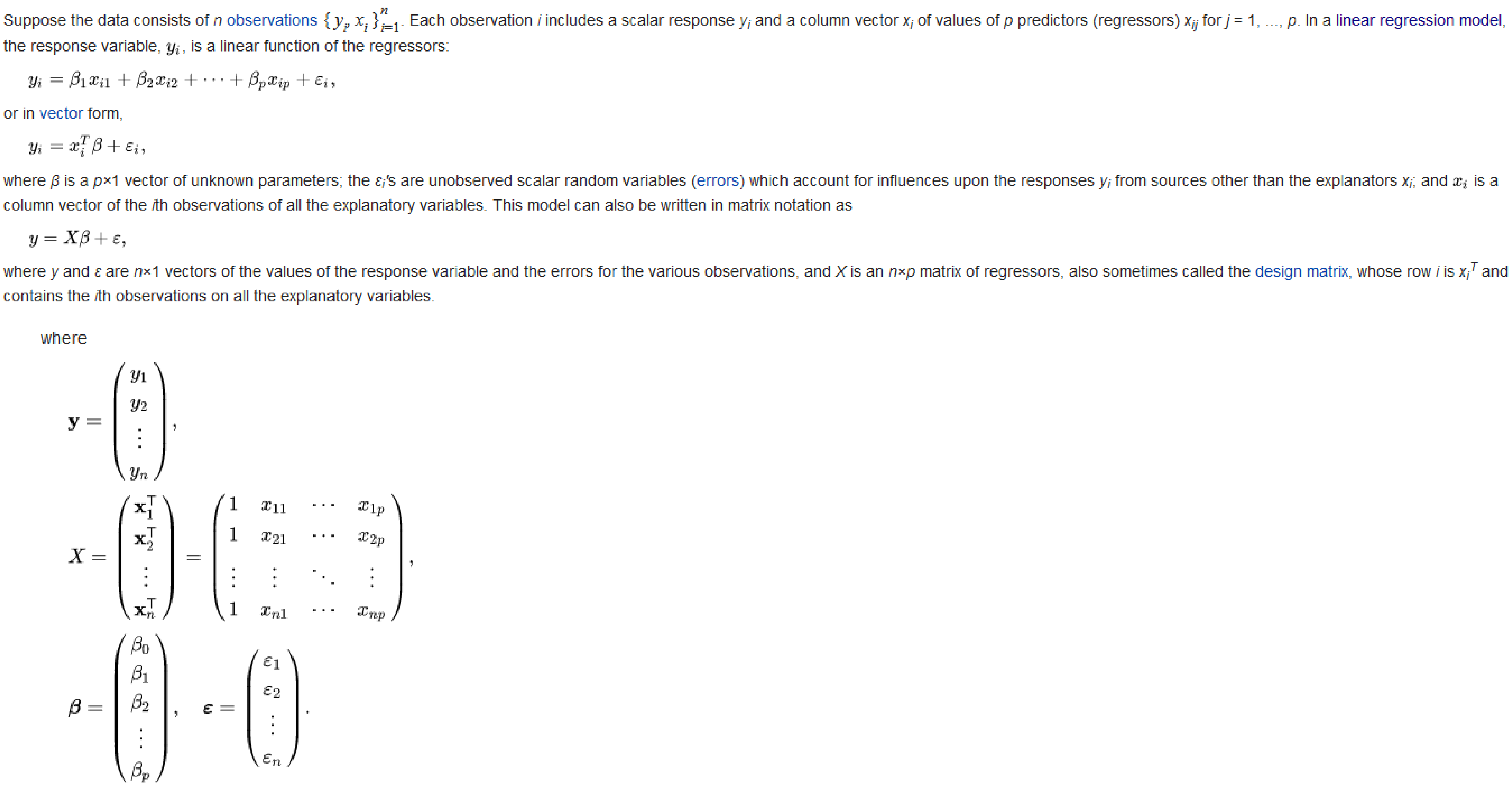
$$y_i = \alpha + \beta x_i + \varepsilon_i.$$

The least squares estimates in this case are given by simple formulas

$$\hat\beta = \frac{\sum x_i y_i - \frac{1}{n}\sum x_i \sum y_i}{\sum x_i^2 - \frac{1}{n}(\sum x_i)^2} = \frac{\operatorname{Cov}[x,y]}{\operatorname{Var}[x]}$$

$$\hat\alpha = \overline{y} - \hat\beta\,\overline{x}\ ,$$

where Var(.) and Cov(.) are sample parameters.



