Interpolare spline

1. Exemplul lui Runge. Necesitatea interpolării pe porţiuni

(Steven C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, 3rd ed, ISBN-13:978-0-07-340110-2)

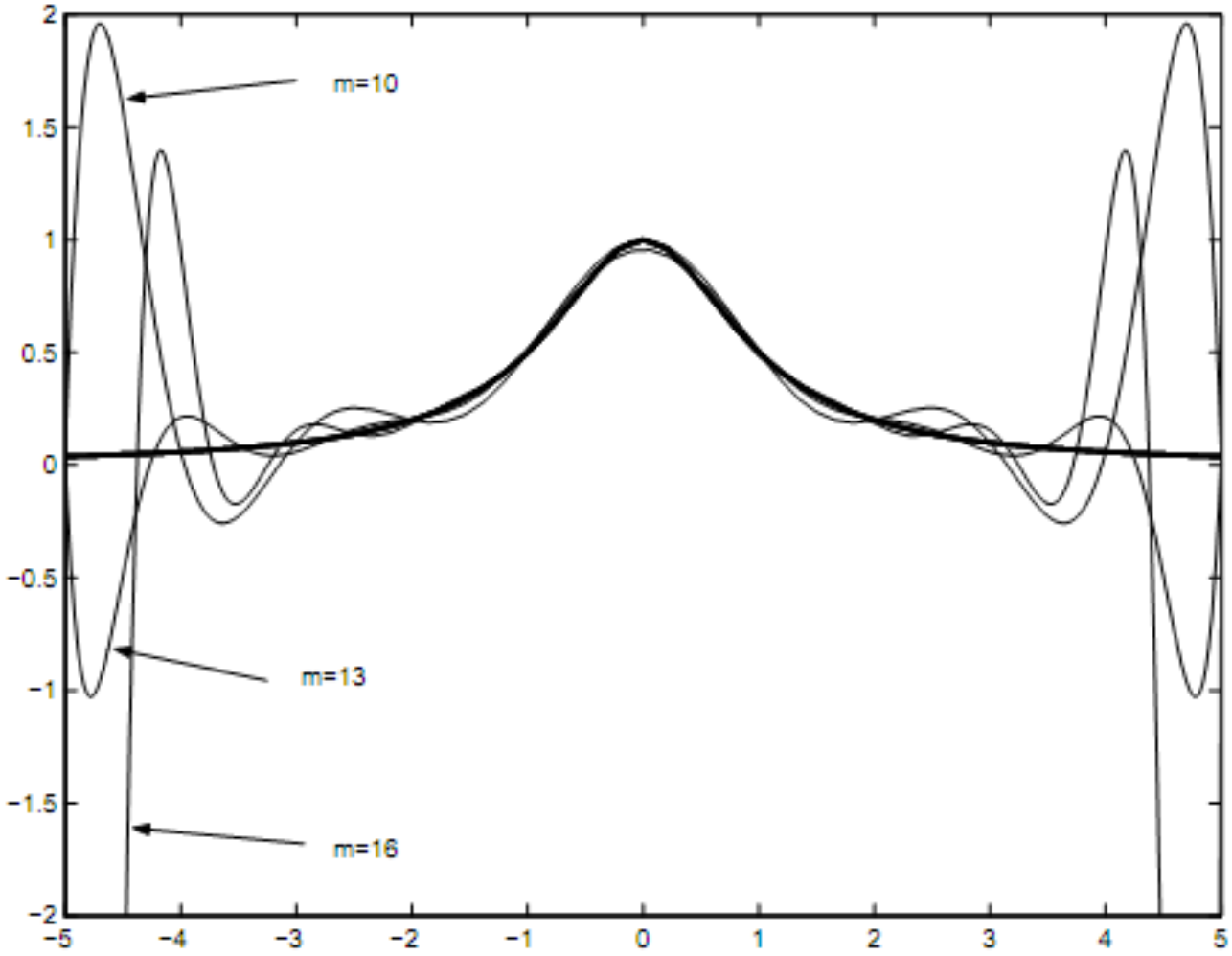
Although "more is better" in many contexts, it is absolutely not true for polynomial interpolation. Higher-order polynomials tend to be very ill-conditioned—that is, they tend to be highly sensitive to round-off error. The following example illustrates this point nicely.

Dangers of Higher-Order Polynomial Interpolation

Problem Statement. In 1901, Carl Runge published a study on the dangers of higher-order polynomial interpolation. He looked at the following simple-looking function:

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

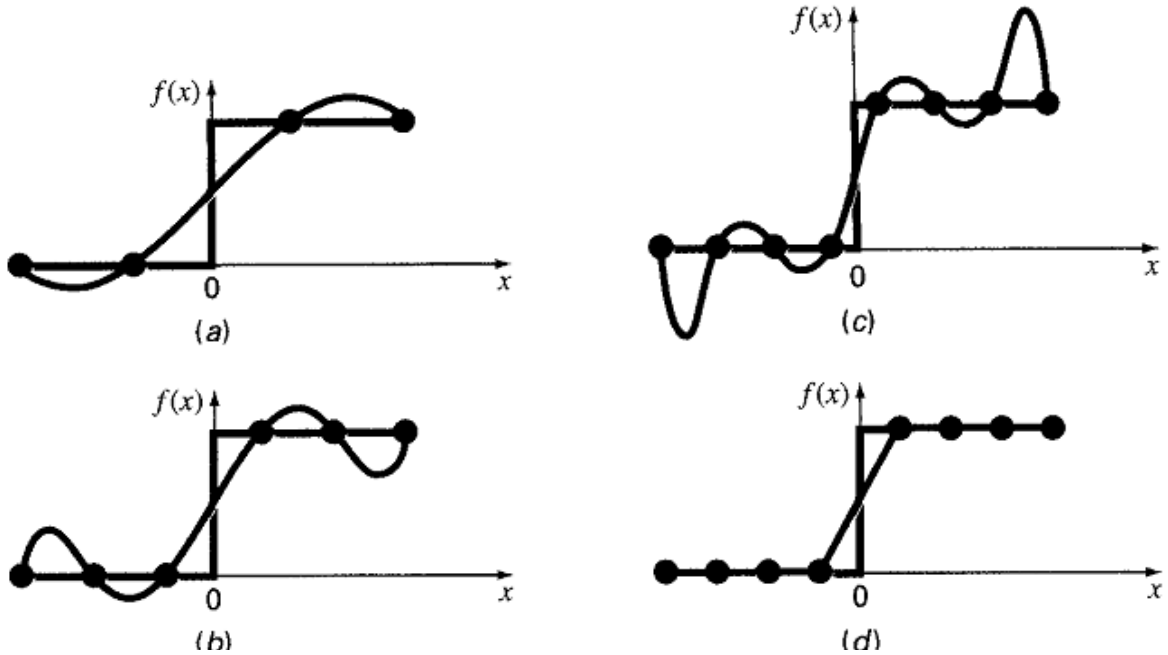
which is now called *Runge's function*. He took equidistantly spaced data points from this function over the interval [–1, 1]. He then used interpolating polynomials of increasing order and found that as he took more points, the polynomials and the original curve differed considerably. Further, the situation deteriorated greatly as the order was increased.



