MATLAB 与工程应用

Root-Finding

Curve Fitting

Mathematical modeling

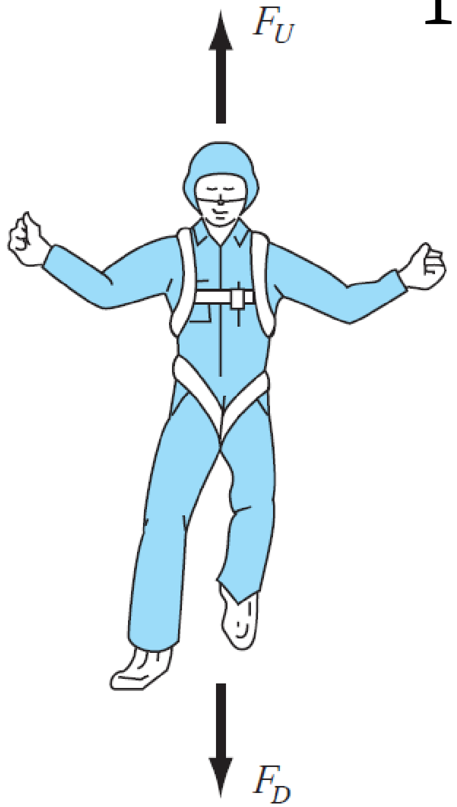


FIGURE 1.2
Schematic diagram of the forces acting on a falling parachutist. F_D is the downward force due to gravity. F_U is the upward force due to air resistance.

$$F = ma$$

$$a = \frac{F}{m}$$

$$\frac{dv}{dt} = \frac{F}{m}$$

$$F = F_D + F_U$$

$$F_D = mg$$

$$F_U = -cv$$

$$\frac{dv}{dt} = \frac{mg - cv}{m}$$

$$v(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$$

$$f(c) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right) - v$$

Type of Mathematical Problem

Roots of equations.
Systems of simultaneous linear algebraic equations.
Optimization.
Curve fitting.
Numerical integration.
Ordinary differential equations.
Partial differential equations.

Root-Finding

Case Study

Vibrations of a Satellite Boom

$$y(x) = A\cosh(\beta x) + B\sinh(\beta x) + C\cos(\beta x) + D\sin(\beta x)$$

$$y(0) = \frac{dy}{dx}(0) = \frac{d^2y}{dx^2}(L) = \frac{d^3y}{dx^3}(L) = 0$$

4th order ordinary differential equation

Apply boundary conditions

Solve for roots and then for frequencies

The Approach

$$y(x) = A\cosh(\beta x) + B\sinh(\beta x) + C\cos(\beta x) + D\sin(\beta x)$$

$$y(0) = \frac{dy}{dx}(0) = \frac{d^2y}{dx^2}(L) = \frac{d^3y}{dx^3}(L) = 0$$

- Apply boundary conditions
- Eliminate A, B, C, D
- Obtain equation for β in terms of L
- Solve for β, then for frequencies

Resulting Equation for Frequencies

$$\frac{\cosh(\beta L)+\cos(\beta L)}{\sinh(\beta L)-\sin(\beta L)}=\frac{\sinh(\beta L)+\sin(\beta L)}{\cosh(\beta L)+\cos(\beta L)}$$

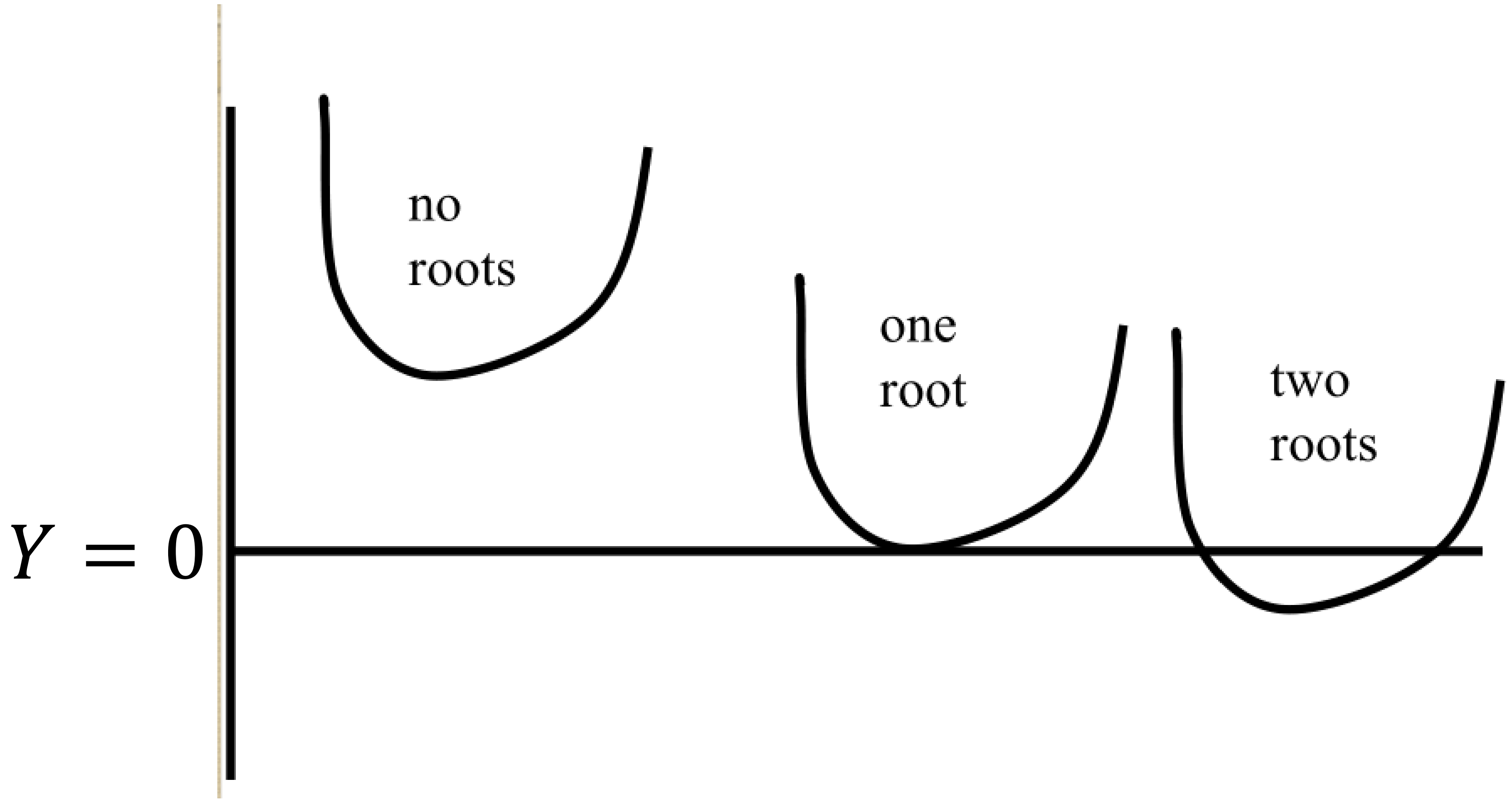
or

$$1+\cosh(\beta L)\cos(\beta L)=0$$

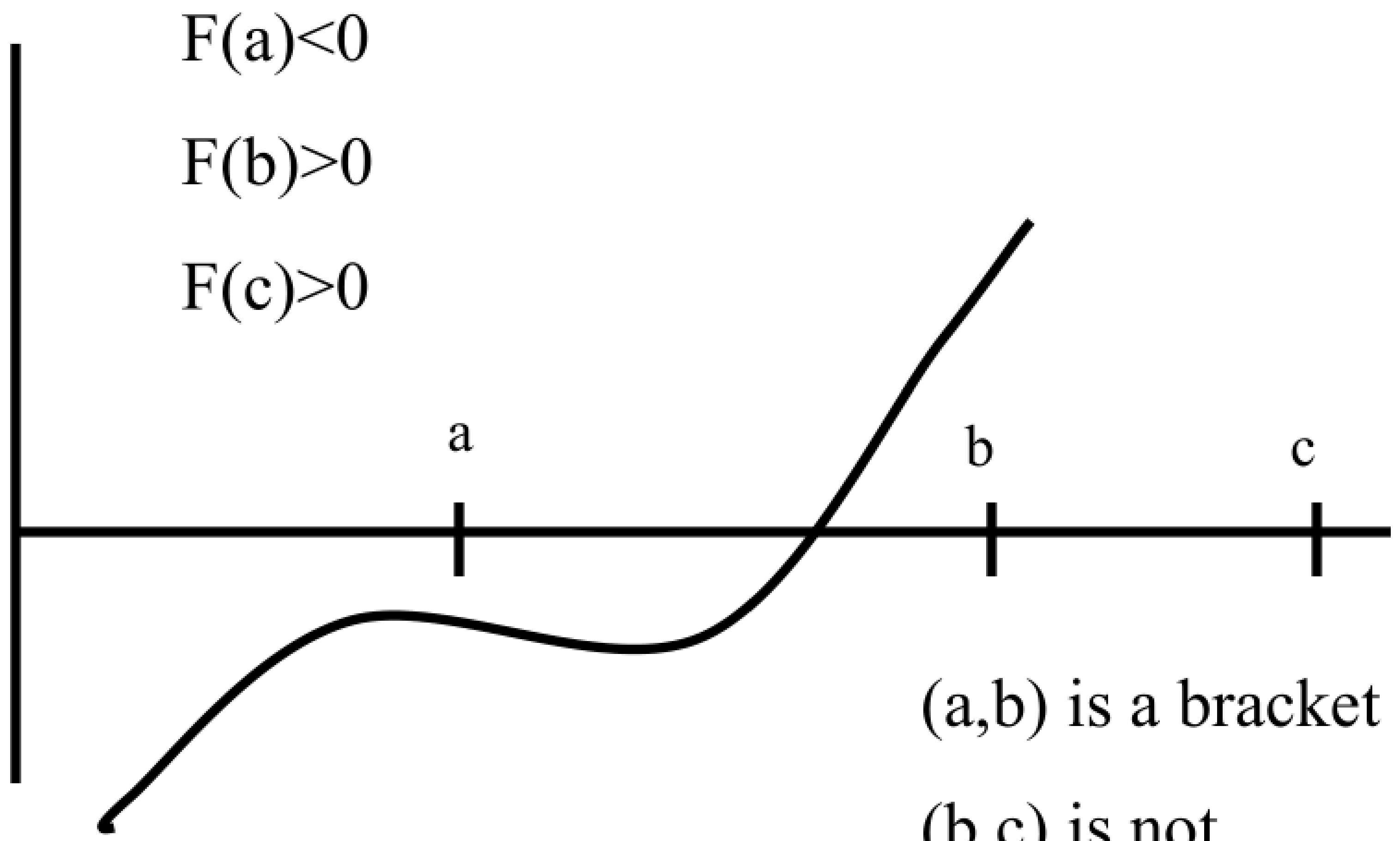
Root Finding

- Goal is to find x such that f(x)=0
- These values of x are called roots or zeroes
- There may be none, 1, more than 1, or an infinite number of roots
- Always plot function first