Suppose the data consists of n observations $\{y_i, x_i\}_{i=1}^{n}$. Each observation i includes a scalar response y_i and a column vector x_i of values of p predictors (regressors) x_{ij} for $j = 1, ..., p$. In a linear regression model, the response variable, y_i, is a linear function of the regressors:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i,$$

or in vector form,

$$y_i = x_i^T \beta + \varepsilon_i,$$

where β is a p×1 vector of unknown parameters; the ε_i's are unobserved scalar random variables (errors) which account for influences upon the responses y_i from sources other than the explanators x_i; and x_i is a column vector of the ith observations of all the explanatory variables. This model can also be written in matrix notation as

$$y = X\beta + \varepsilon,$$

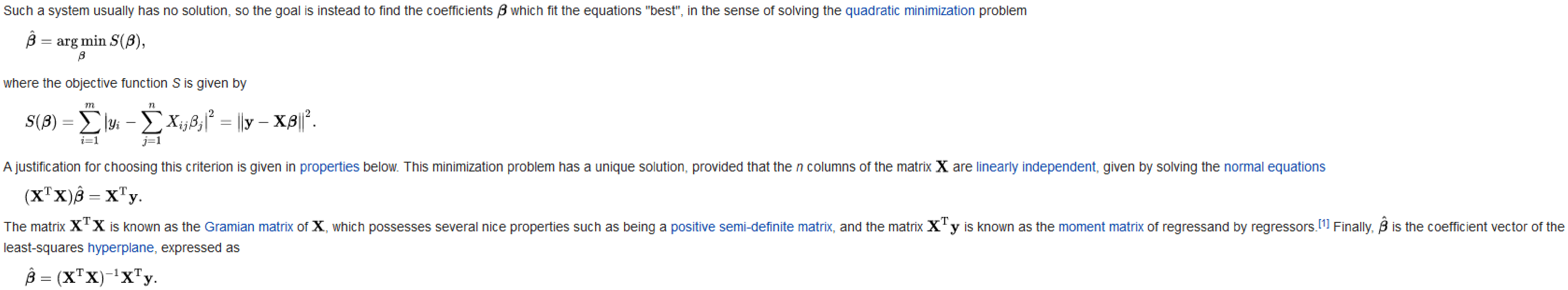
where y and ε are n×1 vectors of the values of the response variable and the errors for the various observations, and X is an n×p matrix of regressors, also sometimes called the design matrix, whose row i is x_i^T and contains the ith observations on all the explanatory variables.

where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

$$X = \begin{pmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \mathbf{x}_2^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_n^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$



Such a system usually has no solution, so the goal is instead to find the coefficients $\boldsymbol{\beta}$ which fit the equations "best", in the sense of solving the quadratic minimization problem

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\arg\min}\, S(\boldsymbol{\beta}),$$

where the objective function *S* is given by

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{m} \left| y_i - \sum_{j=1}^{n} X_{ij}\beta_j \right|^2 = \left\| \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \right\|^2.$$

A justification for choosing this criterion is given in properties below. This minimization problem has a unique solution, provided that the *n* columns of the matrix $\mathbf{X}$ are linearly independent, given by solving the normal equations

$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}^{\mathrm{T}}\mathbf{y}.$$

The matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is known as the Gramian matrix of $\mathbf{X}$, which possesses several nice properties such as being a positive semi-definite matrix, and the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{y}$ is known as the moment matrix of regressand by regressors.[1] Finally, $\hat{\boldsymbol{\beta}}$ is the coefficient vector of the least-squares hyperplane, expressed as