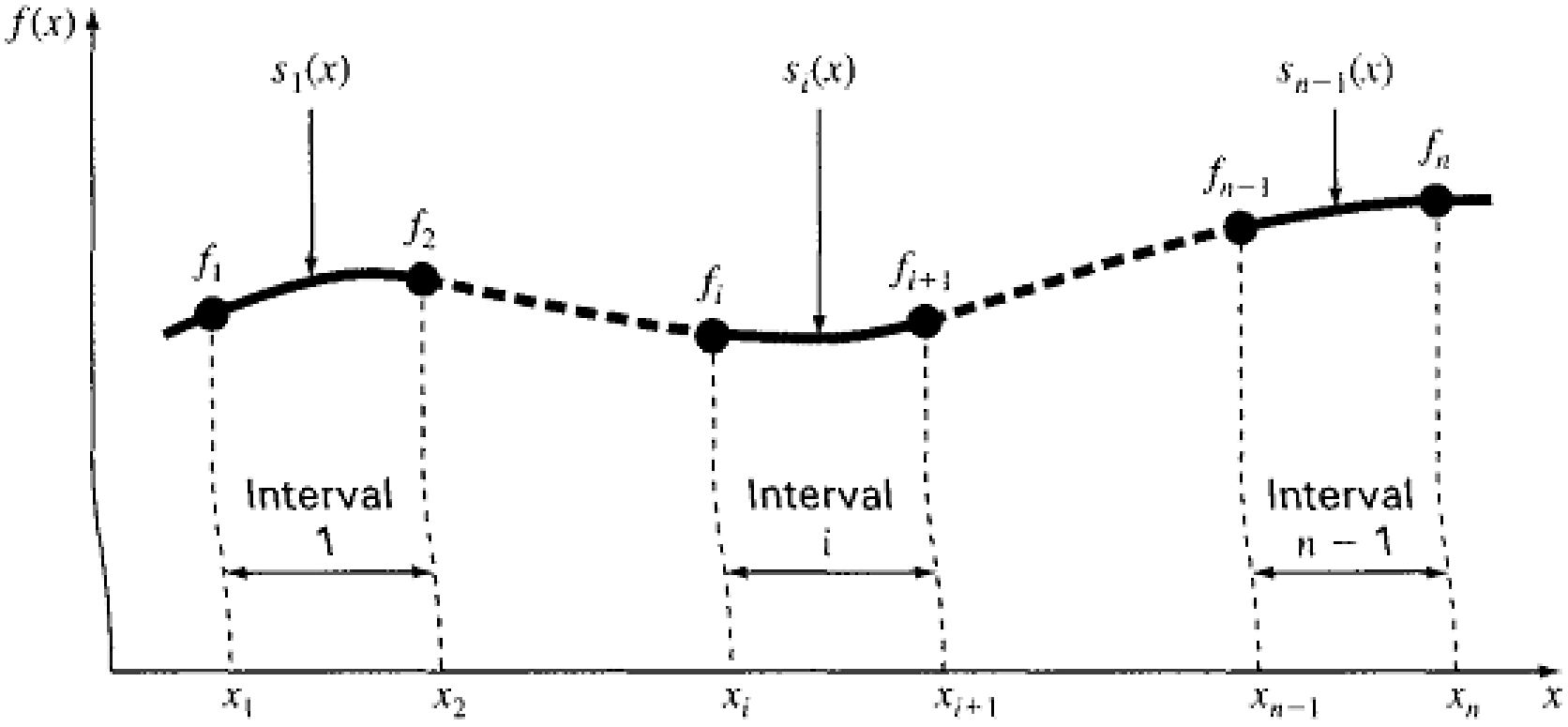
O ilustrare grafică a contraexemplului lui Runge

(R. Trîmbiţaş, 2005, Analiza numerica. Presa Universitară Clujeană)

Although there may be certain contexts where higher-order polynomials are necessary, they are usually to be avoided. In most engineering and scientific contexts, lower-order polynomials of the type described in this chapter can be used effectively to capture the curving trends of data without suffering from oscillations.



The function to be fit undergoes an abrupt increase at $x = 0$. Parts (a) through (c) indicate that the abrupt change induces oscillations in interpolating polynomials. In contrast, because it is limited to straight-line connections, a linear spline (d) provides a much more acceptable approximation.


f(x)
s1(x)
si(x)
sn−1(x)
f1
f2
fi
fi+1
fn−1
fn
Interval 1
Interval i
Interval n − 1
x1
x2
xi
xi+1
xn−1
xn
x

2.Spline cubice.

(Steven C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, 3rd ed, ISBN-13:978-0-07-340110-2)

The concept of the spline originated from the drafting technique of using a thin, flexible strip (called a *spline*) to draw smooth curves through a set of points. The process is depicted in Fig. 16.2 for a series of five pins (data points). In this technique, the drafter places paper over a wooden board and hammers nails or pins into the paper (and board) at the location of the data points. A smooth cubic curve results from interweaving the strip between the pins. Hence, the name "cubic spline" has been adopted for polynomials of this type.

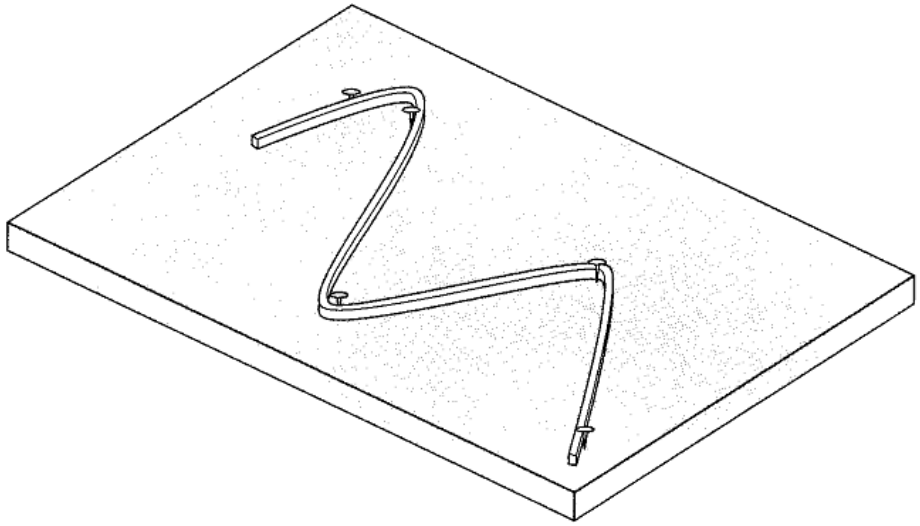


FIGURE 16.2
The drafting technique of using a spline to draw smooth curves through a series of points. Notice how, at the end points, the spline straightens out. This is called a "natural" spline.

(R. Trîmbiţaş, 2005, Analiza numerica. Presa Universitară Clujeană)

Fie Δ o diviziune a lui $[a, b]$

$$\Delta : \ a = x_1 < x_2 < \cdots < x_{n-1} < x_n = b.$$

Vom utiliza un polinom de grad mic pe subintervalul $[x_i, x_{i+1}]$, $i = \overline{1, n-1}$. Motivul este acela că pe intervale suficient de mici funcţiile pot fi aproximate arbitrar de bine prin polinoame de grad mic, chiar 0 sau 1.

Fie

$$\mathbb{S}_m^k(\Delta) = \{s : \ s \in C^k[a, b], \ s|_{[x_i, x_{i+1}]} \in \mathbb{P}_m, \ i = 1, 2, \ldots, n-1\}$$

$m \geq 0$, $k \in \mathbb{N} \cup \{-1\}$, numit spaţiul funcţiilor spline polinomiale de grad m şi clasă de netezime k. Dacă $k = m$, atunci funcţiile $s \in \mathbb{S}_m^m(\Delta)$ sunt polinoame.

Spline liniare

Pentru $m = 1$ şi $k = 0$ se obţin *spline liniare*. Dorim să găsim $s \in \mathbb{S}_1^0(\Delta)$ astfel încât

$$s(x_i) = f_i, \text{ unde } f_i = f(x_i), \quad i = 1, 2, \ldots, n.$$

Soluţia este trivială, vezi figura 5.15. Pe intervalul $[x_i, x_{i+1}]$

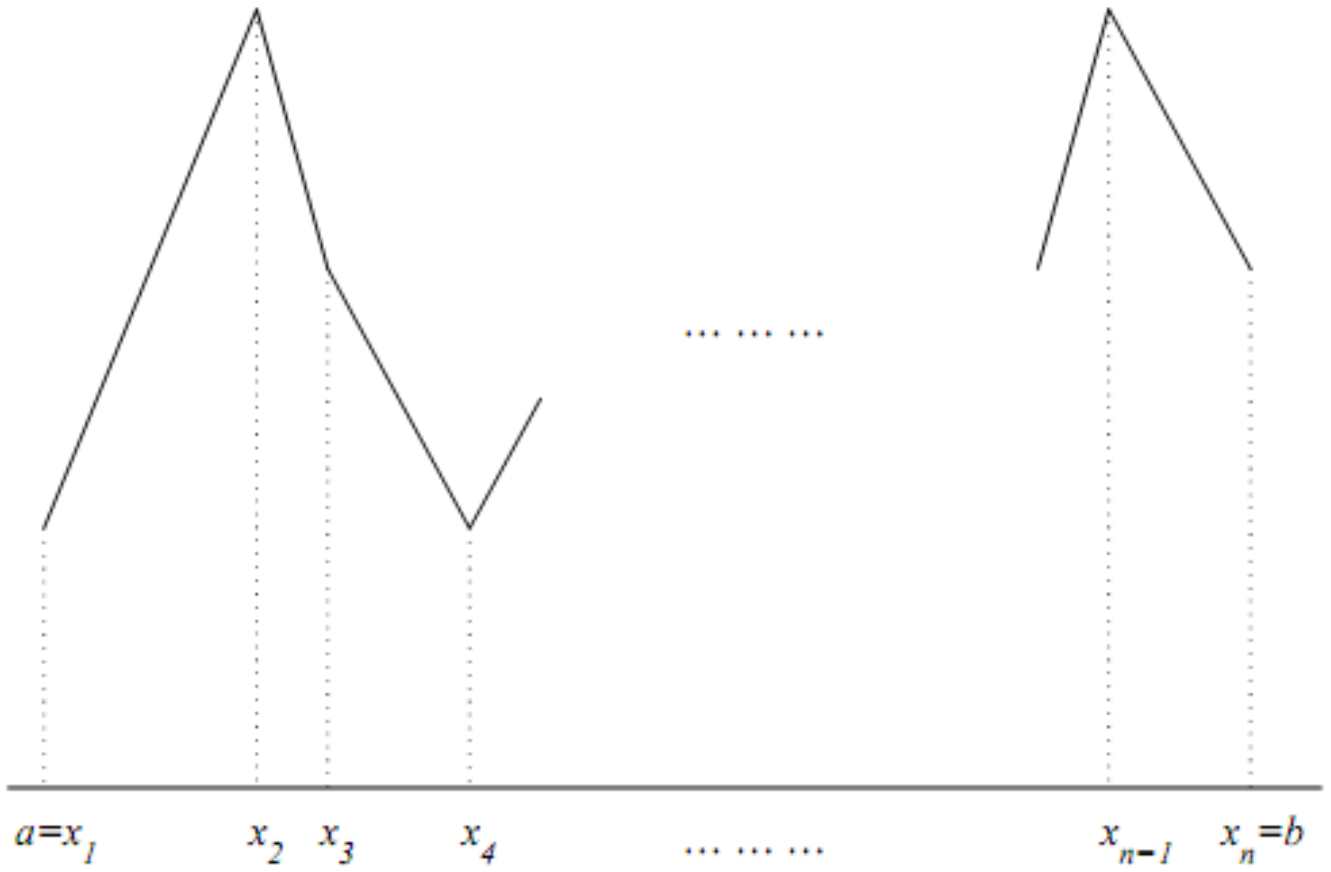
$$s(f; x) = f_i + (x - x_i) f[x_i, x_{i+1}],$$

