MATLAB 与工程应用

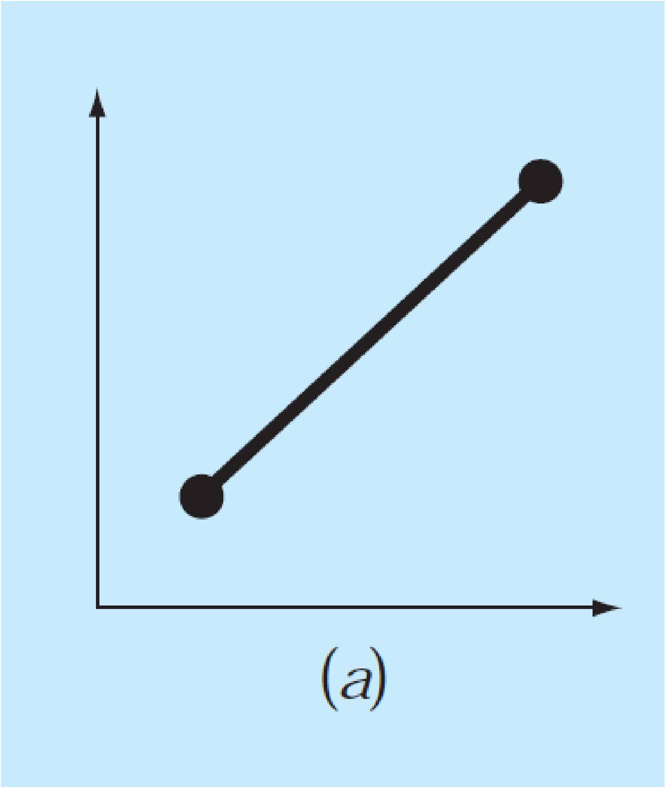
Interpolation

Case Study

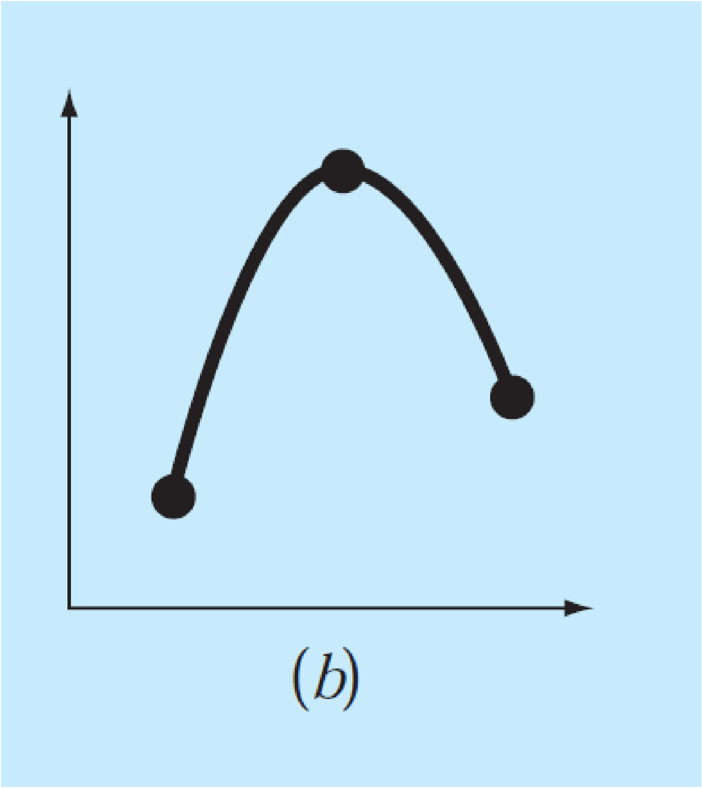
- You've taken data for measured temperature as a function of time from a hot water faucet
- Estimate temperature at t=0.6, 2.5, 4.7, and 8.9 seconds
- Estimate time it will take to reach T=75, 85, 90, and 105 degrees

Time (s)	Temperature (F)
0	72.5
1	78.1
2	86.4
3	92.3
4	110.6
5	111.5
6	109.3
7	110.2
8	110.5
9	109.9
10	110.2