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}.$$

Example

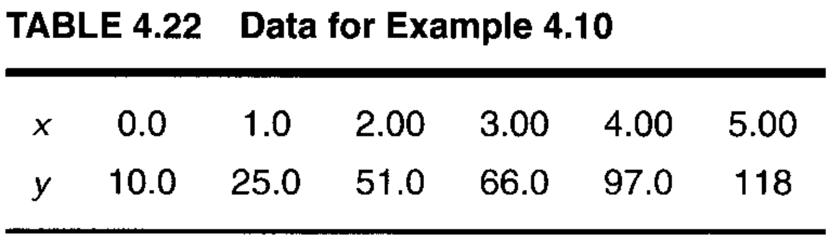
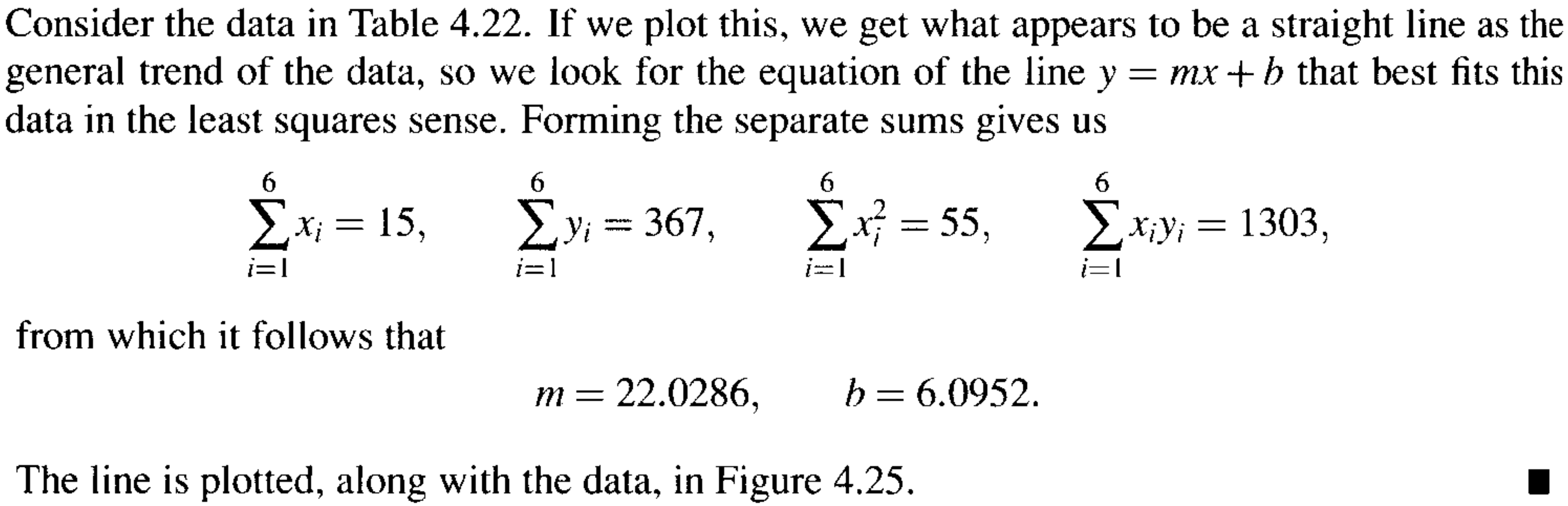


TABLE 4.22 Data for Example 4.10

x	0.0	1.0	2.00	3.00	4.00	5.00
y	10.0	25.0	51.0	66.0	97.0	118



Consider the data in Table 4.22. If we plot this, we get what appears to be a straight line as the general trend of the data, so we look for the equation of the line $y = mx + b$ that best fits this data in the least squares sense. Forming the separate sums gives us

$$\sum_{i=1}^{6} x_i = 15, \qquad \sum_{i=1}^{6} y_i = 367, \qquad \sum_{i=1}^{6} x_i^2 = 55, \qquad \sum_{i=1}^{6} x_i y_i = 1303,$$