iar

$$|f(x) - s(f(x))| \leq \frac{(\Delta x_i)^2}{8} \max_{x \in [x_i, x_{i+1}]} |f''(x)|.$$

Rezultă că

$$\|f(\cdot) - s(f, \cdot)\|_\infty \leq \frac{1}{8} |\Delta|^2 \|f''\|_\infty.$$



Spline liniare

(Steven C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, 3rd ed, ISBN-13:978-0-07-340110-2)

Exemplu

First-Order Splines

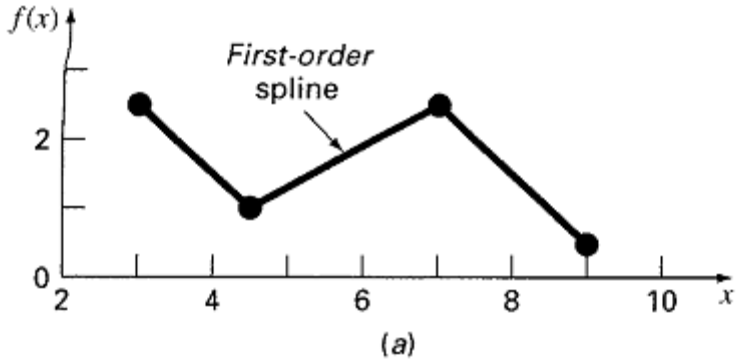
Problem Statement. Fit the data in Table 16.1 with first-order splines. Evaluate the function at $x = 5$.

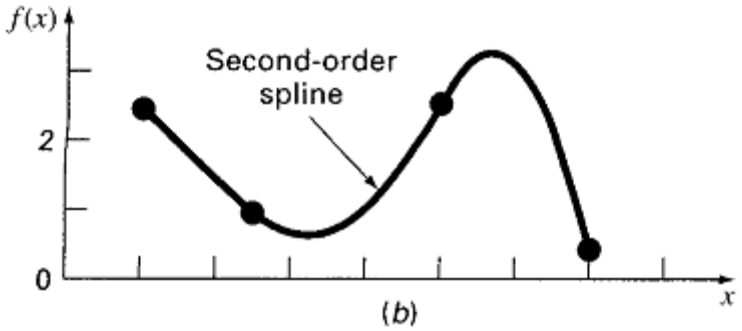
TABLE 16.1 Data to be fit with spline functions.

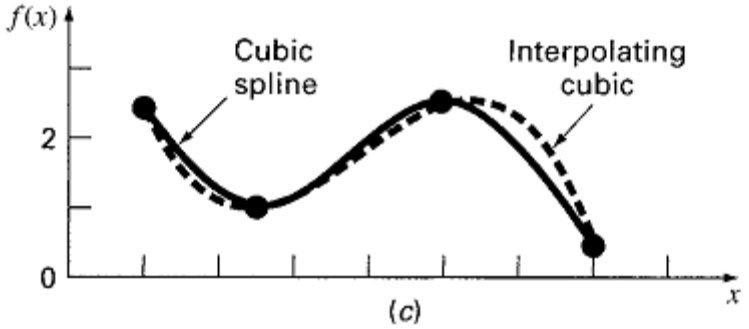
i	x_i	f_i
1	3.0	2.5
2	4.5	1.0
3	7.0	2.5
4	9.0	0.5

Solution. The data can be substituted into Eq. (16.4) to generate the linear spline functions. For example, for the second interval from $x = 4.5$ to $x = 7$, the function is

$$s_2(x) = 1.0 + \frac{2.5 - 1.0}{7.0 - 4.5}(x - 4.5)$$







Derivata de ordinul I este discontinuă, dar există aplicaţii

În general rol teoretic, demonstrează abordarea interpolării pe porţiuni cu polinoame de grad superior

Cea mai utilizată interpolare pe porţiuni, o vom discuta în continuare

Exemplu(Steven C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, 3rd ed)
Căutare în tabele şi interpolare liniară

T, °C	ρ, kg/m^3	μ, N · s/m^2	ν, m^2/s
−40	1.52	1.51×10^{-5}	0.99×10^{-5}
0	1.29	1.71×10^{-5}	1.33×10^{-5}
20	1.20	1.80×10^{-5}	1.50×10^{-5}
50	1.09	1.95×10^{-5}	1.79×10^{-5}
100	0.946	2.17×10^{-5}	2.30×10^{-5}
150	0.835	2.38×10^{-5}	2.85×10^{-5}
200	0.746	2.57×10^{-5}	3.45×10^{-5}
250	0.675	2.75×10^{-5}	4.08×10^{-5}
300	0.616	2.93×10^{-5}	4.75×10^{-5}
400	0.525	3.25×10^{-5}	6.20×10^{-5}
500	0.457	3.55×10^{-5}	7.77×10^{-5}

1. căutare secvenţială

```
function yi = TableLook(x, y, xx)

n = length(x);
if xx < x(1) | xx > x(n)
  error('Interpolation outside range')
end
% sequential search
i = 1;
while(1)
  if xx <= x(i + 1), break, end
  i = i + 1;
end
% linear interpolation
yi = y(i) + (y(i+1)-y(i))/(x(i+1)-x(i))*(xx-x(i));
```

2. căutare binară (volum mare de date)

```
function yi = TableLookBin(x, y, xx)

n = length(x);
if xx < x(1) | xx > x(n)
  error('Interpolation outside range')
end
% binary search
iL = 1; iU = n;
while (1)
  if iU - iL <= 1, break, end
  iM = fix((iL + iU) / 2);
  if x(iM) < xx
    iL = iM;
  else
    iU = iM;
  end
end
% linear interpolation
yi = y(iL) + (y(iL+1)-y(iL))/(x(iL+1)-x(iL))*(xx - x(iL));
```

unde

fix Round toward zero

```
a = [-1.9, -0.2, 3.4, 5.6, 7.0, 2.4+3.6i]
a =
  Columns 1 through 4
  -1.9000        -0.2000         3.4000         5.6000
  Columns 5 through 6
   7.0000         2.4000 + 3.6000i
fix(a)
ans =
  Columns 1 through 4
  -1.0000         0              3.0000         5.0000
  Columns 5 through 6
   7.0000         2.0000 + 3.0000i
```

Aplicaţie

```
>> T = [-40 0 20 50 100 150 200 250 300 400 500];
>> density = [1.52 1.29 1.2 1.09 .946 .935 .746 .675 .616
                .525 .457];
>> TableLookBin(T,density,350)

ans =
    0.5705
```

This result can be verified by the hand calculation:

$$f(350) = 0.616 + \frac{0.525 - 0.616}{400 - 300}(350 - 300) = 0.5705$$

(R. Trîmbiţaş, 2005, Analiza numerica. Presa Universitară Clujeană)