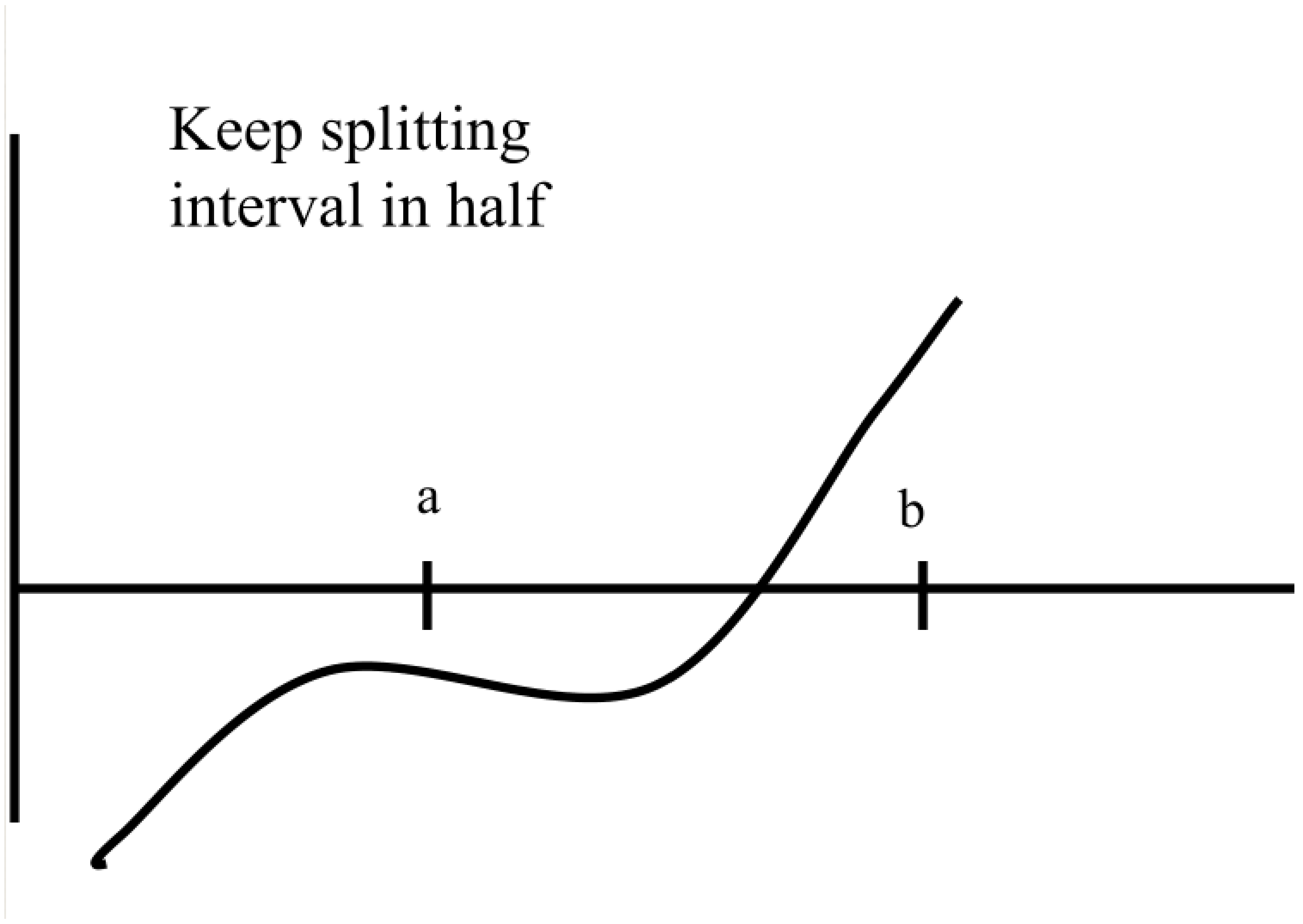
Some Possible scenarios



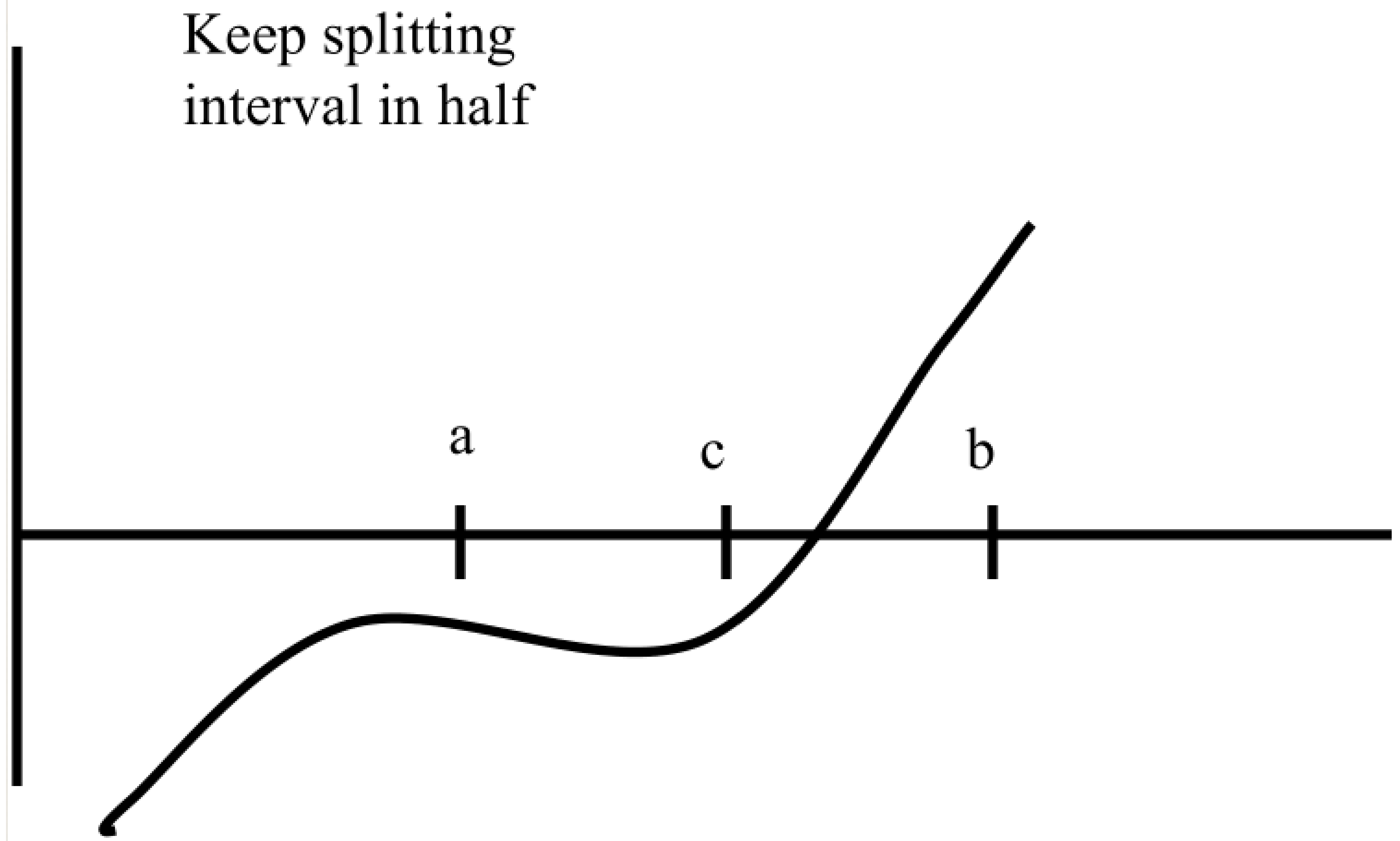
Bracketing



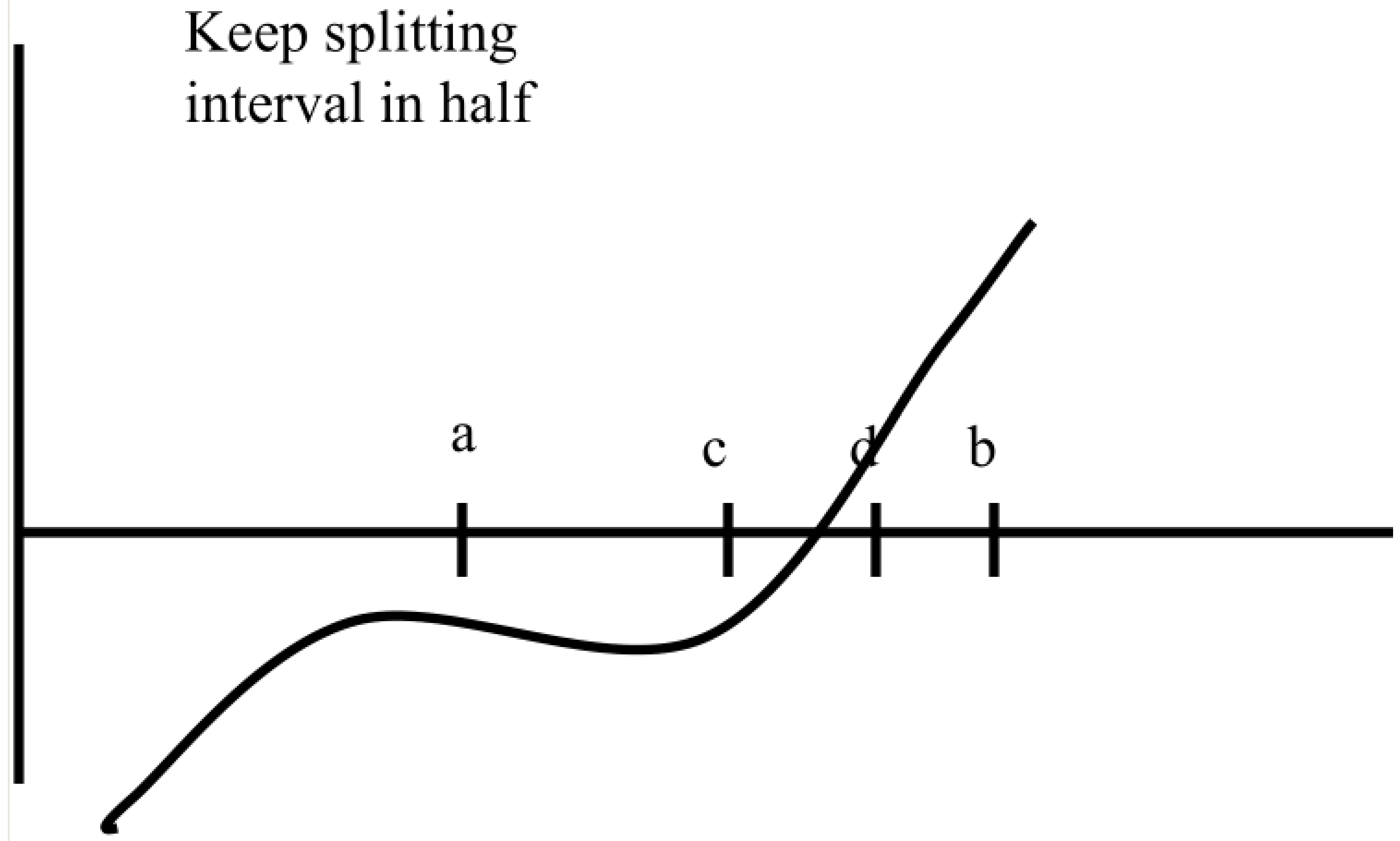
Bisection



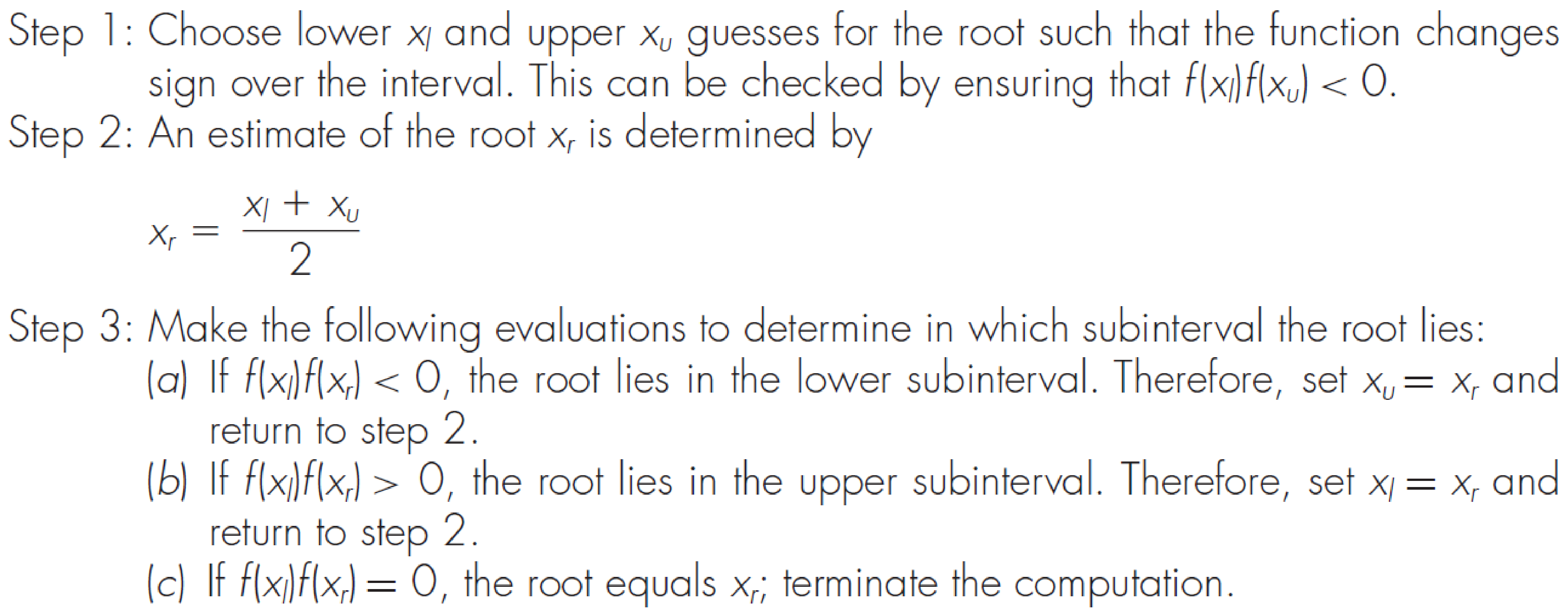
Bisection



Bisection



Bisection



Step 1: Choose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval. This can be checked by ensuring that $f(x_l)f(x_u) < 0$.

Step 2: An estimate of the root x_r is determined by

$$x_r = \frac{x_l + x_u}{2}$$

Step 3: Make the following evaluations to determine in which subinterval the root lies:

(*a*) If $f(x_l)f(x_r) < 0$, the root lies in the lower subinterval. Therefore, set $x_u = x_r$ and return to step 2.

(*b*) If $f(x_l)f(x_r) > 0$, the root lies in the upper subinterval. Therefore, set $x_l = x_r$ and return to step 2.

(*c*) If $f(x_l)f(x_r) = 0$, the root equals x_r; terminate the computation.

Pseudocode for function to implement bisection

```
FUNCTION Bisect(xl, xu, es, imax, xr, iter, ea)
  iter = 0
  DO
    xrold = xr
    xr = (xl + xu) / 2
    iter = iter + 1
    IF xr ≠ 0 THEN
      ea = ABS((xr − xrold) / xr) * 100
    END IF
    test = f(xl) * f(xr)
    IF test < 0 THEN
      xu = xr
    ELSE IF test > 0 THEN
      xl = xr
    ELSE
      ea = 0
    END IF
    IF ea < es OR iter ≥ imax EXIT
  END DO
  Bisect = xr
END Bisect
```

Although bisection is perfectly valid technique for determining roots, its “brute-force” approach is relatively inefficient.

Bracketing

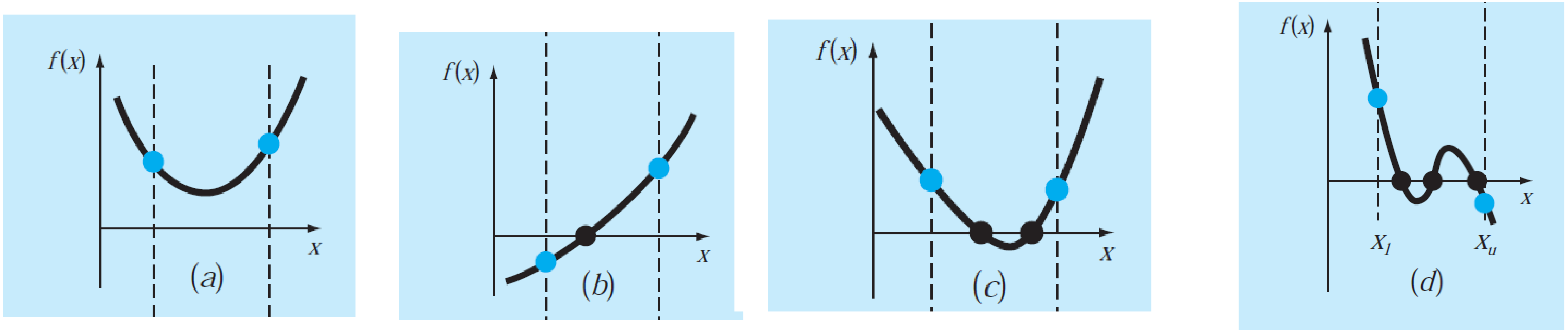
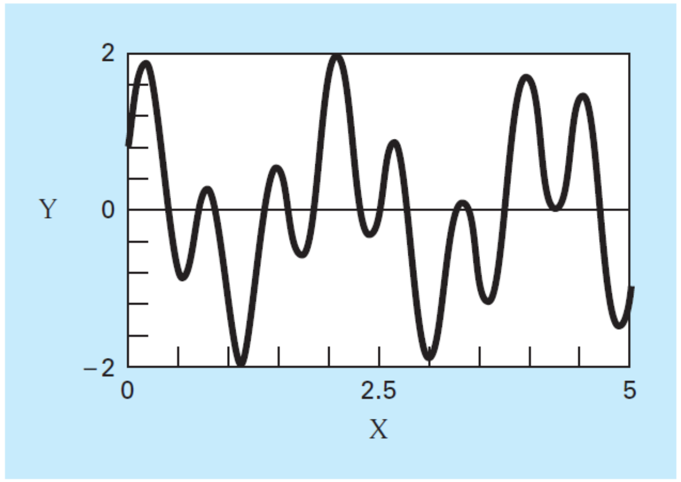


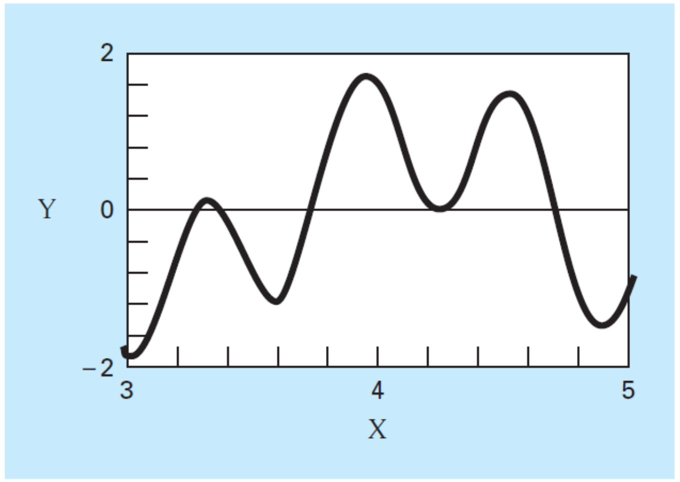
Illustration of a number of general ways that a root may occur in an interval prescribed by a lower bound xl and an upper bound xu. Parts (a) and (c) indicate that if both f(xl) and f (xu) have the same sign, either there will be no roots or there will be an even number of roots withing the interval. Parts(b) and (d) indicate that if the function has different signs at the end points, there will be an odd number of roots in the interval

Graphical methods

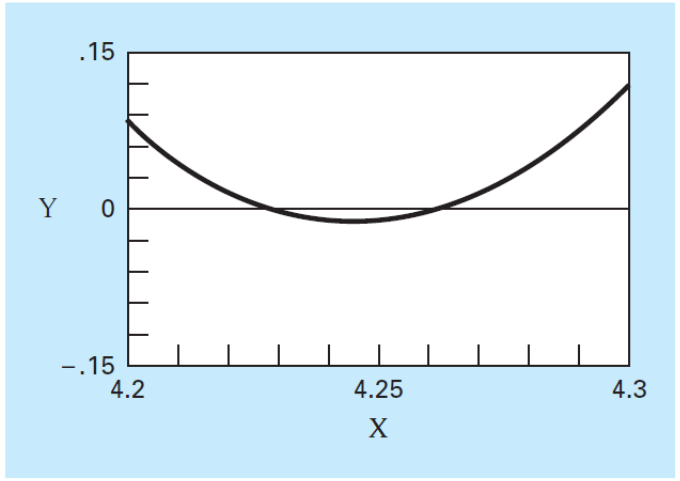
The progressive enlargement of $f(x) = \sin 10x + \cos 3x$ by the computer. Such interactive graphics permits the analyst to determine that two distinct roots exist between $x = 4.2$ and $x = 4.3$.