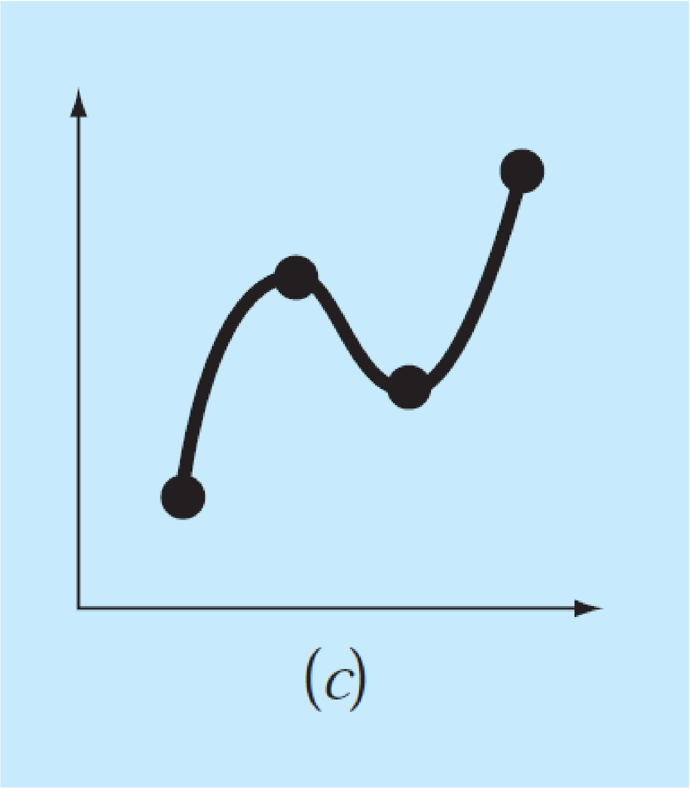
Interpolating polynomials



first-order (linear)



second-order (quadratic or parabolic)

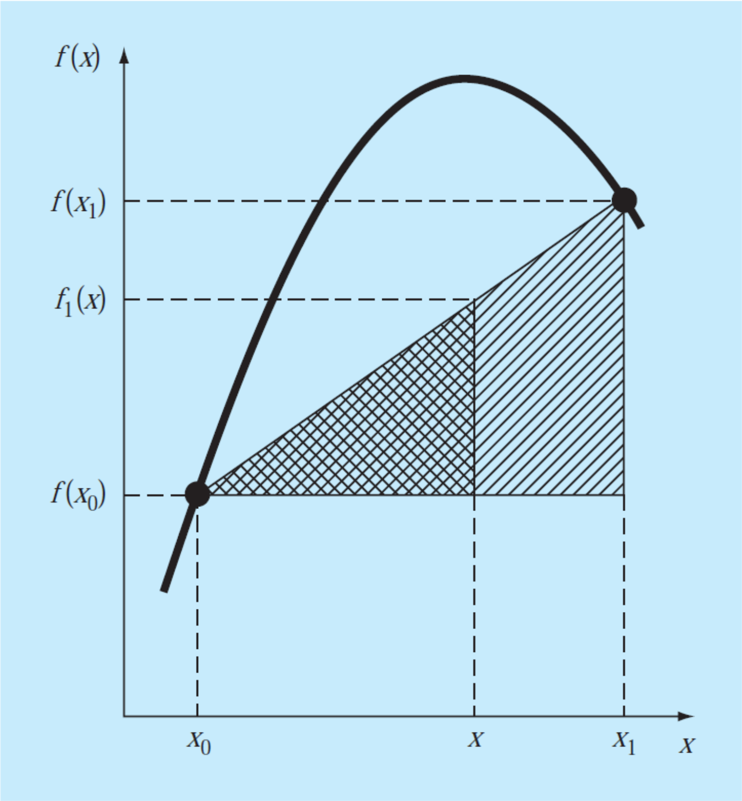


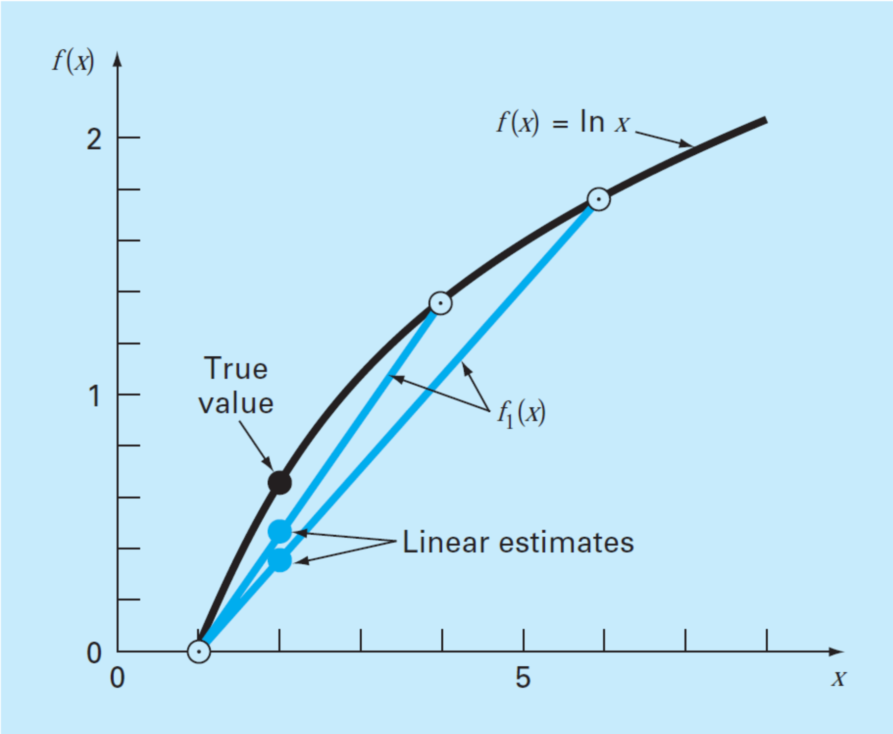
third-order (cubic)