Interpolarea cu spline cubice

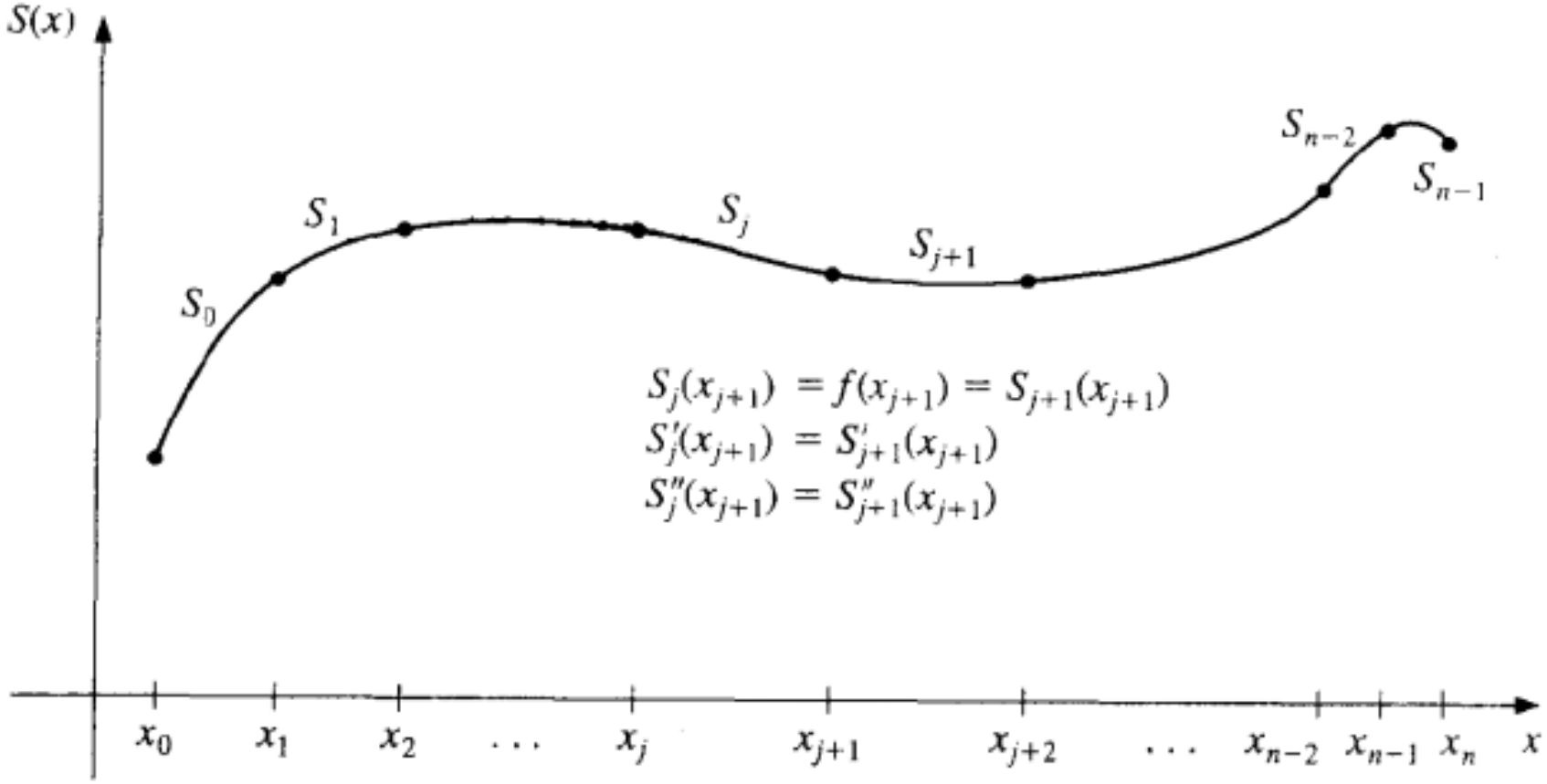
Funcţiile spline cubice sunt cele mai utilizate.

Vom discuta întâi problema interpolării pentru $s \in \mathbb{S}_3^1(\Delta)$. Continuitatea derivatei de ordinul I pentru $s_3(f;\cdot)$ se poate realiza impunând valorile primei derivate în fiecare punct x_i, $i = 1, 2, \ldots, n$. Astfel fie $m_1, m_2, \ldots, m_n$ numere arbitrare date şi notăm

$$s_3(f;\cdot)|_{[x_i, x_{i+1}]} = p_i(x), \quad i = 1, 2, \ldots, n-1$$

Realizăm $s_3(f; x_i) = m_i$, $i = \overline{1, n}$, luând fiecare bucată ca soluţie unică a problemei de interpolare Hermite, şi anume

$$\begin{aligned} &p_i(x_i) = f_i, \quad p_i(x_{i+1}) = f_{i+1}, \quad i = \overline{1, n-1}, \\ &p_i'(x_i) = m_i, \quad p_i'(x_{i+1}) = m_{i+1} \end{aligned}$$


S(x)
S_0
S_1
S_j
S_{j+1}
S_{n-2}
S_{n-1}
S_j(x_{j+1}) = f(x_{j+1}) = S_{j+1}(x_{j+1})
S'_j(x_{j+1}) = S'_{j+1}(x_{j+1})
S''_j(x_{j+1}) = S''_{j+1}(x_{j+1})
x_0
x_1
x_2
. . .
x_j
x_{j+1}
x_{j+2}
. . .
x_{n-2}
x_{n-1}
x_n
x

Vom rezolva problema folosind interpolarea Newton. Diferenţele divizate sunt

$$\begin{array}{cccccc}
x_i & f_i & & & & \\
 & & m_i & & & \\
x_i & f_i & & \frac{f[x_i,x_{i+1}]-m_i}{\Delta x_i} & & \frac{m_{i+1}+m_i-2f[x_i,x_{i+1}]}{(\Delta x_i)^2} \\
 & & f[x_i,x_{i+1}] & & & \\
x_{i+1} & f_{i+1} & & \frac{m_{i+1}-f[x_i,x_{i+1}]}{\Delta x_i} & & \\
 & & m_{i+1} & & & \\
x_{i+1} & f_{i+1} & & & &
\end{array}$$

şi deci forma Newton a polinomului de interpolare Hermite este

$$p_i(x) = f_i + (x-x_i)m_i + (x-x_i)^2\frac{f[x_i,x_{i+1}]-m_i}{\Delta x_i} + \\ + (x-x_i)^2(x-x_{i+1})\frac{m_{i+1}+m_i-2f[x_i,x_{i+1}]}{(\Delta x_i)^2}.$$

Forma Taylor a lui p_i pentru $x_i \leq x \leq x_{i+1}$ este

$$p_i(x) = c_{i,0} + c_{i,1}(x - x_i) + c_{i,2}(x - x_i)^2 + c_{i,3}(x - x_i)^3 \qquad (1)$$

şi deoarece $x - x_{i+1} = x - x_i - \Delta x_i$, prin identificare avem

$$\begin{aligned} c_{i,0} &= f_i \\ c_{i,1} &= m_i \\ c_{i,2} &= \frac{f[x_i, x_{i+1}] - m_i}{\Delta x_i} - c_{i,3}\Delta x_i \\ c_{i,3} &= \frac{m_{i+1} + m_i - 2f[x_i, x_{i+1}]}{(\Delta x_i)^2} \end{aligned} \qquad (2)$$

Deci, pentru a calcula $s_3(f; x)$ într-un punct care nu este nod. trebuie în prealabil să localizăm intervalul $[x_i, x_{i+1}] \ni x$, apoi să calculăm coeficienţii cu (2) şi să evaluăm spline-ul (1)

Vom discuta câteva alegeri posibile pentru $m_1, m_2, \ldots, m_n$.

Interpolare Hermite cubică pe porțiuni

Se alege $m_i = f'(x_i)$ (presupunând că aceste derivate sunt cunoscute). Se ajunge la o schemă strict locală, în care fiecare bucată poate fi determinată independent de cealaltă.

Interpolare cu spline cubice

Cerem ca $s_3(f;\cdot) \in \mathbb{S}_3^2(\Delta)$, adică continuitatea derivatelor de ordinul al II-lea.

$$p''_{i-1}(x_i) = p''_i(x_i), \quad i = \overline{2, n-1},$$

care convertită în coeficienți Taylor este

$$2c_{i-1,2} + 6c_{i-1,3}\Delta x_{i-1} = 2c_{i,2}, \quad i = \overline{2, n-1}.$$

Înlocuind cu valorile explicite (2) pentru coeficienți, se ajunge la sistemul liniar

$$\Delta x_i m_{i-1} + 2(\Delta x_{i-1} + \Delta x_i) m_i + \Delta x_{i-1} m_{i+1} = b_i, \quad i = \overline{2, n-1}, \tag{3}$$

unde

$$b_i = 3\{\Delta x_i f[x_{i-1}, x_i] + \Delta x_{i-1} f[x_i, x_{i+1}]\}. \tag{4}$$

Avem un sistem de $n-2$ ecuații liniare cu n necunoscute $m_1, m_2, \ldots, m_n$. Odată alese m_1 și m_n, sistemul devine tridiagonal și se poate rezolva eficient prin eliminare gaussiană, prin factorizare sau cu o metodă iterativă.