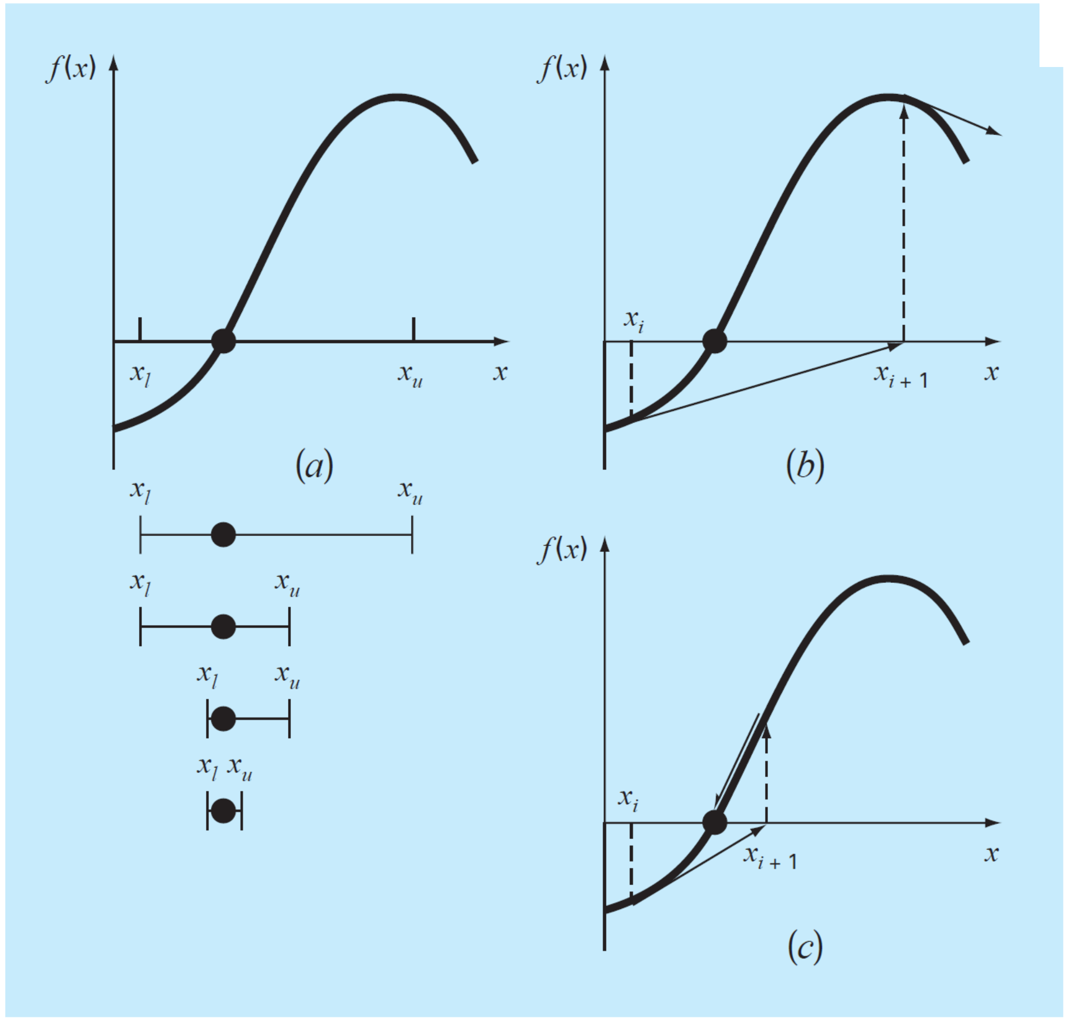
(*a*)



(*b*)



Open Methods



$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

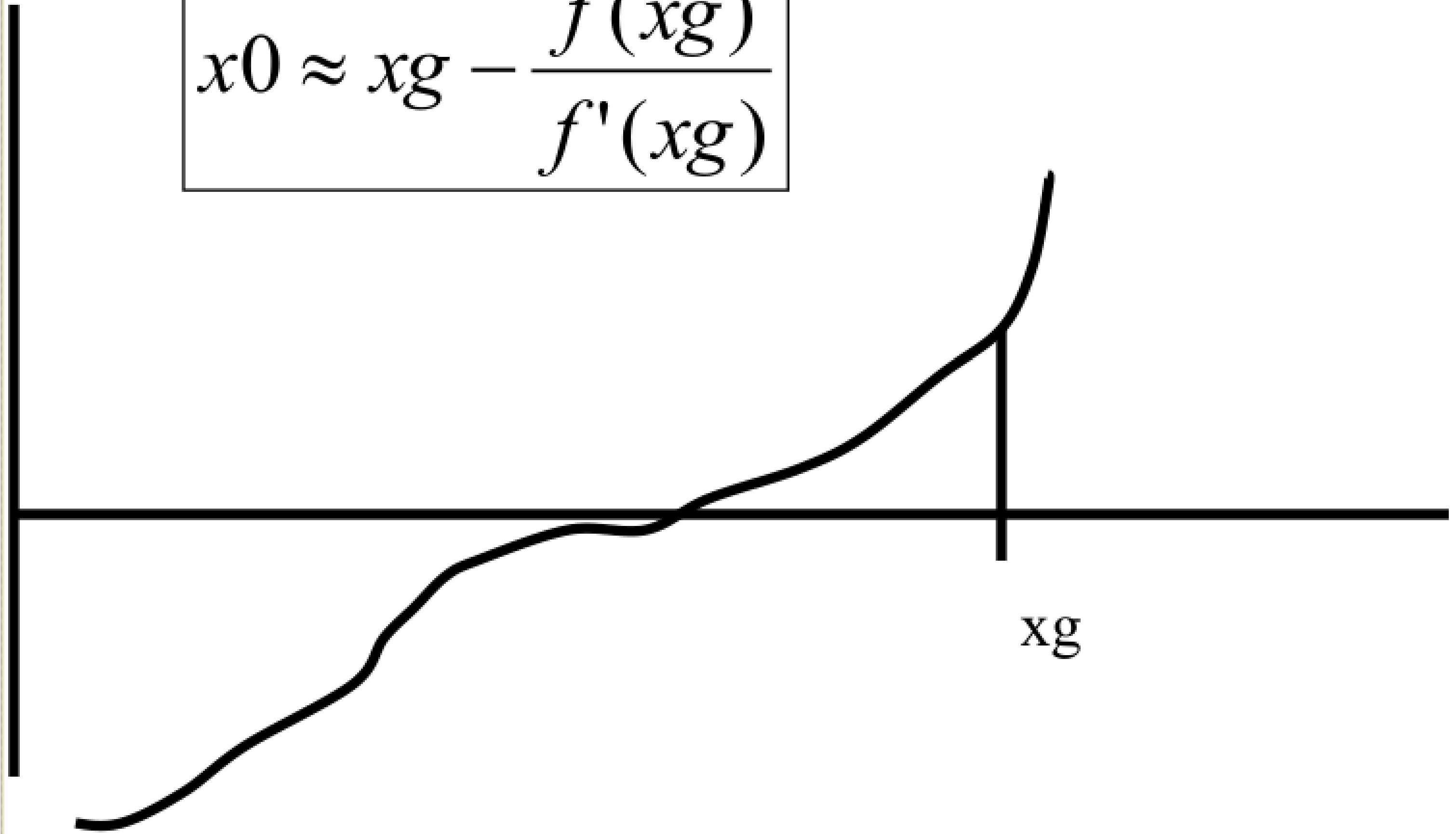
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

The open methods are based on formulas that require only a single starting value of x or two starting values that do not necessarily bracket the root. They sometimes diverge or move away from the true root as the computation progresses. However, when the open methods converge, they usually do so much more quickly than the bracketing methods.