Linear Interpolation

$$\frac{f_1(x) - f(x_0)}{x - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$f_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0)$$





Quadratic Interpolation

$$f_2(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1)$$

$$f_2(x) = b_0 + b_1 x - b_1 x_0 + b_2 x^2 + b_2 x_0 x_1 - b_2 x x_0 - b_2 x x_1$$

$$f_2(x) = a_0 + a_1 x + a_2 x^2$$

$$a_0 = b_0 - b_1 x_0 + b_2 x_0 x_1$$

$$a_1 = b_1 - b_2 x_0 - b_2 x_1$$

$$a_2 = b_2$$

$$b_0 = f(x_0)$$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$b_2 = \frac{\dfrac{f(x_2) - f(x_1)}{x_2 - x_1} - \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}$$

General Form of Newton's Interpolating Polynomials

The nth-order polynomial

$$f_n(x) = b_0 + b_1(x - x_0) + \cdots + b_n(x - x_0)(x - x_1)\cdots(x - x_{n-1})$$

$$
\begin{aligned}
b_0 &= f(x_0) \\
b_1 &= f[x_1, x_0] \\
b_2 &= f[x_2, x_1, x_0] \\
&\cdot \\
&\cdot \\
&\cdot \\
b_n &= f[x_n, x_{n-1}, \ldots, x_1, x_0]
\end{aligned}
$$

$$f[x_i, x_j] = \frac{f(x_i) - f(x_j)}{x_i - x_j}$$

$$f[x_i, x_j, x_k] = \frac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k}$$

the *nth finite divided difference*

$$f[x_n, x_{n-1}, \ldots, x_1, x_0] = \frac{f[x_n, x_{n-1}, \ldots, x_1] - f[x_{n-1}, x_{n-2}, \ldots, x_0]}{x_n - x_0}$$

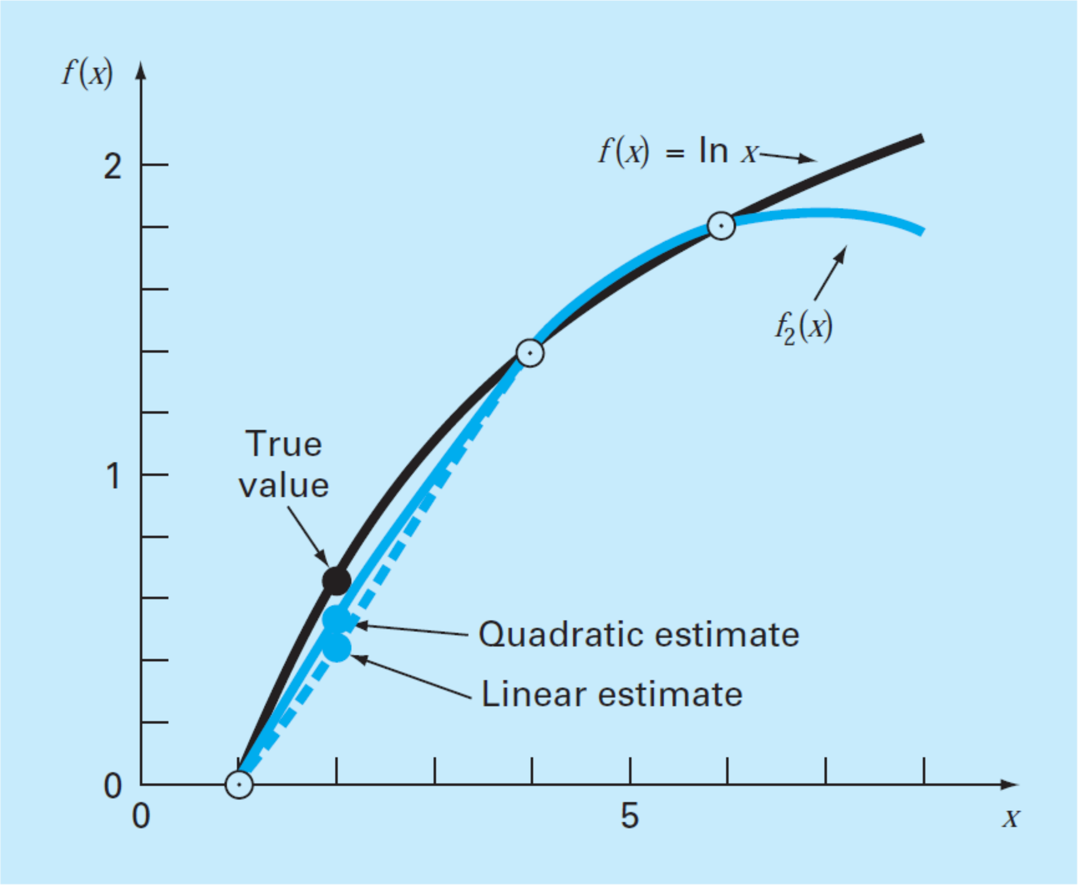
Newton's Divided-Difference Interpolating Polynomials