Se dau în continuare câteva alegeri posibile pentru m_1 și m_n.

Spline complete(racordate, limitate). Luăm $m_1 = f'(a)$, $m_n = f'(b)$. Se știe că pentru acest tip de spline, dacă $f \in C^4[a, b]$

$$\|f^{(r)}(\cdot) - s^{(r)}(f; \cdot)\|_\infty \leq c_r |\Delta|^{4-r} \|f^{(n)}\|_\infty, \quad r = 0, 1, 2, 3$$

unde $c_0 = \frac{5}{384}$, $c_1 = \frac{1}{24}$, $c_2 = \frac{3}{8}$, iar c_3 depinde de raportul $\frac{|\Delta|}{\min_i \Delta x_i}$.

Spline care utilizează derivatele secunde. Impunem condiţiile $s_3''(f;a) = f''(a)$; $s_3''(f;b) = f''(b)$. Aceste condiţii conduc la două ecuaţii suplimentare

$$\begin{aligned} 2m_1 + m_2 &= 3f[x_1, x_2] - \tfrac{1}{2}f''(a)\Delta x_1 \\ m_{n-1} + 2m_n &= 3f[x_{n-1}, x_n] + \tfrac{1}{2}f''(b)\Delta x_{n-1} \end{aligned} \qquad (5)$$

Prima ecuaţie se pune la începutul sistemului (5.5.15), iar a doua la sfârşitul lui, păstrându-se astfel structura tridiagonală a sistemului.

Spline cubice naturale. Impunând $s''(f;a) = s''(f;b) = 0$, se obţin două ecuaţii noi din (5) luând $f''(a) = f''(b) = 0$.

"Not-a-knot spline". (C. deBoor). Cerem ca $p_1(x) \equiv p_2(x)$ şi $p_{n-2}(x) \equiv p_{n-1}(x)$; adică primele două părţi şi respectiv ultimele două trebuie să coincidă. Aceasta înseamnă că primul punct interior x_2 şi ultimul x_{n-1} sunt ambele inactive. Se obţin încă două ecuaţii suplimentare exprimând continuitatea lui $s_3'''(f;x)$ în $x = x_2$ şi $x = x_{n-1}$. Condiţia de continuitate a lui $s_3(f,.)$ în x_2 şi x_{n-1} revine la egalitatea coeficienţilor dominanţi $c_{1,3} = c_{2,3}$ şi $c_{n-2,3} = c_{n-1,3}$. De aici se obţin ecuaţiile

$$
\begin{aligned}
(\Delta x_2)^2 m_1 + [(\Delta x_2)^2 - (\Delta x_1)^2] m_2 - (\Delta x_1)^2 m_3 &= \beta_1 \\
(\Delta x_2)^2 m_{n-2} + [(\Delta x_2)^2 - (\Delta x_1)^2] m_{n-1} - (\Delta x_1)^2 m_n &= \beta_2,
\end{aligned}
$$

unde

$$
\begin{aligned}
\beta_1 &= 2\{(\Delta x_2)^2 f[x_1, x_2] - (\Delta x_1)^2 f[x_2, x_3]\} \\
\beta_2 &= 2\{(\Delta x_{n-1})^2 f[x_{n-2}, x_{n-1}] - (\Delta x_{n-2})^2 f[x_{n-1}, x_n]\}.
\end{aligned}
$$

Prima ecuaţie se adaugă pe prima poziţie iar a doua pe ultima poziţie a sistemului format din cele $n-2$ ecuaţii date de (3) şi (4) . Sistemul obţinut nu mai este tridiagonal, dar el se poate transforma în unul tridiagonal combinând ecuaţiile 1 cu 2 şi $n-1$ cu n. După aceste transformări prima şi ultima ecuaţie devin

$$
\begin{aligned}
\Delta x_2 m_1 + (\Delta x_2 + \Delta x_1) m_2 &= \gamma_1 \\
(\Delta x_{n-1} + \Delta x_{n-2}) m_{n-1} + \Delta x_{n-2} m_n &= \gamma_2,
\end{aligned}
$$

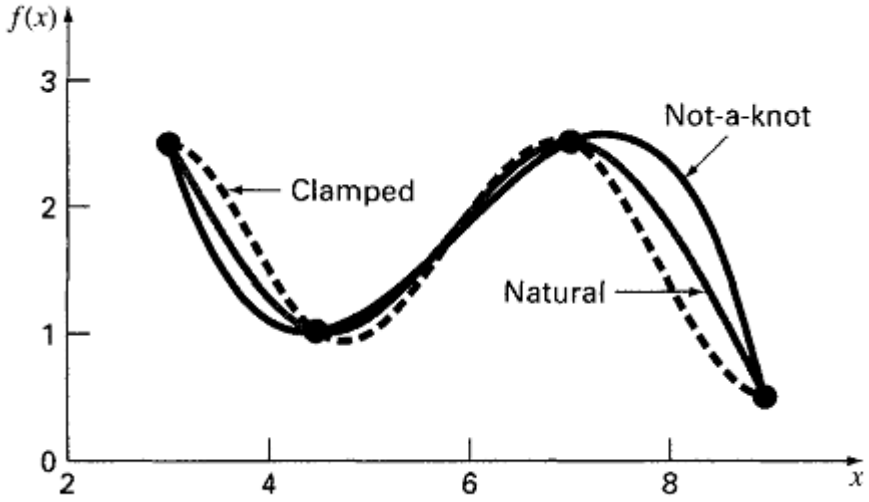
unde

$$\gamma_1 = \frac{1}{\Delta x_2 + \Delta x_1}\left\{f[x_1, x_2]\Delta x_2[\Delta x_1 + 2(\Delta x_1 + \Delta x_2)] + (\Delta x_1)^2 f[x_2, x_3]\right\}$$

$$\gamma_2 = \frac{1}{\Delta x_{n-1} + \Delta x_{n-2}}\left\{\Delta x_{n-1}^2 f[x_{n-2}, x_{n-1}] + [2(\Delta x_{n-1} + \Delta x_{n-2}) + \Delta x_{n-1}]\Delta x_{n-2} f[x_{n-1}, x_n]\right\}.$$

TABLE 16.1 Data to be fit with spline functions.

i	x_i	f_i
1	3.0	2.5
2	4.5	1.0
3	7.0	2.5
4	9.0	0.5



Algoritmul care calculează coeficienţii splin-ului

(R. Trîmbiţaş, 2005, Analiza numerica. Presa Universitară Clujeană)

```
function [a,b,c,d]=Splinecubic(x,f,tip,der)
%SPLINECUBIC - determina coeficientii spline-ului cubic
%x - abscisele
%f - ordonatele
%tip - 0  complet
%      1  cu derivate secunde
%      2  natural
%      3  not a knot (deBoor)
%der - informatii despre derivate
%      [f'(a),f'(b)] pentru tipul 0
%      [f''(a), f''(b)] pentru tipul 1
```

```
if (nargin<4) | (tip==2), der=[0,0]; end

n=length(x);
%sortare noduri
if any(diff(x)<0), [x,ind]=sort(x); else, ind=1:n; end
y=f(ind); x=x(:); y=y(:);
%obtin ecuatiile 2 ... n-1
dx=diff(x);  ddiv=diff(y)./dx;
ds=dx(1:end-1); dd=dx(2:end);
dp=2*(ds+dd);
md=3*(dd.*ddiv(1:end-1)+ds.*ddiv(2:end));
%tratare diferentiata tip - ecuatiile 1,n
switch tip
case 0 %complet
    dp1=1; dpn=1; vd1=0; vdn=0;
    md1=der(1); mdn=der(2);
case 1,2 %d2 si natural
    dp1=2; dpn=2; vd1=1; vdn=1;
    md1=3*ddiv(1)-0.5*dx(1)*der(1);
    mdn=3*ddiv(end)+0.5*dx(end)*der(2);
case 3  %deBoor
    x31=x(3)-x(1);xn=x(n)-x(n-2);
    dp1=dx(2); dpn=dx(end-1);
    vd1=x31; vdn=xn;
    md1=((dx(1)+2*x31)*dx(2)*ddiv(1)+dx(1)^2*ddiv(2))/x31;
    mdn=(dx(end)^2*ddiv(end-1)+(2*xn+dx(end))*dx(end-1)*...
       ddiv(end))/xn;
end
```

```
%construiesc sistemul rar
dp=[dpl;dp;dpn]; dpl=[0;vdl;dd];
dml=[ds;vdn;0]; md=[mdl;md;mdn];
A=spdiags([dml,dp,dpl],-1:1,n,n);
m=A \ md;
d=y(1:end-1);
c=m(1:end-1);
a=[(m(2:end)+m(1:end-1)-2*ddiv)./(dx.^2)];
b=[(ddiv-m(1:end-1))./dx-dx.*a];
```

Calculul valorilor funcţiilor spline se poate face cu o singură funcţie de evaluare pentru toate tipurile:

```
function z=valspline(x,a,b,c,d,t)
%evaluare spline
%apel z=valspline(x,a,b,c,d,t)
%z - valorile
%t - punctele in care se face evaluare
%x - nodurile
%a,b,c,d - coeficientii
n=length(x);
```

```
x=x(:); t=t(:);
k = ones(size(t));
for j = 2:n-1
    k(x(j) <= t) = j;
end
% Evaluare interpolant
s = t - x(k);
z = d(k) + s.*(c(k) + s.*(b(k) + s.*a(k)));
```

Exemplu(Burden, Richard L.; Faires, J. Douglas: Numerical Analysis, 8th ed., ISBN 0534392008.)