from which it follows that

$$m = 22.0286, \qquad b = 6.0952.$$

The line is plotted, along with the data, in Figure 4.25. ■

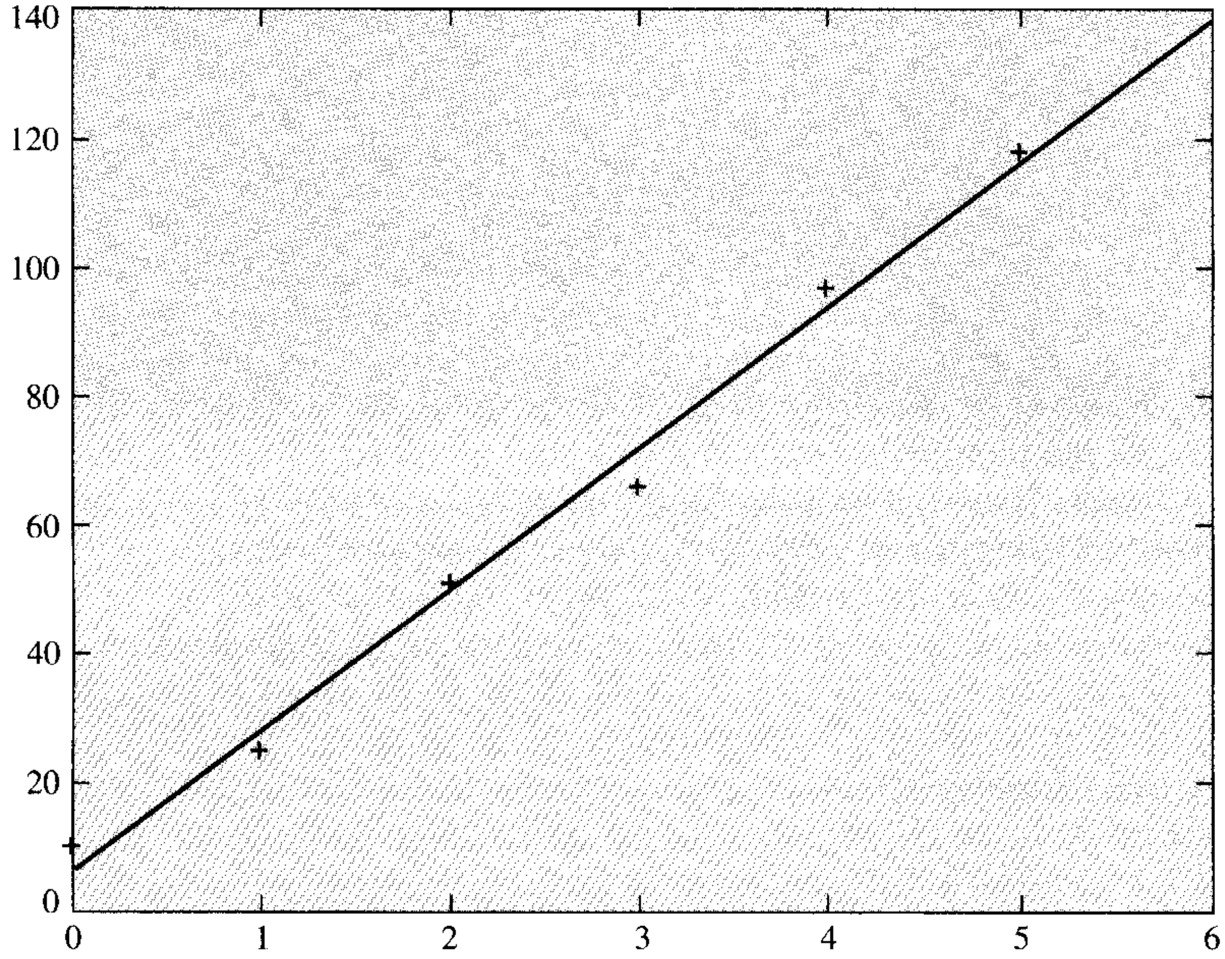


FIGURE 4.25 Least squares fit to the data in Table 4.22.

TP avec Scilab

Exercice 1 : énoncé

Le but de cet exercice est de compléter un programme permettant d'interpoler une fonction continue par un polynôme. Soit $f : [a, b] \to \mathbb{R}$ une fonction continue donnée et soit $t_0 < t_1 < t_2 < \cdots < t_n$, $(n+1)$ points de $[a, b]$. On cherche un polynôme p_n de degré inférieur ou égal à n tel que

$$p_n(t_j) = f(t_j), \qquad 0 \leqslant j \leqslant n.$$

Question 1 : Rappeler la formule du cours permettant d'expliciter p_n en fonction de la base de Lagrange de $\mathbb{P}_n$ associée aux points $t_0 < t_1 < t_2 < \cdots < t_n$.