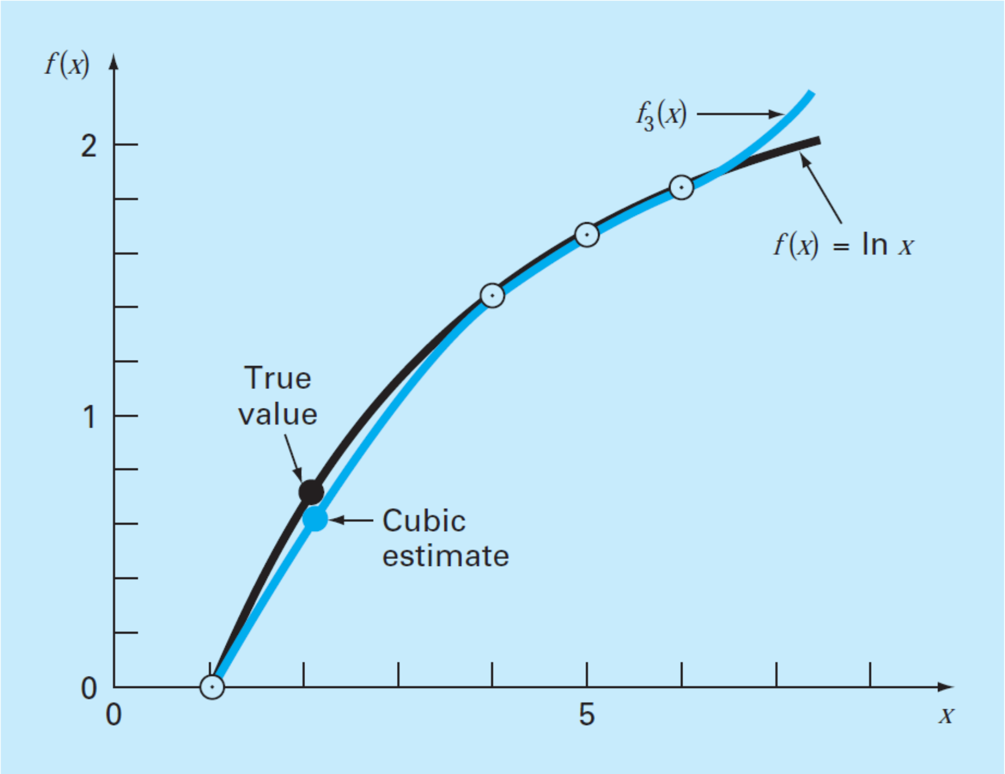
$$f_n(x) = f(x_0) + (x - x_0)\, f[x_1, x_0] + (x - x_0)(x - x_1)\, f[x_2, x_1, x_0]$$
$$+ \cdots + (x - x_0)(x - x_1) \cdots (x - x_{n-1})\, f[x_n, x_{n-1}, \ldots, x_0]$$

i	$\boldsymbol{x_i}$	$\boldsymbol{f(x_i)}$	**First**	**Second**	**Third**
0	x_0	$f(x_0)$	$f[x_1, x_0]$	$f[x_2, x_1, x_0]$	$f[x_3, x_2, x_1, x_0]$
1	x_1	$f(x_1)$	$f[x_2, x_1]$	$f[x_3, x_2, x_1]$	
2	x_2	$f(x_2)$	$f[x_3, x_2]$		
3	x_3	$f(x_3)$			

Graphical depiction of the recursive nature of finite divided differences.