Figure 3.9 shows a ruddy duck in flight. To approximate the top profile of the duck, we have chosen points along the curve through which we want the approximating curve to pass. Table 3.15 lists the coordinates of 21 data points relative to the superimposed coordinate system shown in Figure 3.10 on page 150. Notice that more points are used when the curve is changing rapidly than when it is changing more slowly.

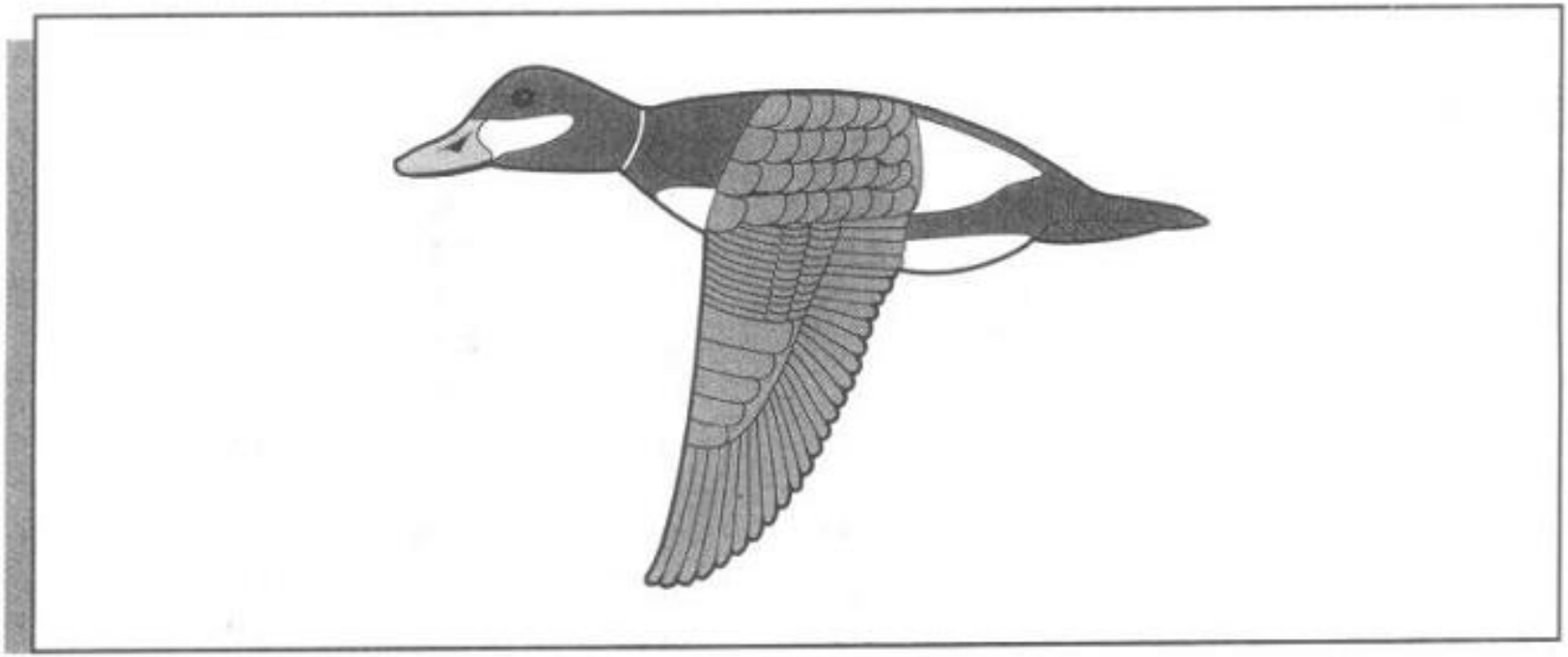
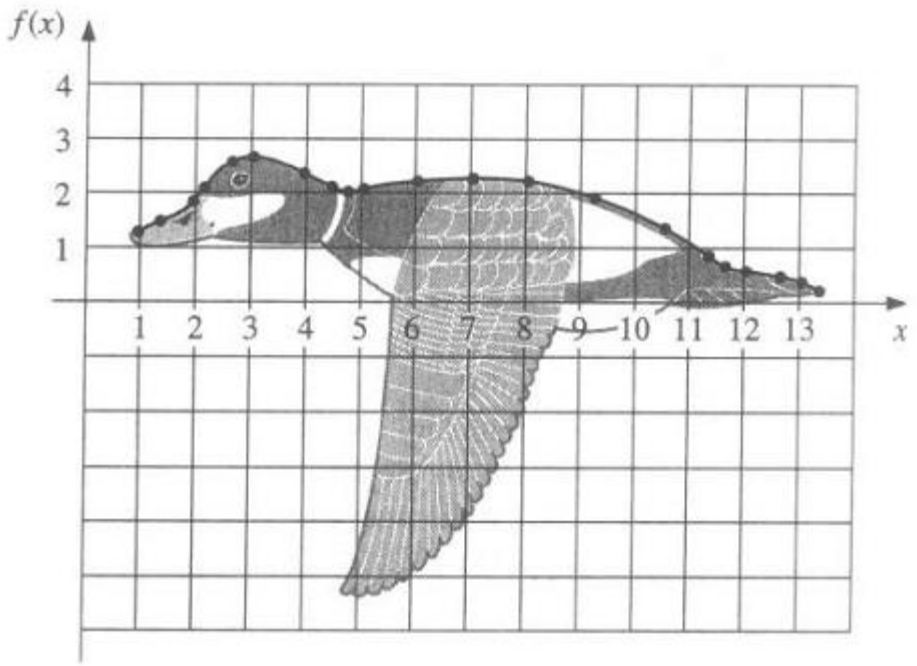


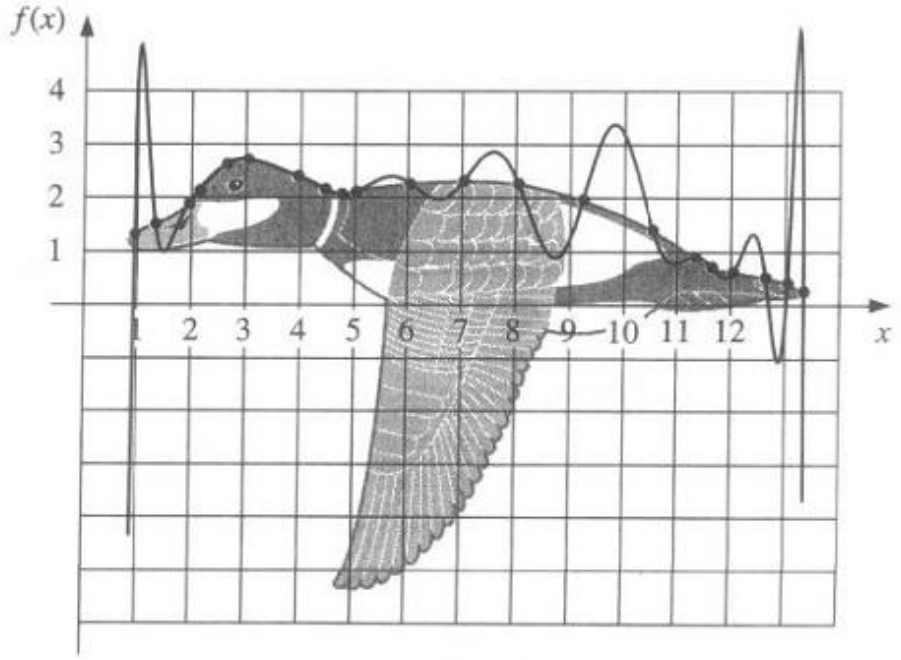
Table 3.15

x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3
$f(x)$	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25

j	x_j	a_j	b_j	c_j	d_j
0	0.9	1.3	5.40	0.00	−0.25
1	1.3	1.5	0.42	−0.30	0.95
2	1.9	1.85	1.09	1.41	−2.96
3	2.1	2.1	1.29	−0.37	−0.45
4	2.6	2.6	0.59	−1.04	0.45
5	3.0	2.7	−0.02	−0.50	0.17
6	3.9	2.4	−0.50	−0.03	0.08
7	4.4	2.15	−0.48	0.08	1.31
8	4.7	2.05	−0.07	1.27	−1.58
9	5.0	2.1	0.26	−0.16	0.04
10	6.0	2.25	0.08	−0.03	0.00
11	7.0	2.3	0.01	−0.04	−0.02
12	8.0	2.25	−0.14	−0.11	0.02
13	9.2	1.95	−0.34	−0.05	−0.01
14	10.5	1.4	−0.53	−0.10	−0.02
15	11.3	0.9	−0.73	−0.15	1.21
16	11.6	0.7	−0.49	0.94	−0.84
17	12.0	0.6	−0.14	−0.06	0.04
18	12.6	0.5	−0.18	0.00	−0.45
19	13.0	0.4	−0.39	−0.54	0.60
20	13.3	0.25			