Question 2 : Le programme `intlag.sci` est à votre disposition à la fin de ce document. A partir de la donnée de f et n, ce programme permet de calculer le polynôme p_n qui interpole f aux points t_j, $0 \leqslant j \leqslant n$, équidistribués uniformément sur l'intervalle $[a, b]$. Ce programme est incomplet, vous devez en particulier programmer les formules issues de la question 1 ci-dessus. Pour lancer le programme avec $n = 5$, tapez

```
[err,t,f,x,p]=intlag(5);
```

dans la fenêtre de commande `scilab`. Ceci aura pour effet de créer les vecteurs `t,f,x,p`. Les résultats pourront être visualisés en tapant ensuite

```
plot(t,f,'o',x,p);
```

Taper

```
err
```

dans la fenêtre de commande `scilab` pour obtenir une approximation de

$$\max_{-1 \leqslant t \leqslant 1} |f(t) - p_n(t)|.$$

Dans la suite on va considérer deux cas de figures : le cas où $f(t) = sin(t)$ et $[a, b] = [-\pi, \pi]$. Ensuite le cas où $f(t) = \dfrac{1}{1 + 25t^2}$ et $[a, b] = [-1, 1]$. Trois lignes sont à compléter dans le programme.

Question 3 : On considère le cas où $f(t) = sin(t)$. Remplir le tableau suivant

n	err
5	
10	
20	

Que vaut $\lim\limits_{n\to\infty} \max\limits_{-\pi\le t\le\pi} |f(t) - p_n(t)|$? Expliquer ce résultat grâce au théorème du cours.

Question 4 (Phénomène de Runge) : On considère le cas où $f(t) = \dfrac{1}{1+25t^2}$. Remplir le tableau suivant

n	err
5	
10	
20	
40	

Que vaut $\lim\limits_{n\to\infty} \max\limits_{-1\le t\le 1} |f(t) - p_n(t)|$?

Question 5 : On considère le cas où $f(t) = \dfrac{1}{1+25t^2}$ mais avec

$$t_j = \cos\frac{(2j-1)\pi}{2(n+1)}, \qquad j = 1, \cdots, n+1,$$

qui sont les points dits de Chebyshev. Que vaut $\lim\limits_{n\to\infty} \max\limits_{-1\le t\le 1} |f(t) - p_n(t)|$?.

Le fichier scilab intlag.sci

```
 // Etant donne un entier n et une fonction continue f,
// le programme interpole la fonction f par un polynome p
// de degre n aux points d interpolation t(1),t(2),...,t(n).
// parametres: entree : n
// sortie : err : erreur max entre la fonction f et l interpolant p
// t : (n+1) vecteur contenant les points d interpolation
// f : (n+1) vecteur contenant les valeurs de la fonction f
// aux points d interpolation
// x : vecteur contenant 1001 points uniformement distribues
// sur [-1,1]
// p : vecteur contenant les valeurs du polynome p au point x(i)
//
function [err,t,f,x,p]=intlag(n)
//
// initialisation des vecteurs t et f
//
for i=1:n+1
t(i)=-1+(2.*(i-1))/n;
f(i)=funct(t(i));
end
//
//calcul de la valeur du polynome d interpolation au point x(i)
//
m=1000;
err=0;
for i=1:m+1
x(i)=-1+(2.*(i-1))/m;
p(i)=0;
for j=1:n+1
p(i) = p(i) + f(j) * ?????;
end
err = max(err,abs(p(i)-funct(x(i))));
end
endfunction
// // calcul de $phi_j$ (la $j^{ieme}$ fonction de la base de Lagrange)
//
function basis = phi(j,n,xx,t)
basis=1;
for k=1:n+1
if k ~= j then
basis = basis * ?????;
end
end
endfunction
// // fonction a interpoler
// function f = funct(xx)
f = ????? ;
endfunction
```

Exercice 2

1. A la lumière de l’exercice précédent, écrire un programme avec Scilab « newton.sci » qui interpole la fonction sin(.) et la fonction de Runge.
2. Comparer les résultats obtenus avec le programme intlag.sci.
3. Conclure.