The use of polynomials to estimate ln2





Interpolation

- Defining a function that takes on specified values at specified points
- Unlike curve fits, interpolation always goes through the data points
- Generally piece-wise, rather than covering entire range
- Often, first approach is to draw straight lines between points

Polynomials

- For N data points, there is a unique polynomial (usually of order n-1) that goes through each point
- This is an interpolating polynomial, because it goes exactly through each data point
- Problem: between data points, function can vary by large amount

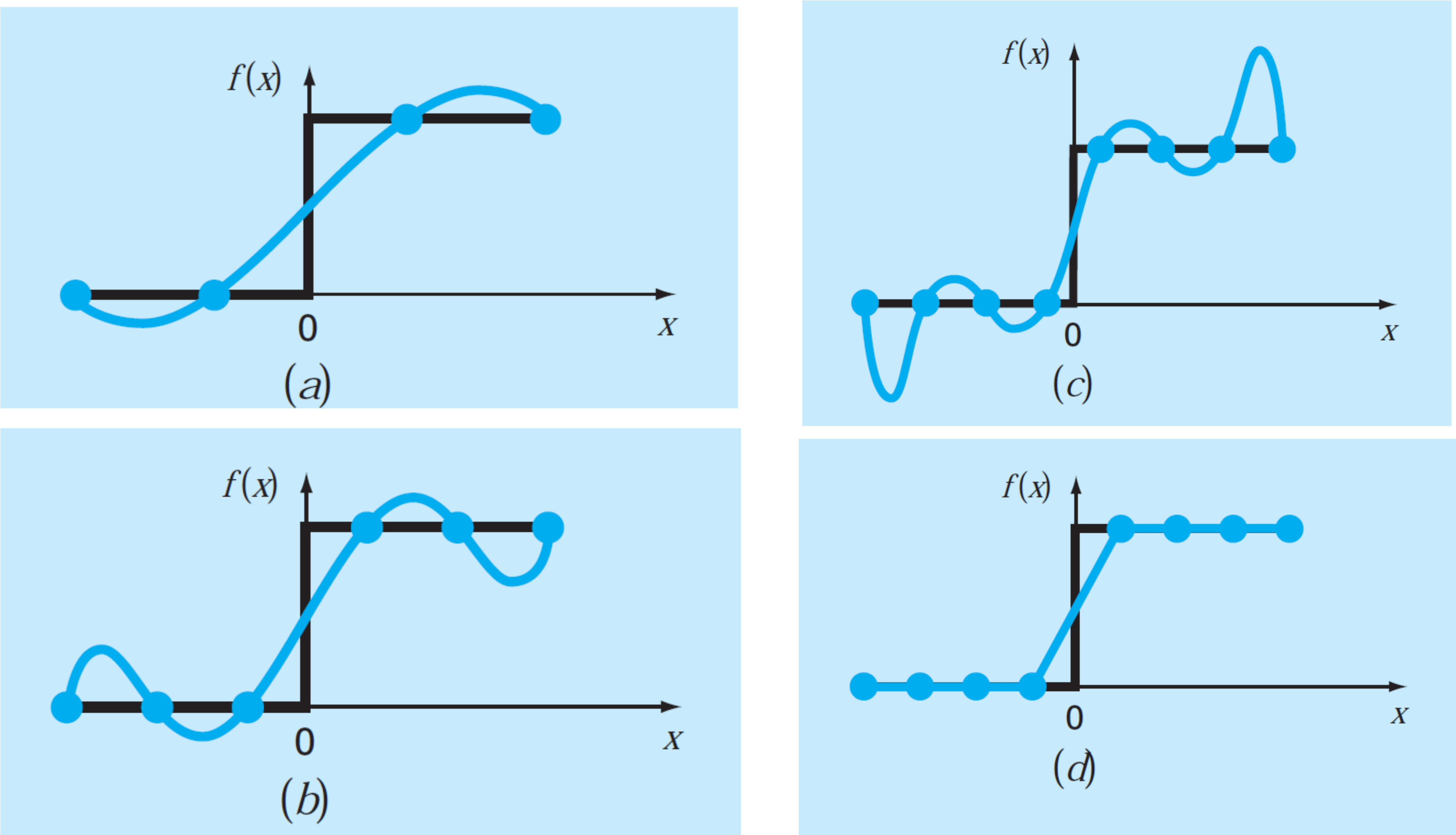
Polynomials

- For N data points, there is a unique polynomial (usually of order n-1) that goes through each point
- This is an interpolating polynomial, because it goes exactly through each data point
- Problem: between data points, function can vary by large amount