(a) Bracketing
(b) (c) open methods

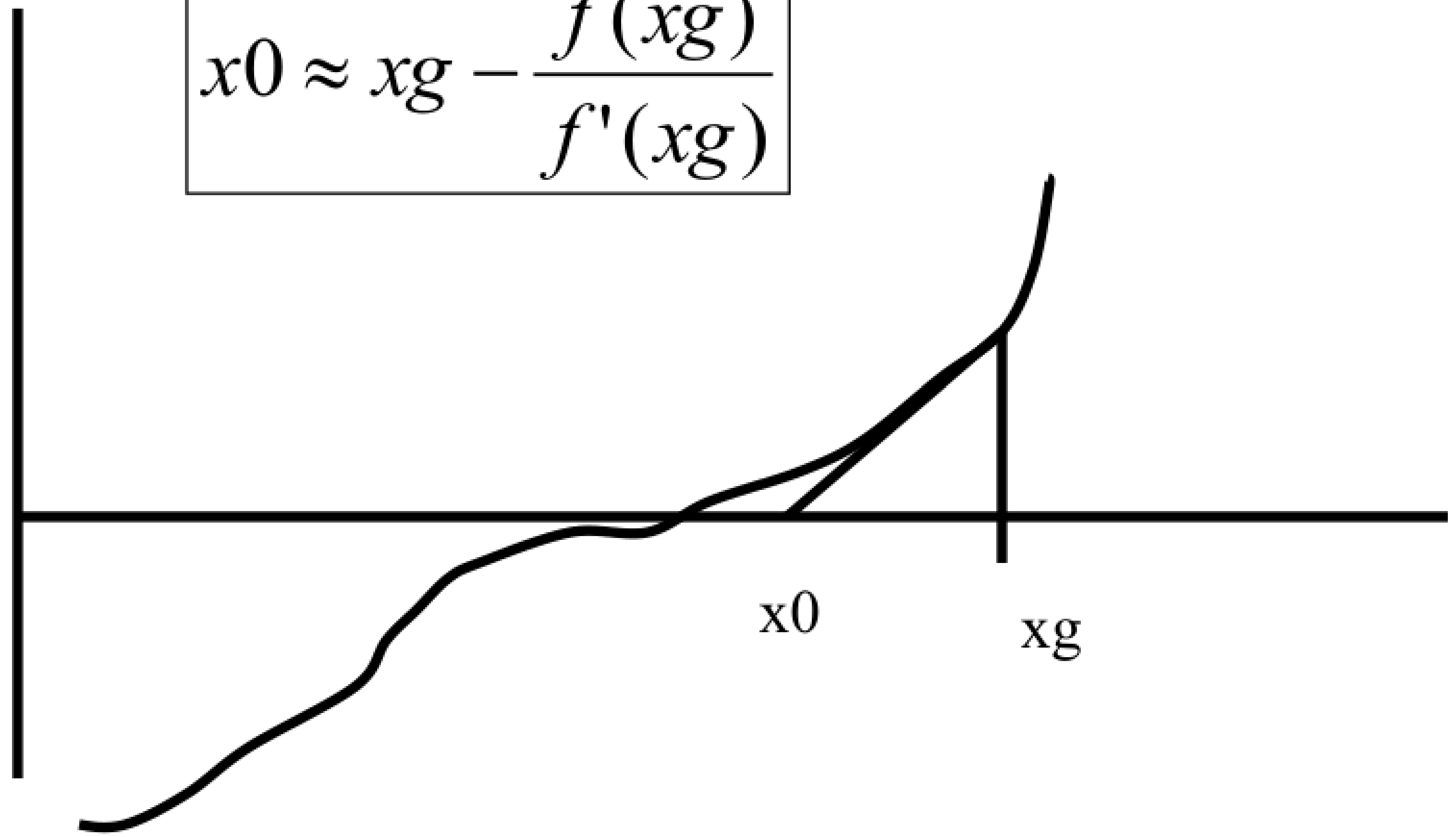
Newton’s Method

$$\boxed{x0 \approx xg - \frac{f(xg)}{f'(xg)}}$$



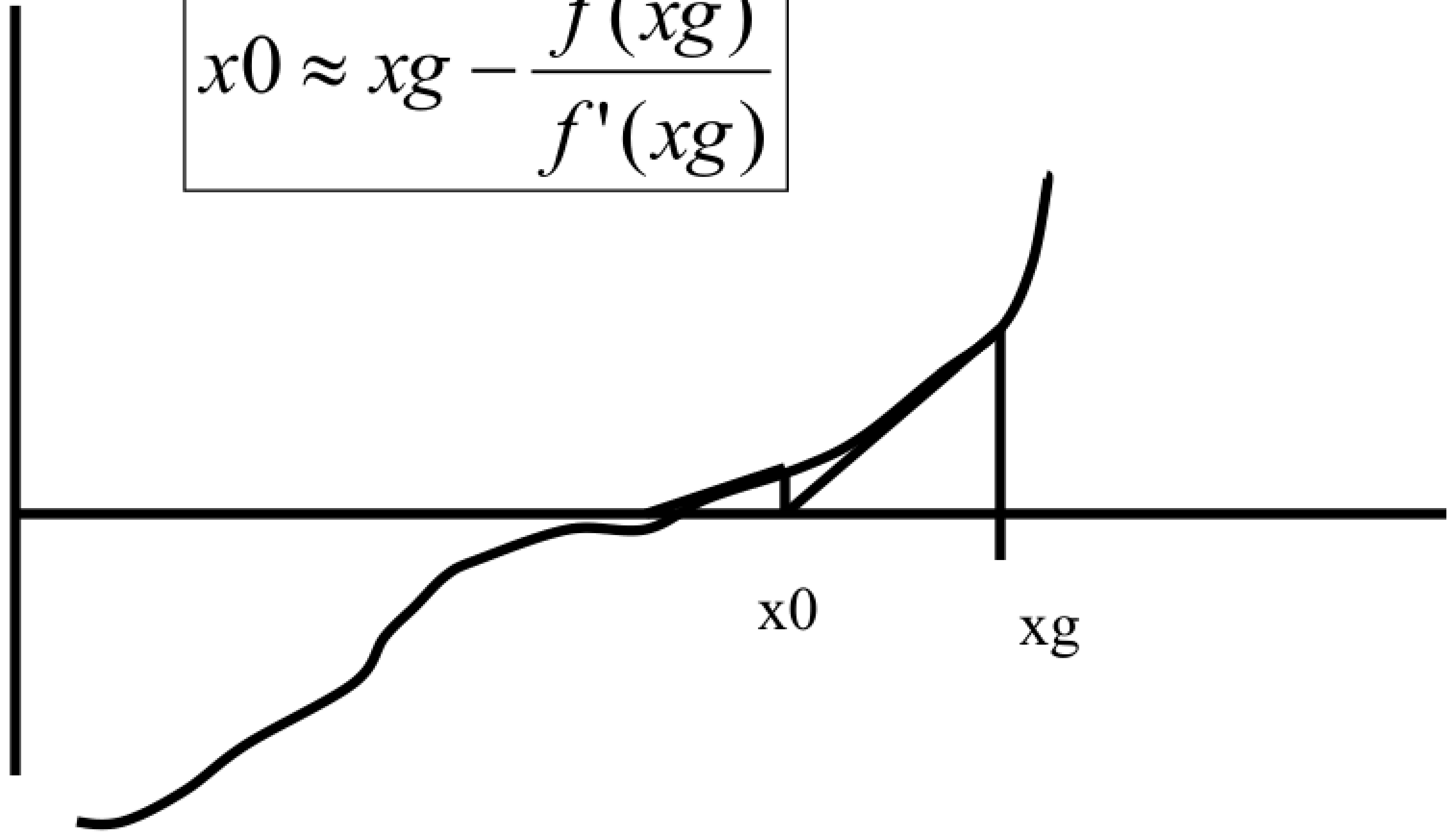
Newton's Method

$$\boxed{x0 \approx xg - \frac{f(xg)}{f'(xg)}}$$

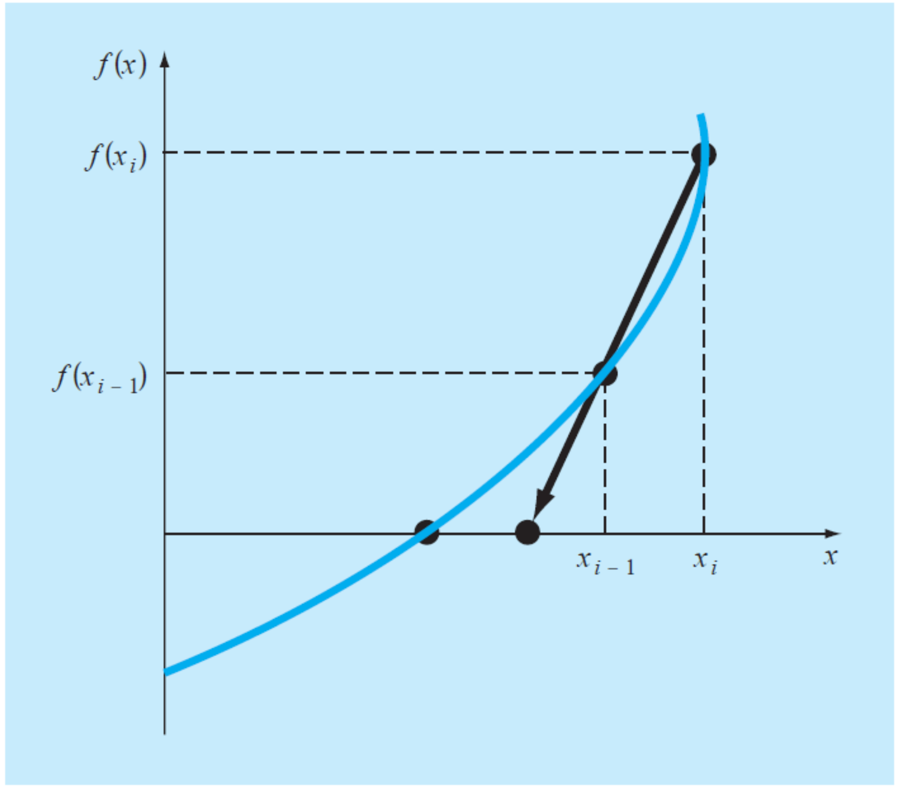


Newton’s Method

$$\boxed{x0 \approx xg - \frac{f(xg)}{f'(xg)}}$$



The secant method



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

Graphical depiction of the secant method. This technique is similar to the Newton-Raphson technique (Fig. 6.5) in the sense that an estimate of the root is predicted by extrapolating a tangent of the function to the *x* axis. However, the secant method uses a difference rather than a derivative to estimate the slope.

Brent’ s method

- Brent’s root-location method is a clever algorithm that combines the reliability of bracketing with the speed of the open methods.

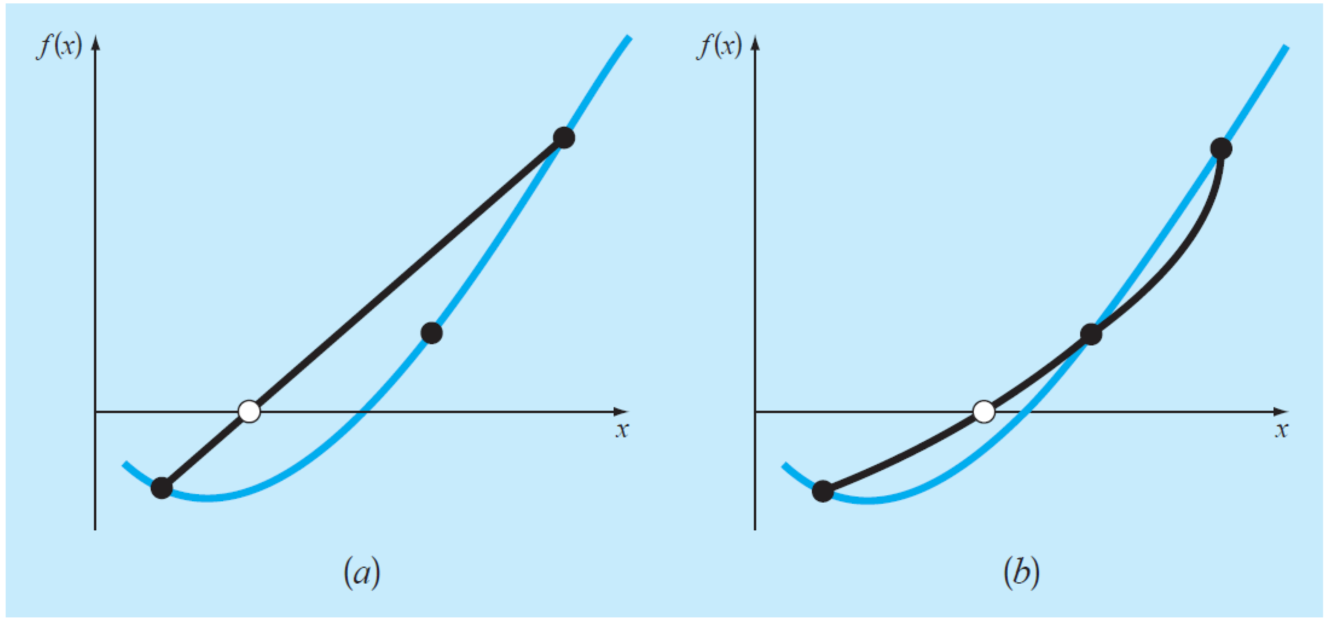
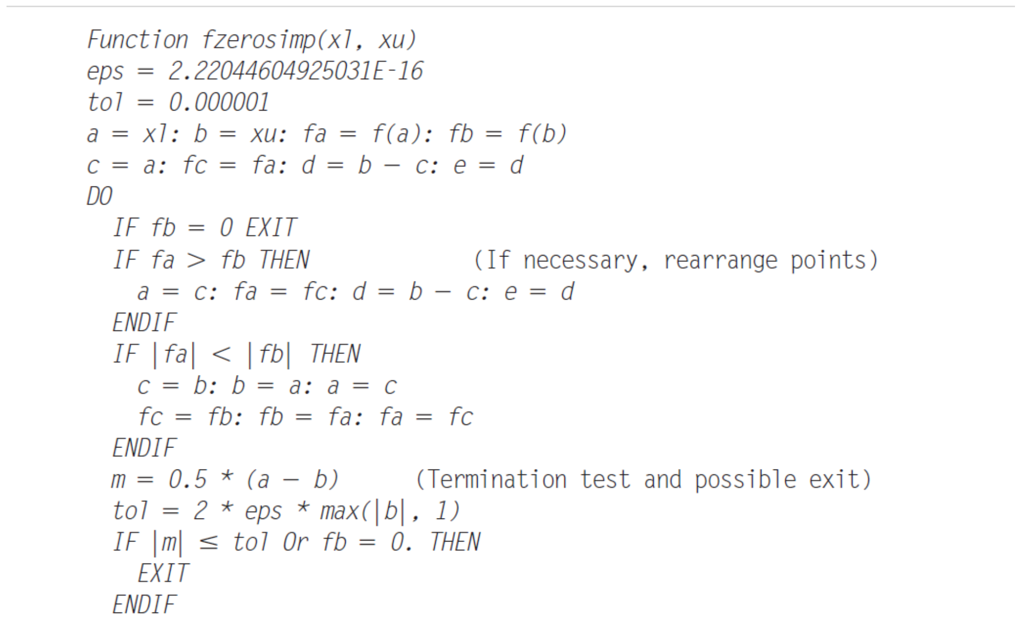


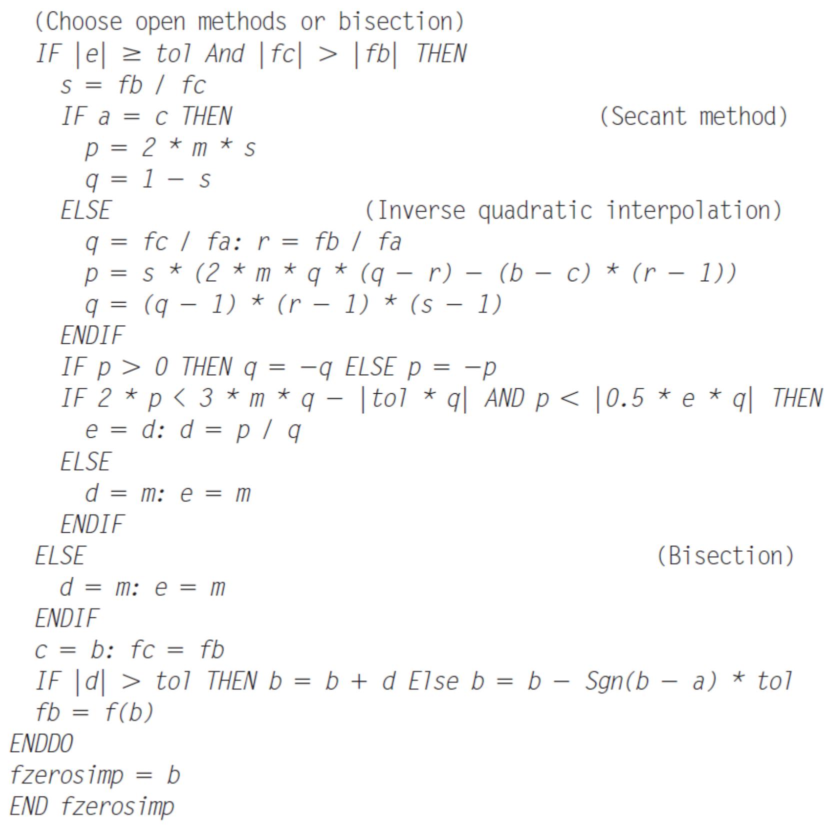
FIGURE 6.10
Comparison of (*a*) the secant method and (*b*) inverse quadratic interpolation. Note that the dark parabola passing through the three points in (*b*) is called “inverse” because it is written in *y* rather than in *x*.

$$g(y) = \frac{(y - y_{i-1})(y - y_i)}{(y_{i-2} - y_{i-1})(y_{i-2} - y_i)}x_{i-2} + \frac{(y - y_{i-2})(y - y_i)}{(y_{i-1} - y_{i-2})(y_{i-1} - y_i)}x_{i-1} + \frac{(y - y_{i-2})(y - y_{i-1})}{(y_i - y_{i-2})(y_i - y_{i-1})}x_i$$