Exercice 3

1. Ecrire un programme avec Scilab « spline.sci » qui interpole la fonction sin(.) et la fonction de Runge.
2. Comparer les résultats obtenus avec les programmes «intlag.sci » et « newton.sci »
3. Conclure.

Interpolation par moindres carrés

Exercice 1 : Soient n points (x_1, y_1),...,(x_n, y_n) du plan. Faire un programme qui trouve la droite $y = ax + b$ minimisant $\sum_{i=1}^{n} |y_i - (ax_i + b)|^2$. Faire une sortie graphique.

2. Utiliser les données du tableau 4.22, pour valider votre programme et tracer la droite représentée dans la figure 4.25
3. Quel est l'ordre de l'erreur dans ce cas ?
4. Conclure.

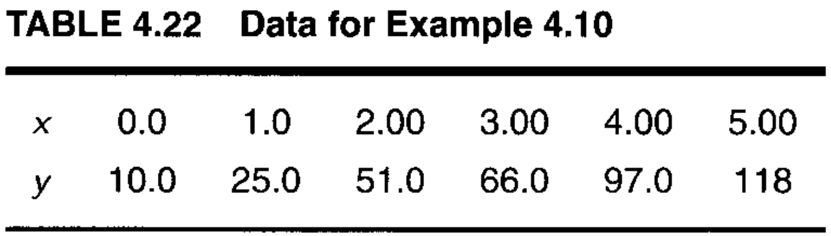


TABLE 4.22 Data for Example 4.10

x	0.0	1.0	2.00	3.00	4.00	5.00
y	10.0	25.0	51.0	66.0	97.0	118

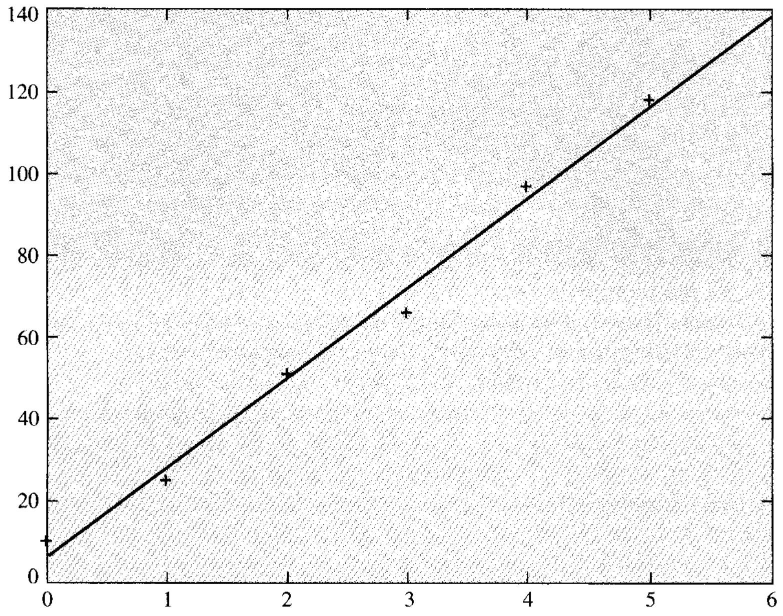


FIGURE 4.25 Least squares fit to the data in Table 4.22.

Références

1. http://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html
2. https://pdfs.semanticscholar.org/fc86/7badcf1b36685bb3f33950c6870075adbd70.pdf
3. https://en.wikipedia.org/wiki/Ordinary_least_squares
4. https://en.wikipedia.org/wiki/Linear_regression
5. http://users.rowan.edu/~hassen/NumerAnalysis/Interpolation_and_Approximation.pdf
6. http://www.rajgunesh.com/resources/downloads/numerical/cubicsplineinterpol.pdf
7. http://www.tf.uns.ac.rs/~omorr/radovan_omorjan_003_prII/s_examples/Scilab/Gilberto/scilab08.pdf
8. http://www.maths.lth.se/na/courses/FMN081/FMN081-06/lecture11.pdf