Piecewise linear interpolation

- Connect each data point by a straight line

SPLINE INTERPOLATION



Linear Splines

$$f(x) = f(x_0) + m_0(x - x_0) \qquad x_0 \le x \le x_1$$

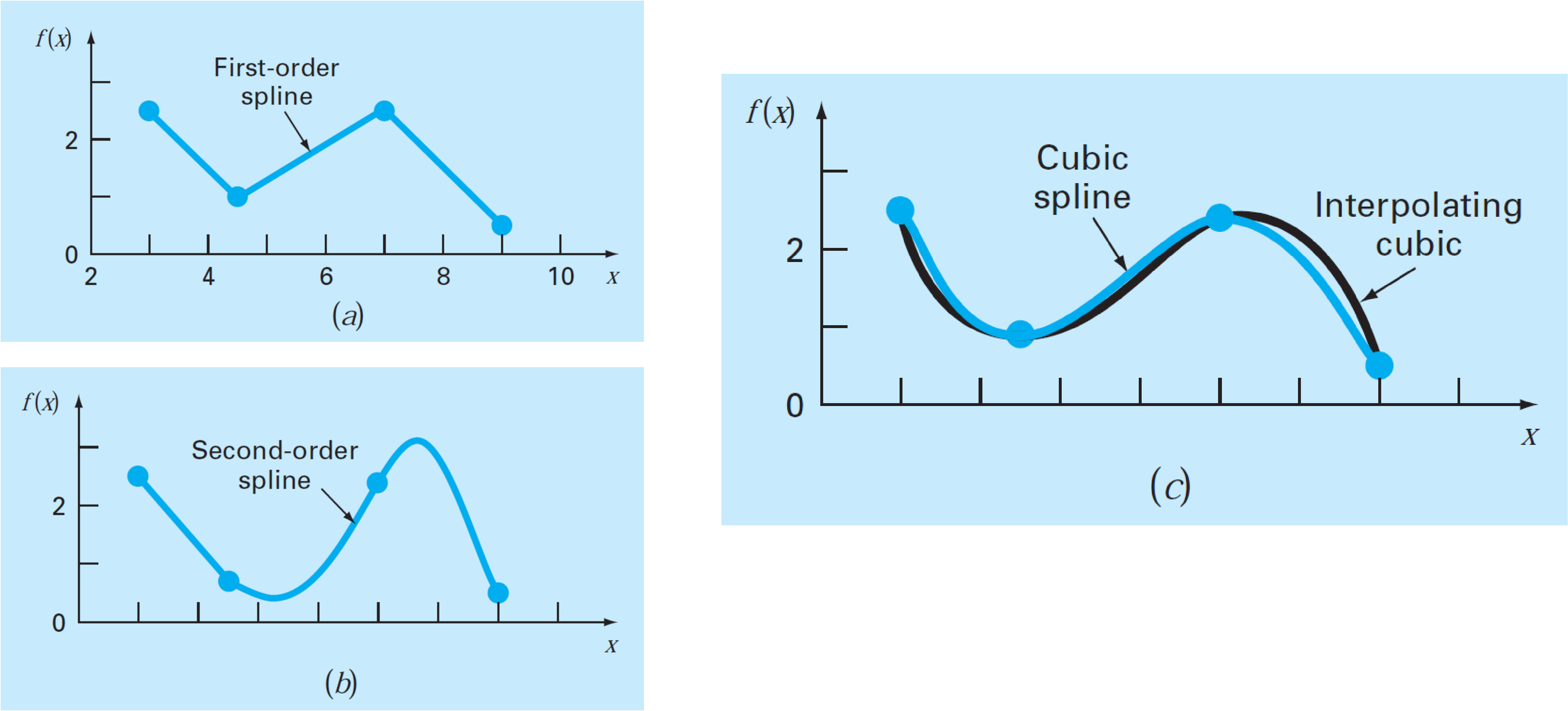
$$f(x) = f(x_1) + m_1(x - x_1) \qquad x_1 \le x \le x_2$$