Pseudocode for Brent’s root finding algorithm



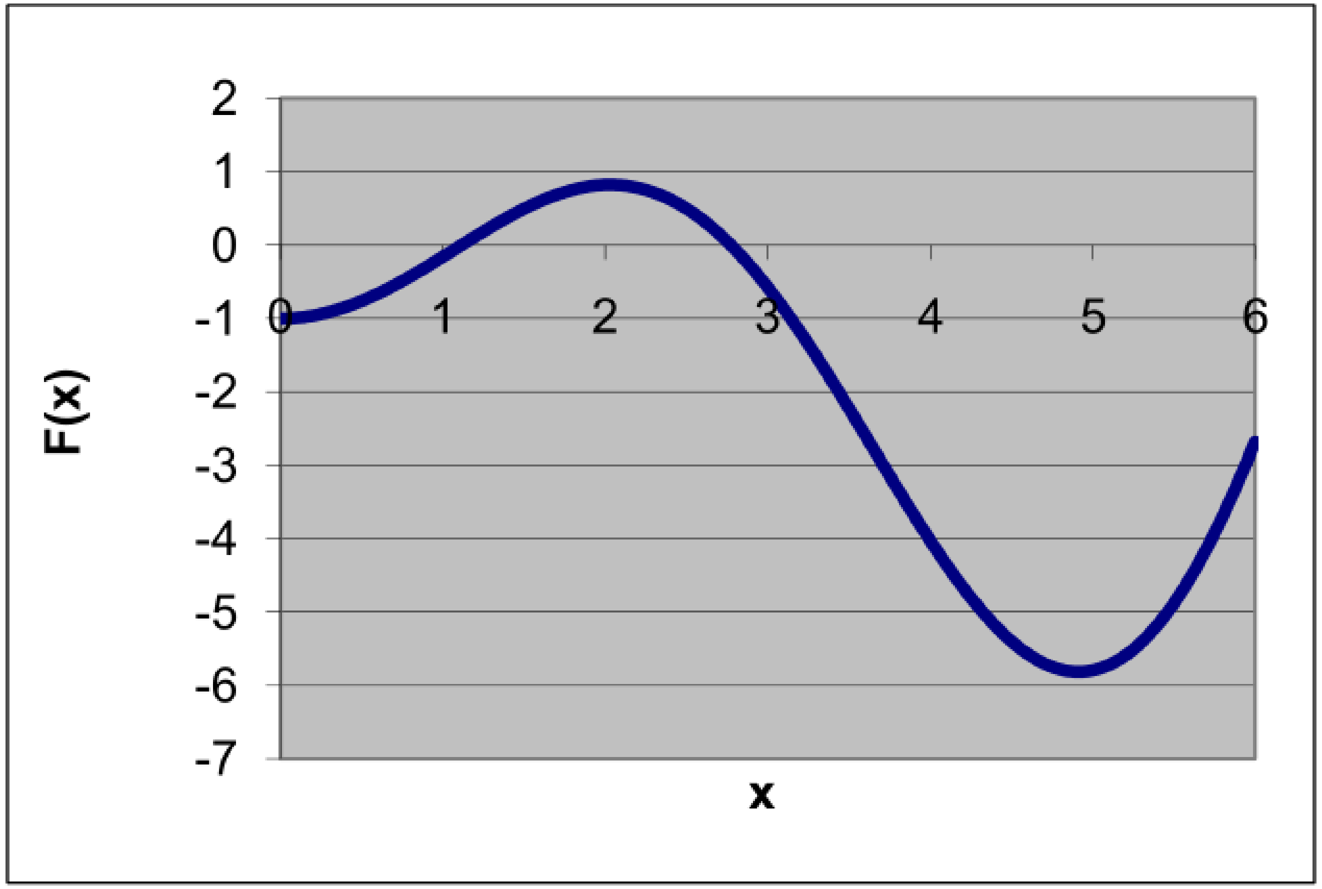
```
Function fzerosimp(xl, xu)
eps = 2.22044604925031E-16
tol = 0.000001
a = xl: b = xu: fa = f(a): fb = f(b)
c = a: fc = fa: d = b − c: e = d
DO
  IF fb = 0 EXIT
  IF fa > fb THEN                (If necessary, rearrange points)
    a = c: fa = fc: d = b − c: e = d
  ENDIF
  IF |fa| < |fb| THEN
    c = b: b = a: a = c
    fc = fb: fb = fa: fa = fc
  ENDIF
  m = 0.5 * (a − b)      (Termination test and possible exit)
  tol = 2 * eps * max(|b|, 1)
  IF |m| ≤ tol Or fb = 0. THEN
    EXIT
  ENDIF
```



```
  (Choose open methods or bisection)
  IF |e| ≥ tol And |fc| > |fb| THEN
    s = fb / fc
    IF a = c THEN                                (Secant method)
      p = 2 * m * s
      q = 1 − s
    ELSE                          (Inverse quadratic interpolation)
      q = fc / fa: r = fb / fa
      p = s * (2 * m * q * (q − r) − (b − c) * (r − 1))
      q = (q − 1) * (r − 1) * (s − 1)
    ENDIF
    IF p > 0 THEN q = −q ELSE p = −p
    IF 2 * p < 3 * m * q − |tol * q| AND p < |0.5 * e * q| THEN
      e = d: d = p / q
    ELSE
      d = m: e = m
    ENDIF
  ELSE                                              (Bisection)
    d = m: e = m
  ENDIF
  c = b: fc = fb
  IF |d| > tol THEN b = b + d Else b = b − Sgn(b − a) * tol
  fb = f(b)
ENDDO
fzerosimp = b
END fzerosimp
```

An Example

$F(x)=x*\sin(x)-1$



Matlab

- **Use fzero function**

```
function findroot
guess=1
fzero('f',guess)
%
function func=f(x)
func=x*sin(x)-1;
```

Practice

- Find roots of x*sin(x)-1 on interval 0<x<12
  - First plot and estimate roots
  - Then run fzero to get more accuracy

Practice

- The temperature of the ground at a depth x for surface temperature T_s and initial temperature T_i is given on the next slide
- How deep should a water main be buried if we want to keep the water from freezing if the surface is at -15 C for 60 days?

Parameters

- t=60*24*3600
- T_s=-15 C
- T_i=20 C
- T=0 C
- $\alpha = 1.38 * 10^{-7}$ m^2/s

$$\boxed{\frac{T - T_s}{T_i - T_s} = erf\left(\frac{x}{2\sqrt{\alpha t}}\right)}$$

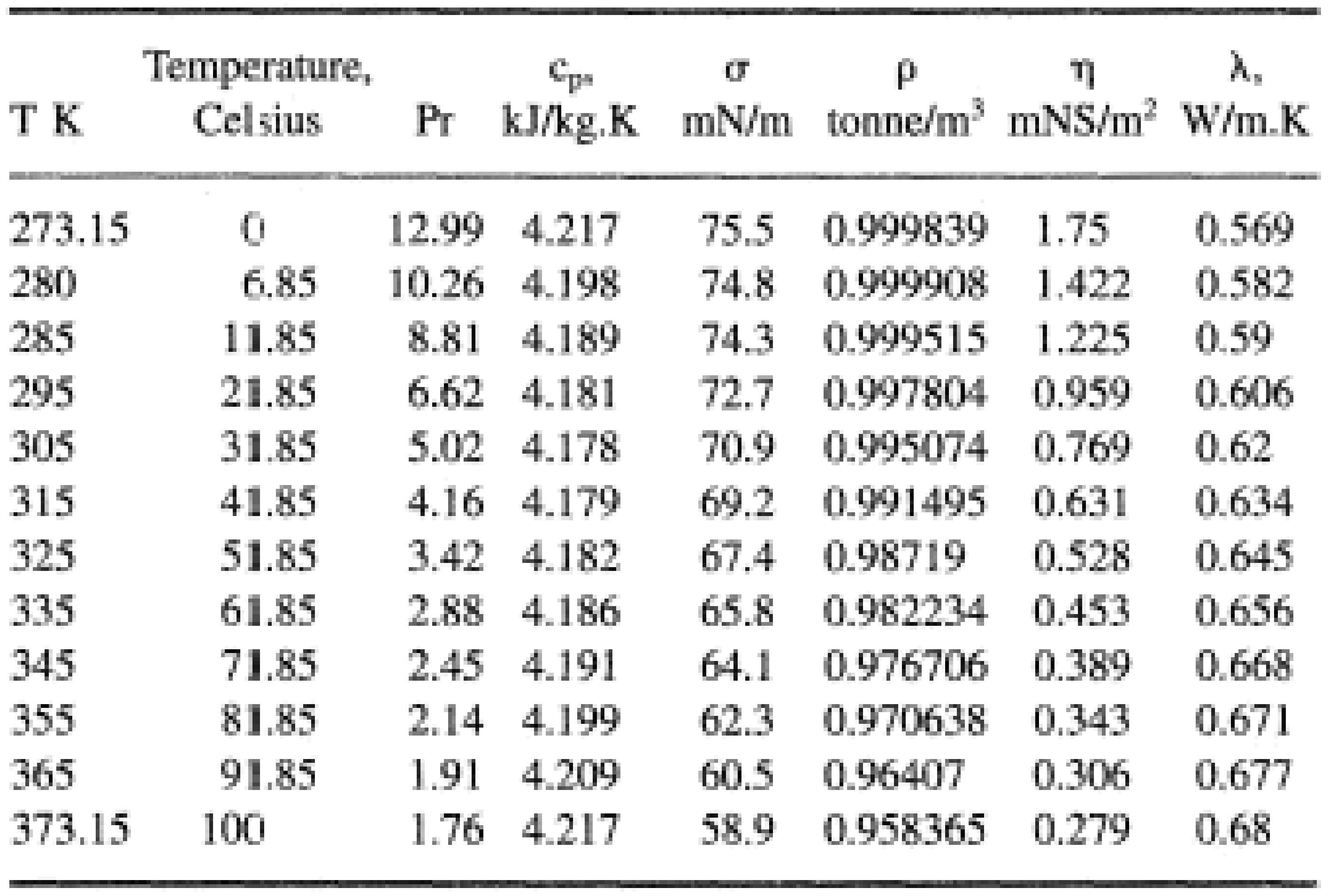
Practice

$$\boxed{1+\cosh(\beta L)\cos(\beta L)=0}$$

- Find first two positive values of β that solve this equation for L=4.2 m
- For EI=21,000 N-m^2 and rho=0.53 kg/m, calculate the frequencies from

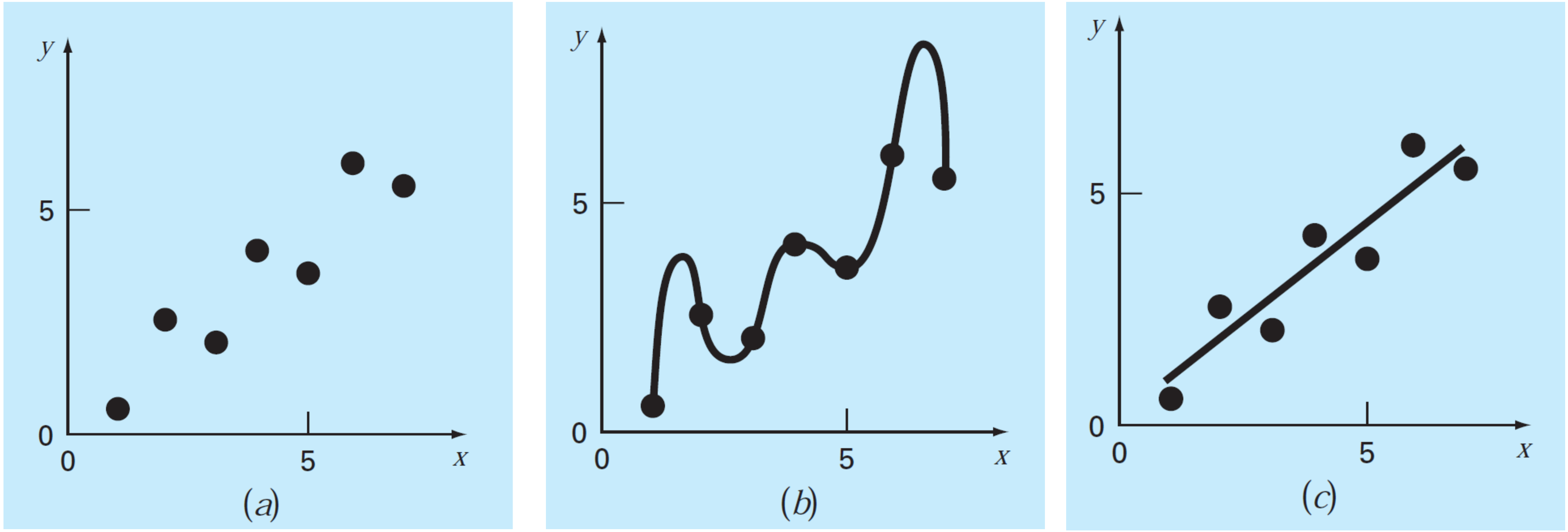
$$\boxed{\omega=\beta^2\sqrt{\frac{EI}{\rho}}}$$

Curve Fitting



T K	Temperature, Celsius	Pr	c_p, kJ/kg.K	σ mN/m	ρ tonne/m^3	η mNS/m^2	λ, W/m.K
273.15	0	12.99	4.217	75.5	0.999839	1.75	0.569
280	6.85	10.26	4.198	74.8	0.999908	1.422	0.582
285	11.85	8.81	4.189	74.3	0.999515	1.225	0.59
295	21.85	6.62	4.181	72.7	0.997804	0.959	0.606
305	31.85	5.02	4.178	70.9	0.995074	0.769	0.62
315	41.85	4.16	4.179	69.2	0.991495	0.631	0.634
325	51.85	3.42	4.182	67.4	0.98719	0.528	0.645
335	61.85	2.88	4.186	65.8	0.982234	0.453	0.656
345	71.85	2.45	4.191	64.1	0.976706	0.389	0.668
355	81.85	2.14	4.199	62.3	0.970638	0.343	0.671
365	91.85	1.91	4.209	60.5	0.96407	0.306	0.677
373.15	100	1.76	4.217	58.9	0.958365	0.279	0.68

Curve Fitting



Curve Fitting

The simplest way to fit a set of 2D data is a straight line.

Least squares regression is a method of fitting data with a line.

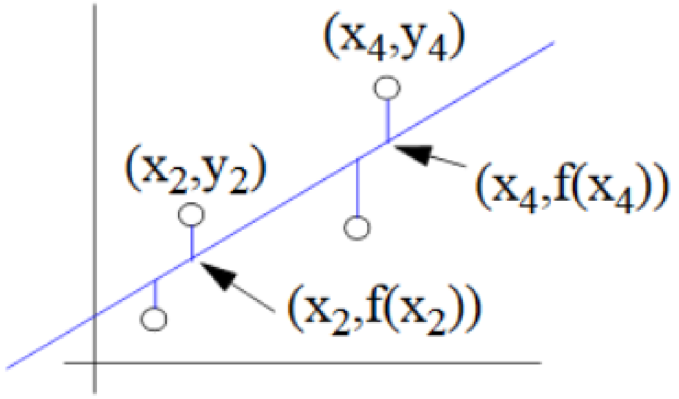
Least squares regression minimizes the squared distance between data points and the equation modeling the data points. This prevents positive and negative “errors” from canceling.