Spline



Lagrange

Interpolare în Matlab

(Steven C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, 3rd ed)

MATLAB Function: `spline`

Cubic splines can be easily computed with the built-in MATLAB function, `spline`. It has the general syntax,

```
yy = spline(x, y, xx)
```

where x and y = vectors containing the values that are to be interpolated, and yy = a vector containing the results of the spline interpolation as evaluated at the points in the vector xx.

By default, `spline` uses the not-a-knot condition. However, if y contains two more values than x has entries, then the first and last value in y are used as the derivatives at the end points. Consequently, this option provides the means to implement the clamped-end condition.

Exemplu(Steven C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, 3rd ed)

Problem Statement. Runge's function is a notorious example of a function that cannot be fit well with polynomials (recall Example 15.7):

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

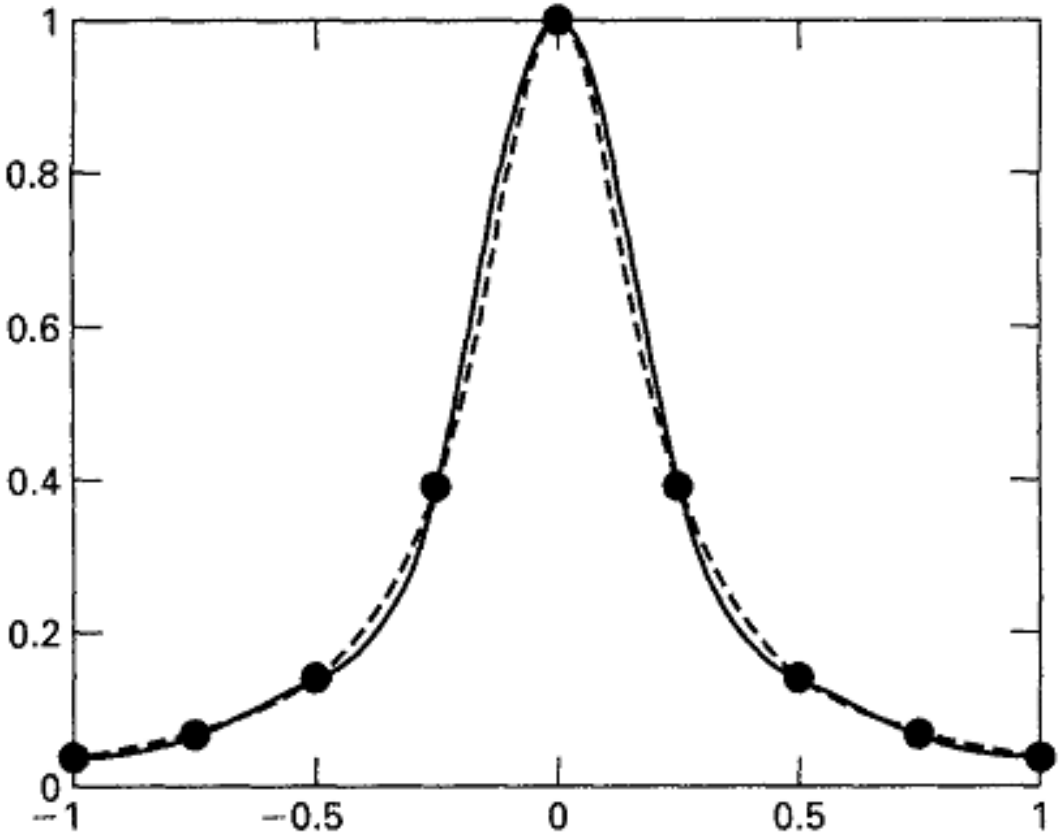
Use MATLAB to fit nine equally spaced data points sampled from this function in the interval $[-1, 1]$. Employ **(a)** a not-a-knot spline and **(b)** a clamped spline with end slopes of $f_1' = 1$ and $f_{n-1}' = -4$.

a)

```
>> x = linspace(-1,1,9);
>> y = 1./(1+25*x.^2);

>> xx = linspace(-1,1);
>> yy = spline(x,y,xx);

>> yr = 1./(1+25*xx.^2);
>> plot(x,y,'o',xx,yy,xx,yr,'--')
```

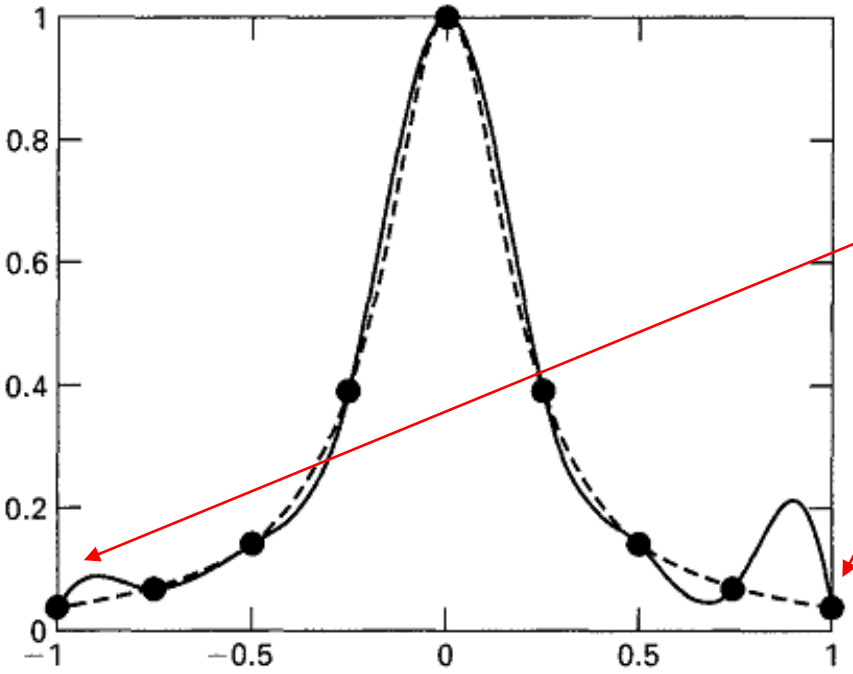


Linie frântă – funcţia lui Runge

Linie continuă – aproximare spline

(b) The clamped condition can be implemented by creating a new vector `yc` that has the desired first derivatives as its first and last elements.

```
>> yc = [1 y -4];
>> yyc = spline(x,yc,xx);
>> plot(x,y,'o',xx,yyc,xx,yr,'--')
```



Comparison of Runge's function (dashed line) with a 9-point clamped end spline fit generated with MATLAB (solid line). Note that first derivatives of 1 and −4 are specified at the left and right boundaries, respectively.

Din cauza condiţiilor impuse avem oscilaţii la capete. Dacă se dau derivatele corecte atunci rezultatul se îmbunătăţeşte

(R. Trîmbiţaş, 2005, Analiza numerica. Presa Universitară Clujeană)

MATLAB are funcţii pentru interpolare în una, două sau mai multe dimensiuni. Funcţia `polyfit` returnează coeficienţii polinomului de interpolare Lagrange dacă gradul n este egal cu numărul de observaţii minus 1.

Funcţia `interp1` acceptă perechi de date `x(i), y(i)` şi un vector `xi` al punctelor în care se face evaluarea. Ea construieşte interpolantul corespunzător datelor `x` şi `y` şi returnează valorile interpolantului în punctele din `xi`:

```
yi = interp1(x,y,xi,metoda)
```

Elementele vectorului `x` trebuie să fie ordonate crescător. Se admit patru tipuri de interpolanţi, precizate de parametrul `metoda`, care poate avea una din următoarele valori

- `'nearest'` - interpolare bazată pe vecinul cel mai apropiat;
- `'linear'` - interpolare liniară pe porţiuni (metoda implicită);
- `'spline'` - interpolare cu spline cubice;
- `'cubic'` sau `'pchip'` - interpolare Hermite cubică pe porţiuni.