$$\vdots$$

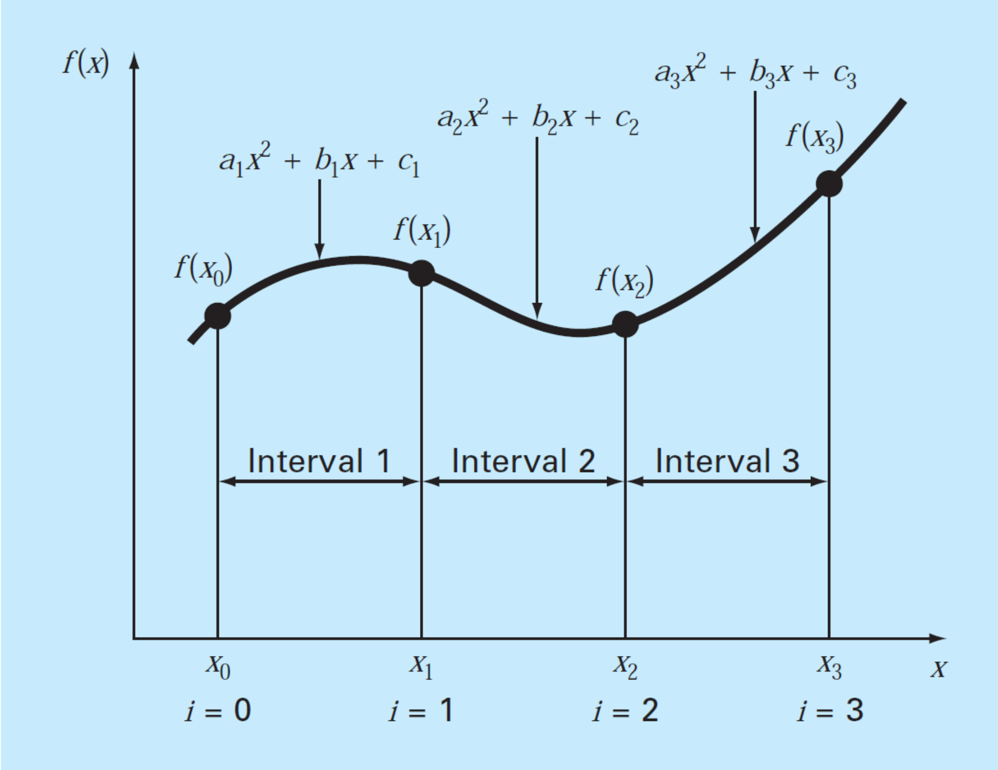
$$f(x) = f(x_{n-1}) + m_{n-1}(x - x_{n-1}) \qquad x_{n-1} \le x \le x_n$$

$$m_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

Quadratic Splines



Quadratic Splines



The objective in quadratic splines is to derive a second-order polynomial for each interval between data points. The polynomial for each interval can be represented generally as

$$f_i(x) = a_i x^2 + b_i x + c_i$$

1. *The function values of adjacent polynomials must be equal at the interior knots.* This condition can be represented as

$$a_{i-1}x_{i-1}^2 + b_{i-1}x_{i-1} + c_{i-1} = f(x_{i-1})$$

$$a_i x_{i-1}^2 + b_i x_{i-1} + c_i = f(x_{i-1})$$

2. *The first and last functions must pass through the end points.* This adds two additional equations:

$$a_1 x_0^2 + b_1 x_0 + c_1 = f(x_0)$$

$$a_n x_n^2 + b_n x_n + c_n = f(x_n)$$

3. *The first derivatives at the interior knots must be equal.*

$$f'(x) = 2ax + b$$

$$2a_{i-1}x_{i-1} + b_{i-1} = 2a_i x_{i-1} + b_i$$

4. *Assume that the second derivative is zero at the first point.*

$$a_1 = 0$$

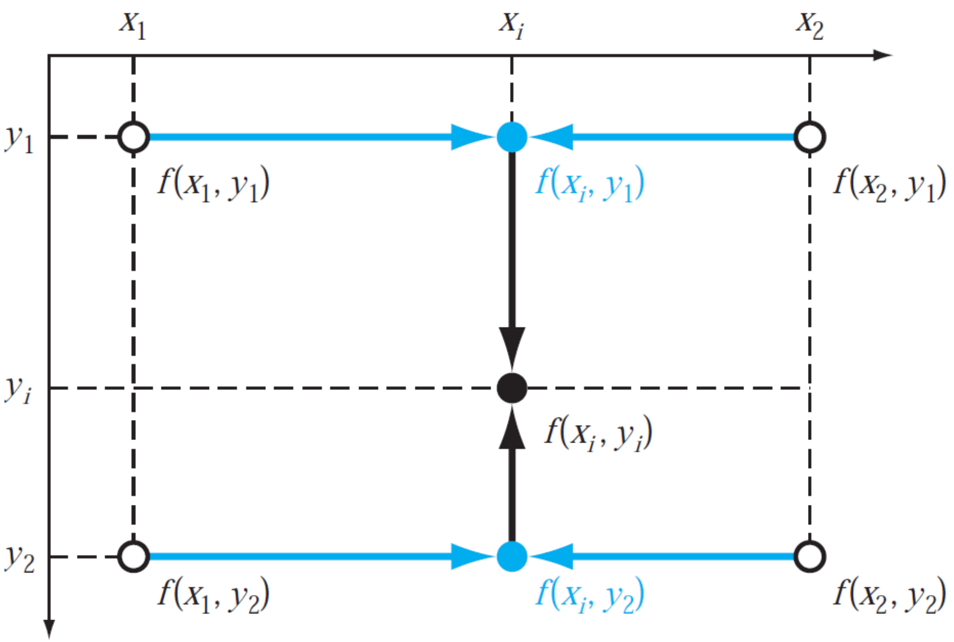
Cubic Splines

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$

Thus, for $n + 1$ data points ($i = 0, 1, 2, \ldots, n$), there are n intervals and, consequently, $4n$ unknown constants to evaluate. Just as for quadratic splines, $4n$ conditions are required to evaluate the unknowns. These are:

1. The function values must be equal at the interior knots ($2n - 2$ conditions).
2. The first and last functions must pass through the end points (2 conditions).
3. The first derivatives at the interior knots must be equal ($n - 1$ conditions).
4. The second derivatives at the interior knots must be equal ($n - 1$ conditions).
5. The second derivatives at the end knots are zero (2 conditions).