Least square regression

$$err = \sum (d_i)^2 = (y_1 - f(x_1))^2 + (y_2 - f(x_2))^2 + (y_3 - f(x_3))^2 + (y_4 - f(x_4))^2$$

Our fit is a straight line, so now substitute $f(x) = ax + b$

$$err = \sum_{i=1}^{\#\text{ data points}} (y_i - f(x_i))^2 = \sum_{i=1}^{\#\text{ data points}} (y_i - (ax_i + b))^2$$



The ‘best’ line has **minimum error** between line and data points

1) derivative describes the slope
2) slope = zero is a minimum

Least square regression

$$\frac{\partial err}{\partial a} = -2 \sum_{i=1}^{n} x_i(y_i - ax_i - b) = 0$$

$$\frac{\partial err}{\partial b} = -2 \sum^{n} (y_i - ax_i - b) = 0$$

Solve for the a and b so that the previous two equations both = 0
re-write these two equations

$$a\sum x_i^2 + b\sum x_i = \sum(x_i y_i)$$
$$a\sum x_i + b*n = \sum y_i$$

put these into **matrix form**

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum (x_i y_i) \end{bmatrix}$$

$$A = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}, \quad X = \begin{bmatrix} b \\ a \end{bmatrix}, \quad B = \begin{bmatrix} \sum y_i \\ \sum (x_i y_i) \end{bmatrix}$$

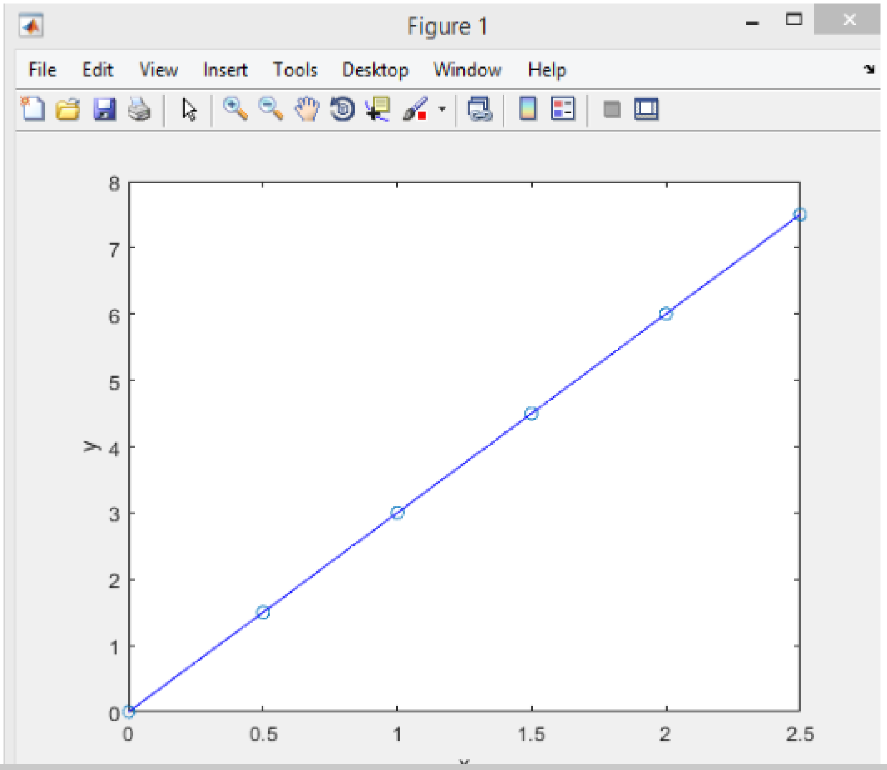
$$X = A^{-1} * B$$

Practice

Use a line to fit the following data set

i	1	2	3	4	5	6
x	0	0.5	1.0	1.5	2.0	2.5
y	0	1.5	3.0	4.5	6.0	7.5

```
clear clc
x=[0,0.5,1.0,1.5,2.0,2.5];    %x data
y=[0,1.5,3.0,4.5, 6.0,7.5];   %y data
plot(x,y,'o')
hold on
A=[length(x) sum(x); sum(x),sum(x.^2)]
%calculate matrix A
B=[sum(y);sum(x.*y)] %calculate matrix B
b=A\B; %solve the linear equation
x1=linspace(x(1),x(end),100);
plot(x1,b(2).*x1+b(1),'b') %plot the fitted
linear function
xlabel('x')
ylabel('y')
```



Polynomial Curve Fitting

Consider the general form for a polynomial of order j

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_j x^j = a_0 + \sum_{k=1}^{j} a_k x^k$$

The general expression for any error using the least squares approach is

$$err = \sum (d_i)^2 = (y_1 - f(x_1))^2 + (y_2 - f(x_2))^2 + (y_3 - f(x_3))^2 + (y_4 - f(x_4))^2$$

where we want to minimize this error. Now substitute the form of our eq.

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_j x^j = a_0 + \sum_{k=1}^{j} a_k x^k$$

into the general least squares error eq.

$$err = \sum_{i=1}^{n}\left(y_i - \left(a_0 + a_1x_i + a_2x_i^{\ 2} + a_3x_i^{\ 3} + \ldots + a_jx_i^{\ j}\right)\right)^2$$

$$err = \sum_{i=1}^{n}\left(y_i - \left(a_0 + \sum_{k=1}^{j} a_kx^k\right)\right)^2$$

[illegible]

To minimize eq.

take the derivative with respect to each coefficient $a_0, a_k \;\; k = 1, \ldots, j$ set each to zero

$$\frac{\partial err}{\partial a_0} = -2\sum_{i=1}^{n}\left(y_i - \left(a_0 + \sum_{k=1}^{j} a_k x^k\right)\right) = 0$$

$$\frac{\partial err}{\partial a_1} = -2\sum_{i=1}^{n}\left(y_i - \left(a_0 + \sum_{k=1}^{j} a_k x^k\right)\right)x = 0$$

$$\frac{\partial err}{\partial a_2} = -2\sum_{i=1}^{n}\left(y_i - \left(a_0 + \sum_{k=1}^{j} a_k x^k\right)\right)x^2 = 0$$

$$\vdots$$

$$\vdots$$

$$\frac{\partial err}{\partial a_j} = -2\sum_{i=1}^{n}\left(y_i - \left(a_0 + \sum_{k=1}^{j} a_k x^k\right)\right)x^j = 0$$

re-write these $j+1$ equations, and put into matrix form

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 & \dots & \sum x_i^j \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \dots & \sum x_i^{j+1} \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \dots & \sum x_i^{j+2} \\ \vdots & \vdots & \vdots & & \vdots \\ \sum x_i^j & \sum x_i^{j+1} & \sum x_i^{j+2} & \dots & \sum x_i^{j+j} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_j \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum (x_i y_i) \\ \sum \left(x_i^2 y_i \right) \\ \vdots \\ \sum \left(x_i^j y_i \right) \end{bmatrix}$$

$$A = \begin{bmatrix} n & \sum x_i & \sum x_i^2 & \cdots & \sum x_i^j \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \cdots & \sum x_i^{j+1} \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \cdots & \sum x_i^{j+2} \\ \vdots & \vdots & \vdots & & \vdots \\ \sum x_i^j & \sum x_i^{j+1} & \sum x_i^{j+2} & \cdots & \sum x_i^{j+j} \end{bmatrix}, \quad X = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_j \end{bmatrix}, \quad B = \begin{bmatrix} \sum y_i \\ \sum (x_i y_i) \\ \sum \left(x_i^2 y_i \right) \\ \vdots \\ \sum \left(x_i^j y_i \right) \end{bmatrix}$$

$$AX = B$$

$$X = A^{-1} * B$$

Nonlinear Curve Fitting

Linearization

Find: a function to fit data of the general exponential form $y = Ce^{Ax}$

1) Take logarithm of both sides to get rid of the exponential $\ln(y) = \ln(Ce^{Ax}) = Ax + \ln(C)$

2) Introduce the following change of variables: $Y = \ln(y)$, $X = x$, $B = \ln(C)$

Now we have: $Y = AX + B$ which is a LINEAR equation

The original data points in the $x - y$ plane get mapped into the $X - Y$ plane.

This is called data linearization. The data is transformed as: $(x, y) \Rightarrow (X, Y) = (x, \ln(y))$

Now we use the method for solving a first order linear curve fit $\begin{bmatrix} n & \sum X \\ \sum X & \sum X^2 \end{bmatrix} \begin{bmatrix} B \\ A \end{bmatrix} = \begin{bmatrix} \sum Y \\ \sum XY \end{bmatrix}$

for A and B, where above $Y = \ln(y)$, and $X = x$

Finally, we operate on $B = \ln(C)$ to solve $C = e^{B}$

And we now have the coefficients for $y = Ce^{Ax}$

We can find the values of b and m with the MATLAB polyfit function. This function finds the coefficients of a polynomial of specified degree n that best fits the data, in the so-called least-squares sense.

```
p = polyfit(x,y,n)
```

$$p(x) = p_1x^n + p_2x^{n-1} + ... + p_nx + p_{n+1}$$

Because we are assuming that our data will form a straight line on a rectilinear, semilog, or log-log plot, we are interested only in a polynomial that corresponds to a straight lien, that is, a first-degree polynomial, which we will denote as $w = p_1z + p_2$. Thus, referring to the equation above, we see the vector p will be $[p_1, p_2]$ if n is 1.

This polynomial has a different interpretation in each of the three cases:

- **The linear function:** $y = mx + b$. In this case the variables w and z in the polynomial $w = p_1 z + p_2$ are the original data variables x and y, and we can find the linear function that fits the data by typing $p = polyfit(x, y, 1)$. The first element p_1 of the vector p will be m, and the second element p_2 will be b.

- **The power function:** $y = bx^m$. In this case $log_{10}y = log_{10}x + log_{10}b$, which has the form $w = p_1 z + p_2$, where the polynomial variables w and z are related to the original data variables x and y by $w = log_{10}y$ and $z = log_{10}x$. Thus we can find the power function that fits the data by typing $p = polyfit(log10(x), log10(y), 1)$. The first element p_1 of the vector p will be m, and the second element p_2 will be $log_{10}b$. We can find b from $b = 10^{p_2}$

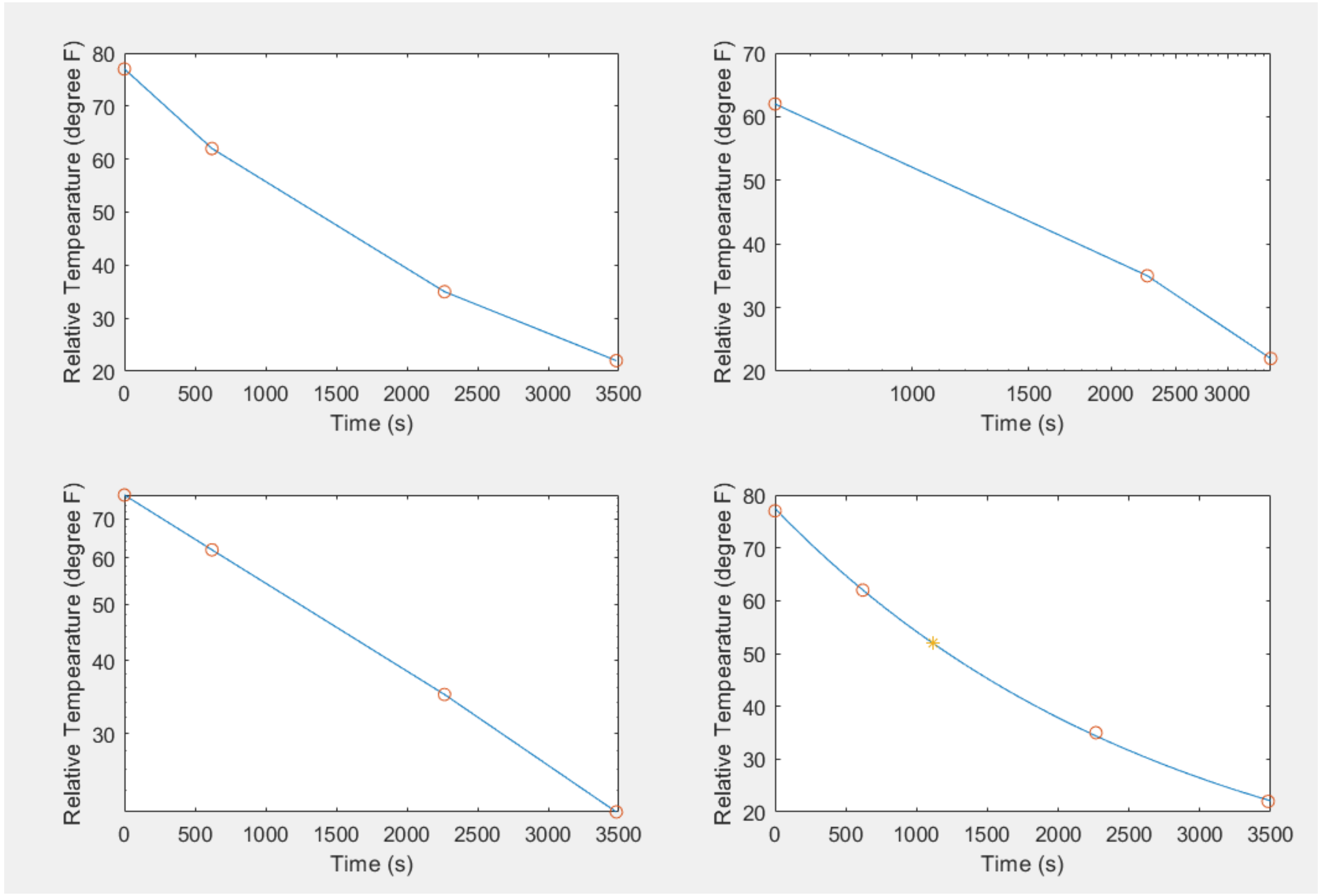
- **The exponential function:** $y = b(10)^{mx}$. In this case $log_{10}y = mx + log_{10}b$, which has the form $w = p_1 z + p_2$, where the polynomial variables w and z are related to the original data variables x and y by $w = log_{10}y$ and $z = x$. Thus we can find the exponential function that fits the data by typing $p = polyfit(x, log10(y), 1)$. The first element p_1 of the vector p will be m, and the second element p_2 will be $log_{10}b$. We can find b from $b = 10^{p_2}$

The temperature of coffee cooling in a porcelain mug at room temperature (68°F) was measured at various times. The data follow.

Time t (sec)	**Temperature T (°F)**
0	145
620	130
2266	103
3482	90

Develop a model of the coffee's temperature as a function of time, and use the model to estimate how long it took the temperature to reach 120°F.

Because $T(0)$ is finite but nonzero, the power function cannot describe these data, so we do not bother to plot the data on log-log axes. Common sense tells us that coffee will cool and its temperature will eventually equal the room temperature. So we subtract the room temperature from the data and plot the relative temperature, $T - 68$, versus time. If the relative temperature is a linear function of time, the model is $T - 68 = mt + b$. If the relative temperature is an exponential function of time, the model is $T - 68 = b(10)^{mt}$.