Exemplu(Steven C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, 3rd ed)

Problem Statement. You perform a test drive on an automobile where you alternately accelerate the automobile and then hold it at a steady velocity. Note that you never decelerate during the experiment. The time series of spot measurements of velocity can be tabulated as

t	0	20	40	56	68	80	84	96	104	110
v	0	20	20	38	80	80	100	100	125	125

Use MATLAB's `interp1` function to fit this data with **(a)** linear interpolation, **(b)** nearest neighbor, **(c)** cubic spline with not-a-knot end conditions, and **(d)** piecewise cubic Hermite interpolation.

Solution. **(a)** The data can be entered, fit with linear interpolation, and plotted with the following commands:

```
>> t = [0 20 40 56 68 80 84 96 104 110];
>> v = [0 20 20 38 80 80 100 100 125 125];
>> tt = linspace(0,110);
>> vl = interp1(t,v,tt);
>> plot(t,v,'o',tt,vl)
```

The results (Fig. 16.8*a*) are not smooth, but do not exhibit any overshoot.

(b) The commands to implement and plot the nearest neighbor interpolation are

```
>> vn = interp1(t,v,tt,'nearest');
>> plot(t,v,'o',tt,vn)
```

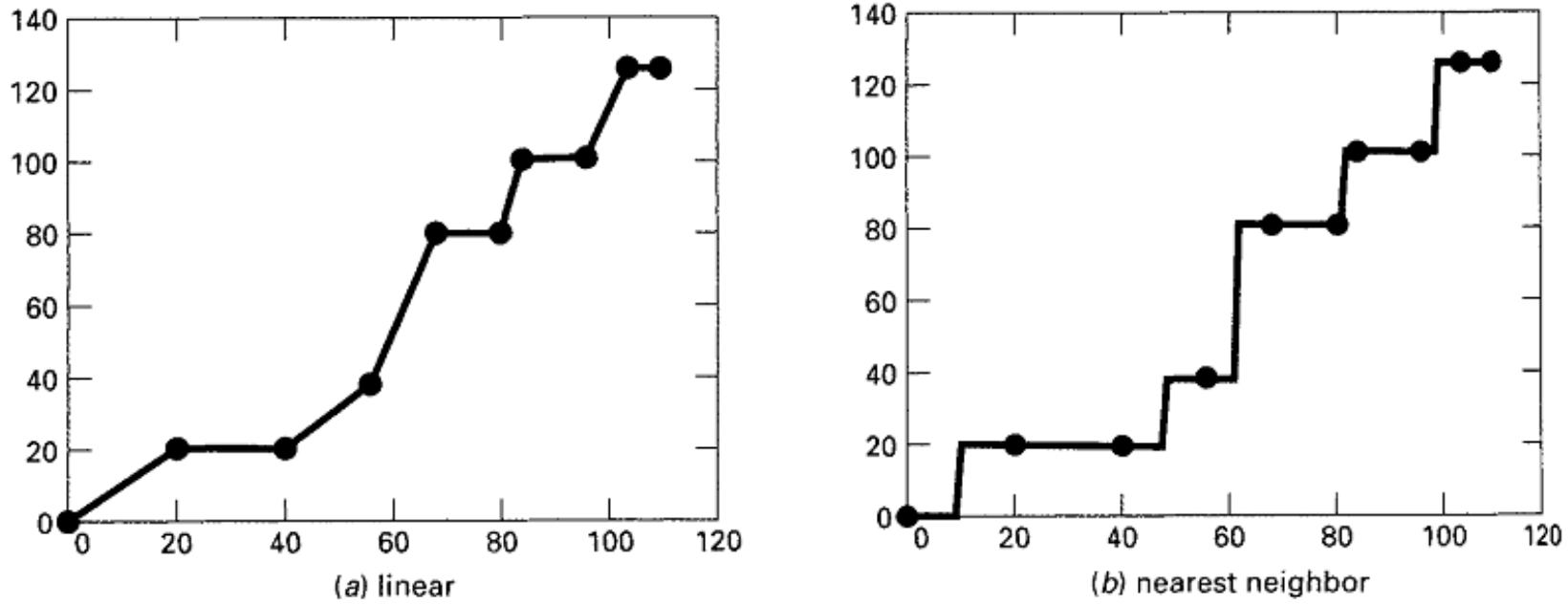
(c) The commands to implement the cubic spline are

```
>> vs = interp1(t,v,tt,'spline');
>> plot(t,v,'o',tt,vs)
```

(d) The commands to implement the piecewise cubic Hermite interpolation are

```
>> vh = interp1(t,v,tt,'pchip');
>> plot(t,v,'o',tt,vh)
```

Viteza:



The results (Fig. 16.8*a*) are not smooth, but do not exhibit any overshoot.

As in Fig. 16.8*b*, the results look like a series of plateaus. This option is neither a smooth nor an accurate depiction of the underlying process.

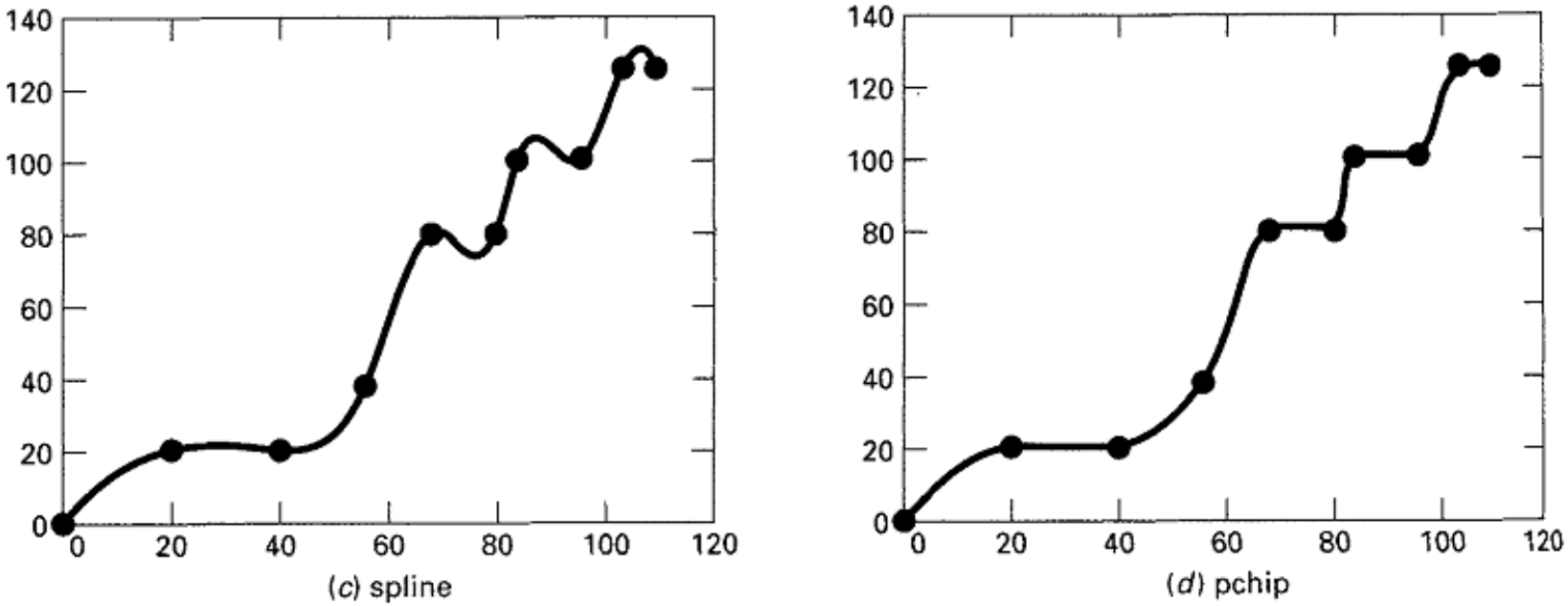


FIGURE 16.8
Use of several options of the `interp1` function to perform piecewise polynomial interpolation on a velocity time series for an automobile.

These results (Fig. 16.8*c*) are quite smooth. However, severe overshoot occurs at several locations. This makes it appear that the automobile decelerated several times during the experiment.

For this case, the results (Fig. 16.8*d*) are physically realistic. Because of its shape-preserving nature, the velocities increase monotonically and never exhibit deceleration. Although the result is not as smooth as for the cubic splines, continuity of the first derivatives at the knots makes the transitions between points more gradual and hence more realistic.