MULTIDIMENSIONAL INTERPOLATION



$$f(x_i, y_1) = \frac{x_i - x_2}{x_1 - x_2} f(x_1, y_1) + \frac{x_i - x_1}{x_2 - x_1} f(x_2, y_1)$$

$$f(x_i, y_2) = \frac{x_i - x_2}{x_1 - x_2} f(x_1, y_2) + \frac{x_i - x_1}{x_2 - x_1} f(x_2, y_2)$$

$$f(x_i, y_i) = \frac{y_i - y_2}{y_1 - y_2} f(x_i, y_1) + \frac{y_i - y_1}{y_2 - y_1} f(x_i, y_2)$$

$$f(x_i, y_i) = \frac{x_i - x_2}{x_1 - x_2}\frac{y_i - y_2}{y_1 - y_2} f(x_1, y_1) + \frac{x_i - x_1}{x_2 - x_1}\frac{y_i - y_2}{y_1 - y_2} f(x_2, y_1) + \frac{x_i - x_2}{x_1 - x_2}\frac{y_i - y_1}{y_2 - y_1} f(x_1, y_2) + \frac{x_i - x_1}{x_2 - x_1}\frac{y_i - y_1}{y_2 - y_1} f(x_2, y_2)$$

Bilinear Interpolation

Problem Statement. Suppose you have measured temperatures at a number of coordinates on the surface of a rectangular heated plate:

$$T(2, 1) = 60 \qquad T(9, 1) = 57.5$$

$$T(2, 6) = 55 \qquad T(9, 6) = 70$$

Use bilinear interpolation to estimate the temperature at $x_i = 5.25$ and $y_i = 4.8$.

$$f(5.5, 4) = \frac{5.25 - 9}{2 - 9}\,\frac{4.8 - 6}{1 - 6}60 + \frac{5.25 - 2}{9 - 2}\,\frac{4.8 - 6}{1 - 6}57.5$$

$$+ \frac{5.25 - 9}{2 - 9}\,\frac{4.8 - 1}{6 - 1}55 + \frac{5.25 - 2}{9 - 2}\,\frac{4.8 - 1}{6 - 1}70 = 61.2143$$

Matlab functions

- interp1 – 1-D linear interpolation
- interp2 – 2-D linear interpolation

1-D interpolation

- **yi = interp1(x,y,xi,'linear')**
- **yi = interp1(x,y,xi,'cubic')** – shape-preserving
- **yi = interp1(x,y,xi,'spline')**
- x,y=data vectors
- xi is vector of interpolation points

Script

```
time=0:10;
temps=[72.5 78.1 86.4 92.3 110.6 111.5 109.3
   110.2 110.5 109.9 110.2];
plot(time,temps,'o')
xlabel('Time (s)')
ylabel('Temperature (F)')
plotvals=0:0.1:10;
yvals=interp1(time,temps,plotvals,'linear')
hold on
plot(plotvals,yvals)
yvals=interp1(time,temps,plotvals,'cubic')
plot(plotvals,yvals,'r')
yvals=interp1(time,temps,plotvals,'spline')
plot(plotvals,yvals,'g')
```

Time (s)	Temperature (F)
0	72.5
1	78.1
2	86.4
3	92.3
4	110.6
5	111.5
6	109.3
7	110.2
8	110.5
9	109.9
10	110.2

Practice

- Computer controlled machines are used to shape a car fender
- Use interpolation to define the entire fender

Fender Data

X (ft)	0	.25	.75	1.25	1.5	1.75	1.875	2	2.125	2.25
Y	1.2	1.18	1.1	1	0.92	0.8	0.7	0.55	0.35	0

The following data defines the sea-level concentration of dissolved oxygen for fresh water as a function of temperature:

T, °C	0	8	16	24	32	40
o, mg/L	14.621	11.843	9.870	8.418	7.305	6.413

Estimate $o(27)$ using **(a)** linear interpolation, **(b)** Newton's interpolating polynomial, and **(c)** cubic splines. Note that the exact result is 7.986 mg/L.

Generate eight equally-spaced points from the function

$$f(t) = \sin^2 t$$

from $t = 0$ to 2π. Fit this data with **(a)** a seventh-order interpolating polynomial and **(b)** a cubic spline.

Practice – Trace of My hand

- Download and run **handdata.m**
- Plot x vs. y
- Let t=1:76
- Interpolate x vs. t and y vs. t
- Now plot curve for hand vs. data