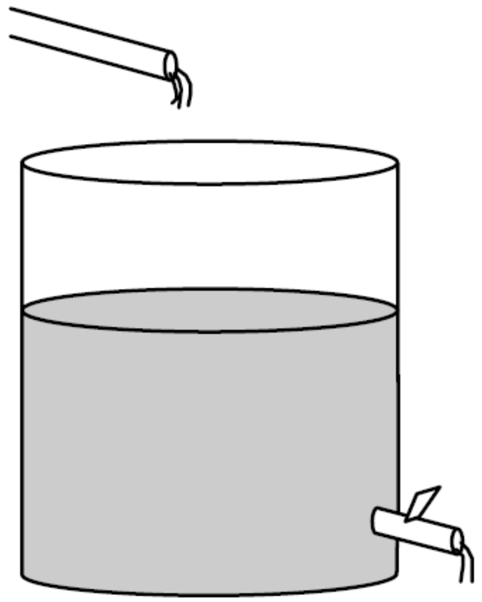


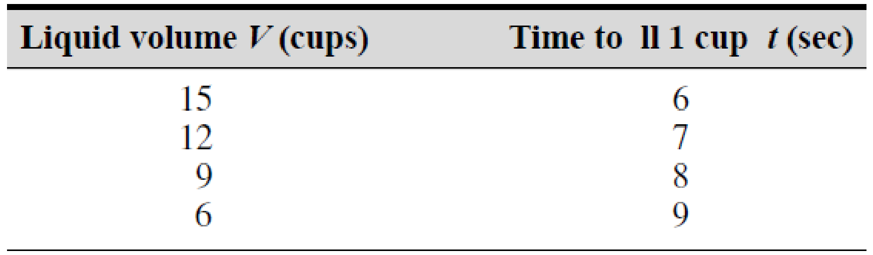
Temperature of a cooling cup of coffee, plotted on various coordinates

```
clc
clear
% coffee temperature evolution
figure (1)
t=[0, 620,2266, 3482];  % time
T=[145, 130, 103, 90];  % Corresponding temperature
RT=T-68;                % Relative temperature
subplot(2,2,1)
plot(t,RT,t,RT,'o')    % plot data
xlabel('Time (s)')
ylabel('Relative Tempearature (degree F)’)

subplot(2,2,2)
semilogx(t,RT,t,RT,'o') % logrithmic x
xlabel('Time (s)')
ylabel('Relative Tempearature (degree F)’)

subplot(2,2,3)
semilogy(t,RT,t,RT,'o') % Logrithmic y
xlabel('Time (s)')
ylabel('Relative Tempearature (degree F)')
```

```
% Through comparing the four plots, the third plot turns to be the right
% one.
p=polyfit(t,log10(RT),1); % Apply first order polynomial
m=p(1);    %get the coefficient of m
b=10^(p(2));  % get the coefficient of b
x1=linspace(0,3500,4000);
y1=b.*10.^(m.*x1);
subplot(2,2,4)
% Compute the time to reach 120oF
y_120=120-68;
x_120=(log10(y_120)-log10(b))/m;
plot(x1,y1,t,RT,'o',x_120,y_120,'*')
xlabel('Time (s)')
ylabel('Relative Tempearature (degree F)')
```

A 15-cup coffee pot was placed under a water faucet and filled to the 15-cup line. With the outlet valve open, the faucet's flow rate was adjusted until the water level remained constant at 15 cups, and the time for one cup to flow out of the pot was measured. This experiment was repeated with the pot filled to the various levels shown in the following table:



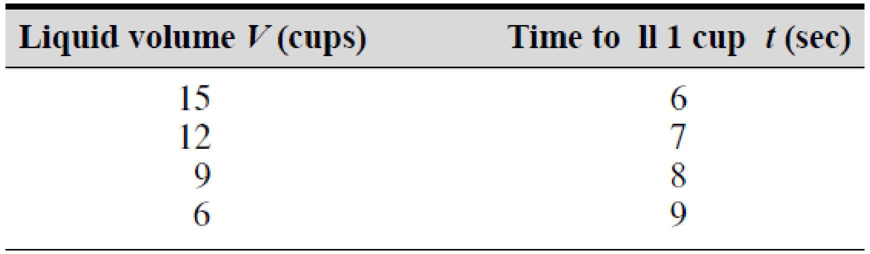


Liquid volume V (cups)	Time to ll 1 cup t (sec)
15	6
12	7
9	8
6	9

An experiment to verify Torricelli's principle

Torricelli's principle in hydraulics states that $f = rV^{1/2}$, where f is the flow rate through the outlet valve in cups per second, V is the value of liquid in the pot in cups, and r is a constant whose value is to be found. We see that this relation is a power function where the exponent is 0.5. Thus if we plot $log_{10}(f)$ versus $log_{10}(V)$, we should obtain a straight line. The values for f are obtained from the reciprocals of the given data for t. That is $f = 1/t$ cups per second.

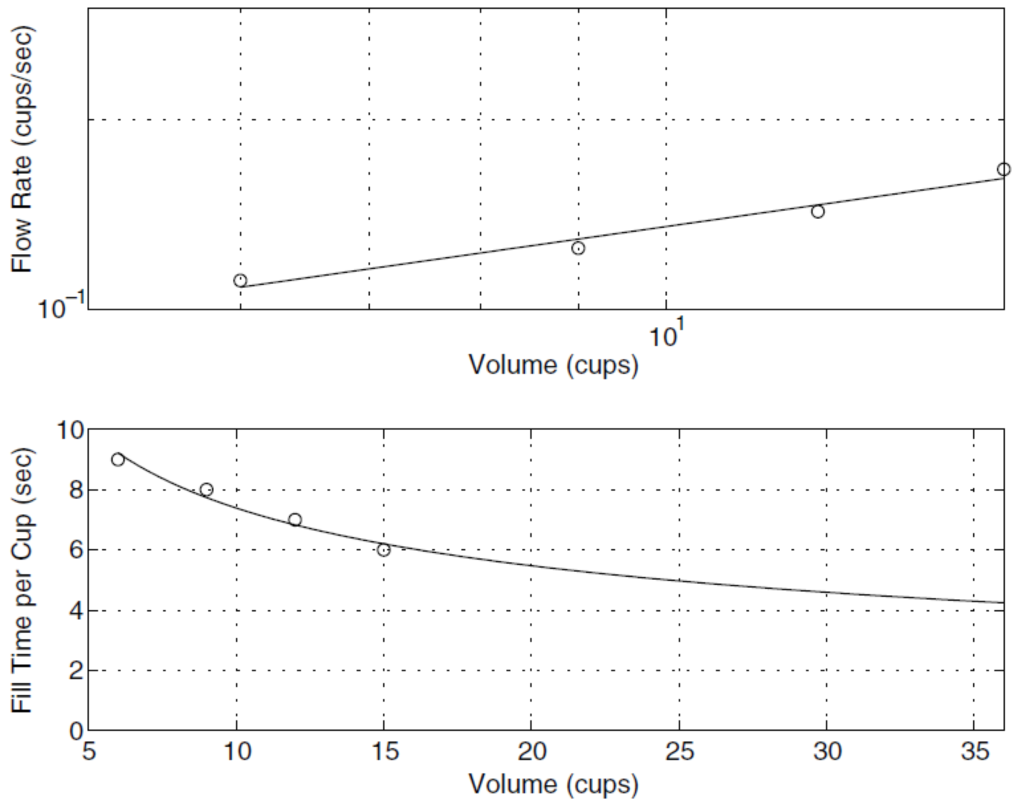
(a) Use the preceding data to obtain a relation between the flow rate and the number of cups in the pot. (b) The manufacturer wants to make a 36-cup pot using the same outlet valve but is concerned that a cup will fill too quickly, causing spills. Extrapolate the relation developed in part (a) and predict how long it will take to fill one cup when the pot containing 36 cups.



Liquid volume V (cups)	Time to ll 1 cup t (sec)
15	6
12	7
9	8
6	9

Solution

(a) Torricelli's principle in hydraulics states that $f = rV^{1/2}$, where f is the flow rate through the outlet valve in cups per second, V is the volume of liquid in the pot in cups, and r is the constant whose value is to be found. We see that this relation is a power function where the exponent is 0.5. Thus if we plot $log_{10}(f)$ versus $log_{10}(V)$, we should obtain a straight line.



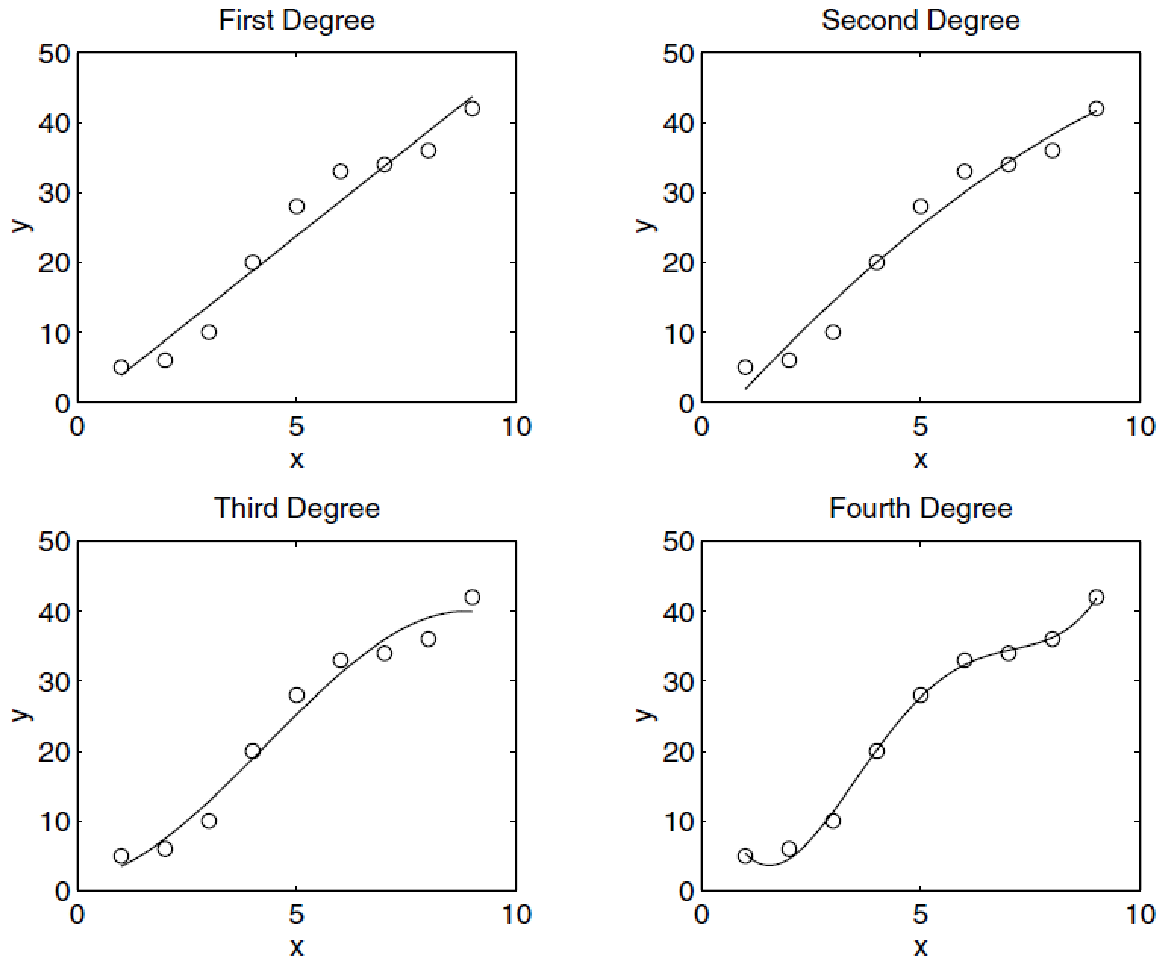
```
clc
clear
% Data for the problem
% coffee temperature evolution
Volume=[15, 12, 9, 6]; % Liquid volume
t=[6 7 8 9]; % Time to fill 1 cup
cup_persecond=1./t; % the cup amount filled within one second

%
% Fit a straight line to the transformed data;
p=polyfit(log10(Volume),log10(cup_persecond),1); % polynomial fitting of data
m=p(1)
b=10.^(p(2))
%
% Plot the data and the fitted line on a loglog plot to see how weel
% the line fits the data
x=6:0.01:40;
y=b*x.^m;
```

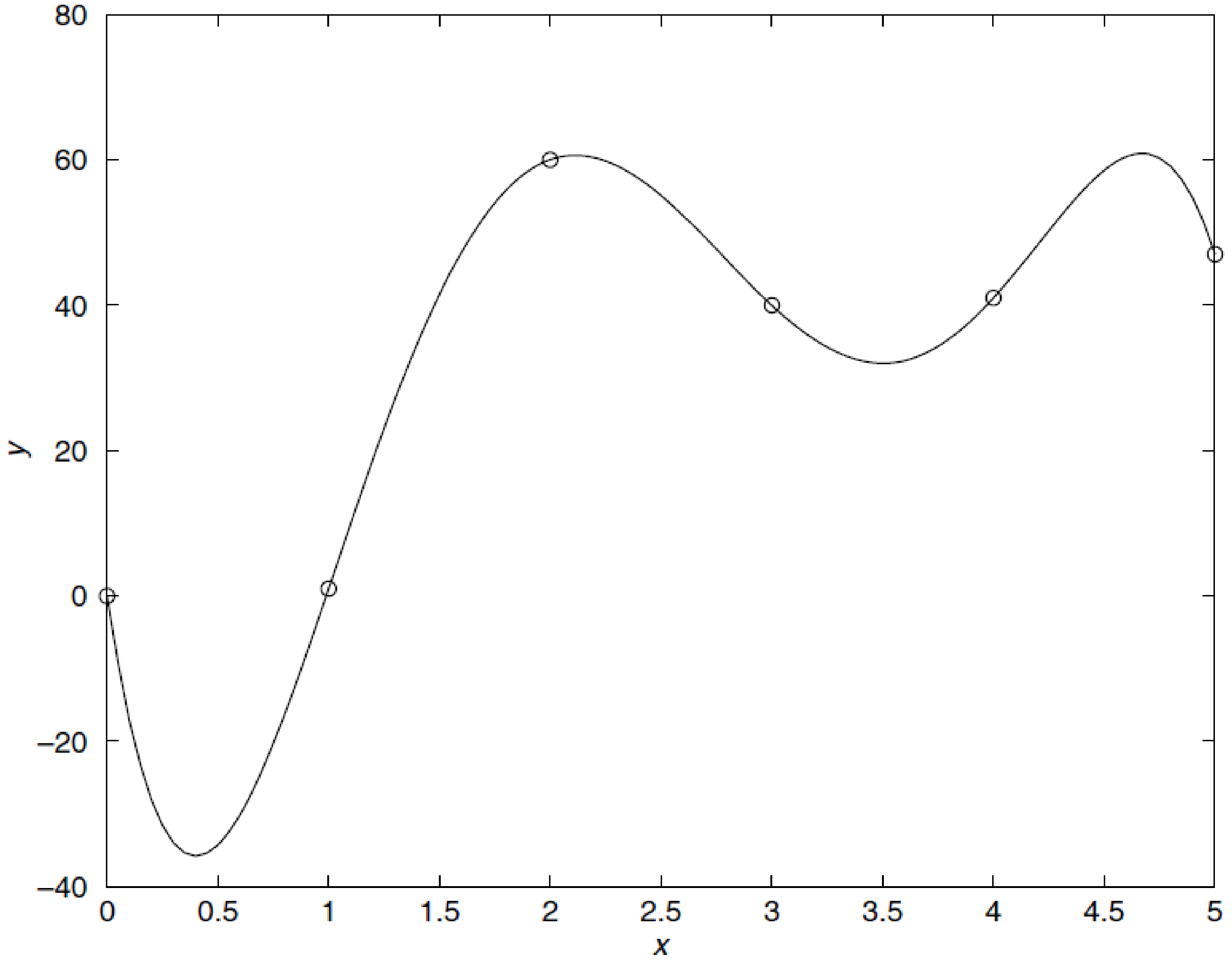
The computed values are $m = 0.433$ and $b = 0.0499$, and our derived relation is $f = 0.0499V^{0.433}$

```
%
% Display the equation
fprintf('The derived equation is f=%.4fV^%.4f',b,m)
subplot(2,1,1)
loglog(x,y, Volume,cup_persecond,'o')
grid
xlabel ('Volume (cups)')
ylabel ('Flow Rate (cups/sec)')
x1=linspace(2,40,38);
y1=b.*x1.^(m);
subplot(2,1,2)
plot(x1,1./y1,Volume,t,'*')
grid
xlabel('Volume (cups)')
ylabel('Fill Time per Cup (sec)')
axis([5 36 0 10])
%
% Compute the fill time for V=35 cups.
fill_time=1/(b*36^m)
```

The predicted fill time for 1 cup is 4.2 sec. In fact, the manufacturer did construct a 36-cup pot, and the fill time is approximately 4 sec, which agrees with our prediction.

Regression using polynomials of first through fourth degree





An example of a fifth-degree polynomial that passes through all six data points but exhibits large excursion between points

Caution: It is tempting to use a high-degree polynomial to obtain the best possible fit. However, there are two dangers in using high-degree polynomials.

- High-degree polynomials often exhibit large excursions between the data points and thus should be avoided if possible.
- High-degree polynomials can produce large errors if their coefficients are not represented with a large number of significant figures.
- In some cases it might not be possible to fit the data with a low-degree polynomial. In such cases we might be able to use several cubic polynomials, called cubic splines

The Quality of a Curve Fit

The least-squares criterion used to fit a function $f(x)$ is the sum of the squares of the residuals J.

$$J = \sum_{i=1}^{m} [f(x_i) - y_i]^2$$

We can use the J value to compare the quality of the curve fit for two or more functions used to describe the same data. The function that gives the smallest J value gives the best fit.

We denote the sum of the squares of the deviation of the y values from their mean $\bar{y}$ by S , which can be computed from

$$S = \sum_{i=1}^{m} (y_i - \bar{y})^2$$

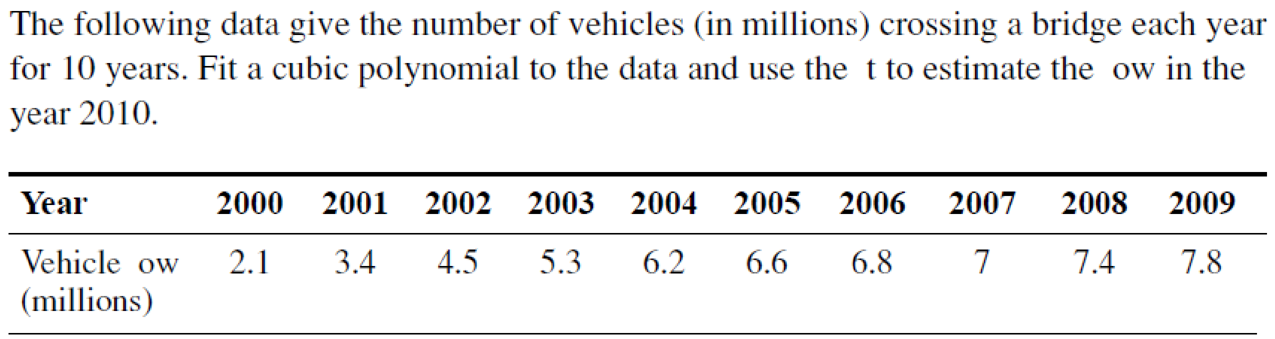
This formula can be used to compute another measure of the quality of the curve fit, the $coefficeint\ of\ determination$, also known as the $r - squared$ value. It is denoted as

$$r^2 = 1 - \frac{J}{S}$$

For a perfect fit, $J = 0$ and thus $r^2 = 1$. Thus the closer r^2 is to 1, the better the fit. The value of S indicates how much the data is spread around the mean, and the value of J indicates how much of the data spread is unaccounted for by the model. It is possible for J to be larger than S, and thus it is possible for r^2 to be negative. Such cases, however, are indicative of a very poor model that should not be used. As a rule of thumb, a good fit accounts for at least 99 percent of the data variation. This value corresponds to $r^2 \geq 0.99$

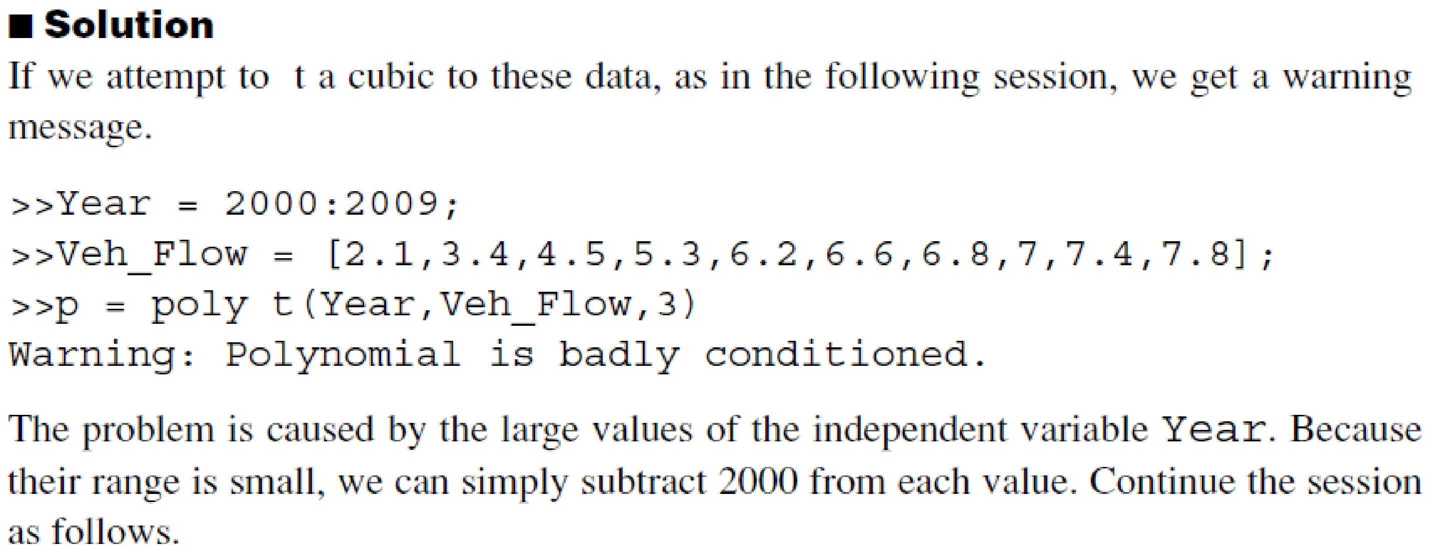
Practice: Estimation of Traffic Flow

The following data give the number of vehicles (in millions) crossing a bridge each year for 10 years. Fit a cubic polynomial to the data and use the fit to estimate the flow in the year 2010.



The following data give the number of vehicles (in millions) crossing a bridge each year for 10 years. Fit a cubic polynomial to the data and use the t to estimate the ow in the year 2010.

Year	**2000**	**2001**	**2002**	**2003**	**2004**	**2005**	**2006**	**2007**	**2008**	**2009**
Vehicle ow (millions)	2.1	3.4	4.5	5.3	6.2	6.6	6.8	7	7.4	7.8



■ Solution

If we attempt to t a cubic to these data, as in the following session, we get a warning message.

```
>>Year = 2000:2009;
>>Veh_Flow = [2.1,3.4,4.5,5.3,6.2,6.6,6.8,7,7.4,7.8];
>>p = poly t(Year,Veh_Flow,3)
Warning: Polynomial is badly conditioned.
```

The problem is caused by the large values of the independent variable `Year`. Because their range is small, we can simply subtract 2000 from each value. Continue the session as follows.

```
>>x = Year-2000; y = Veh_Flow;
>>p = poly t(x,y,3)
p =
   0.0087     -0.1851   1.5991  2.0362
>>J = sum((polyval(p,x)-y).^2);
>>S = sum((y-mean(y)).^2);
>>r2 = 1 - J/S
r2 =
   0.9972
```

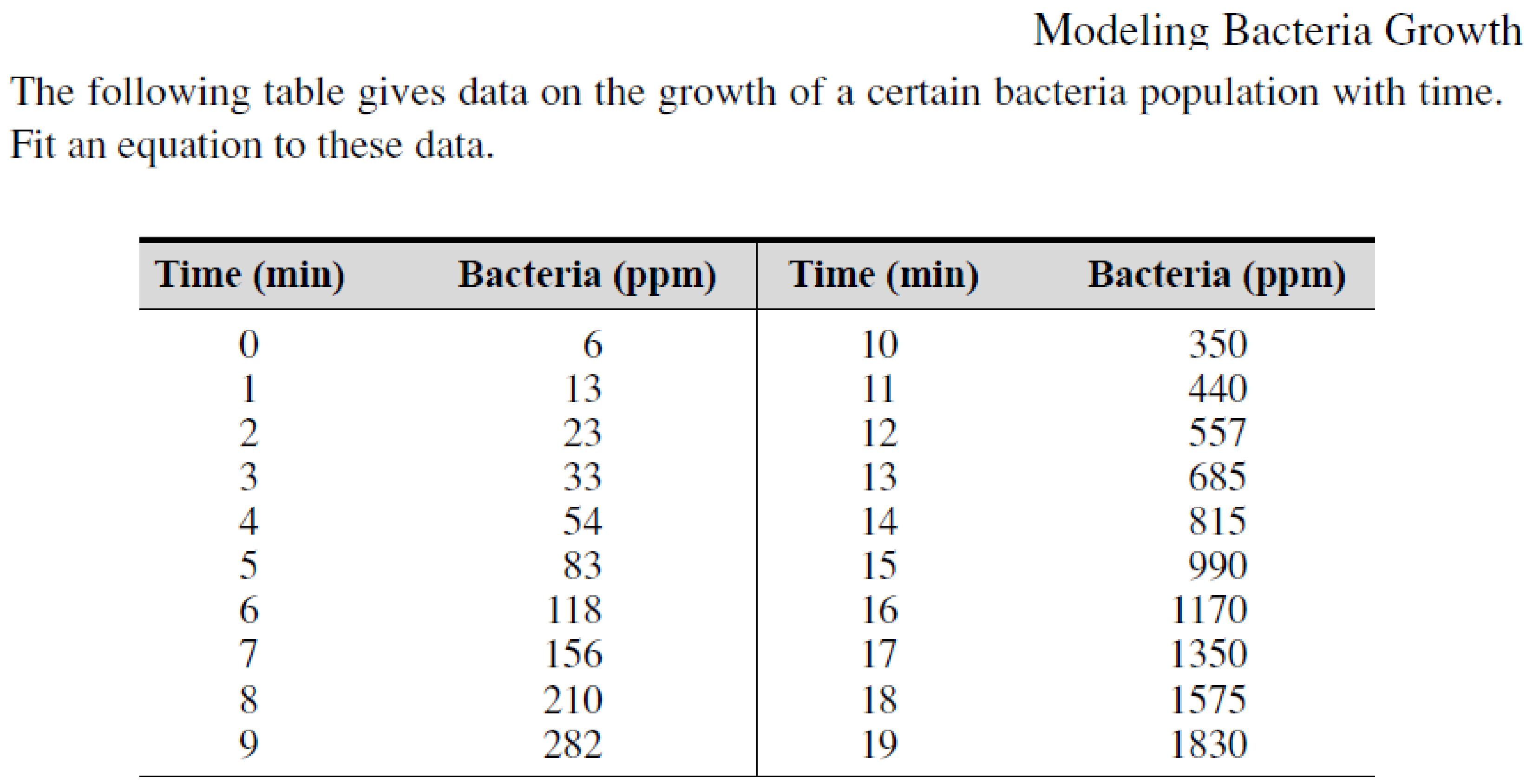
Thus the polynomial t is good because the coef cient of determination is 0.9972. The corresponding polynomial is

$$f = 0.0087(t - 2000)^3 - 0.1851(t - 2000)^2 + 1.5991(t - 2000) + 2.0362$$

where f is the traf c ow in millions of vehicles and t is the time in years measured from 0. We can use this equation to estimate the ow at the year 2010 by substituting $t = 2010$, or by typing in MATLAB `polyval(p,10)`. Rounded to one decimal place, the answer is 8.2 million vehicles.

Using Residuals

We now show how to use the residuals as a guide to choosing an appropriate function to describe the data. In general, if you see a pattern in the plot of the residuals, it indicates that another function can be found to describe the data better.



Modeling Bacteria Growth

The following table gives data on the growth of a certain bacteria population with time. Fit an equation to these data.

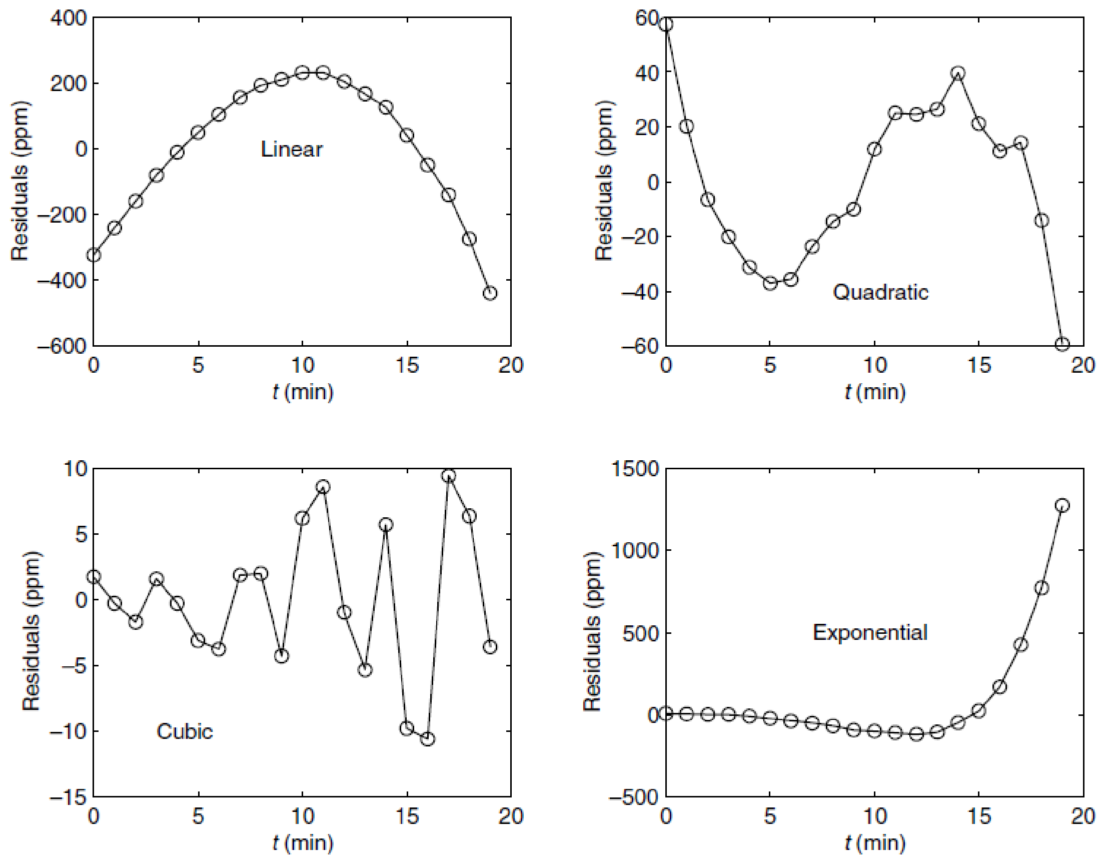
Time (min)	Bacteria (ppm)	Time (min)	Bacteria (ppm)
0	6	10	350
1	13	11	440
2	23	12	557
3	33	13	685
4	54	14	815
5	83	15	990
6	118	16	1170
7	156	17	1350
8	210	18	1575
9	282	19	1830

We try three polynomials fits(linear, quadratic, and cubic) and an exponential fit. Note that we can write the exponential form as $y = b(10)^{mt} = 10^{mt+a}$, where $b = 10^{a}$

```
% Time data
x = 0:19;
% Population data
y = [6,13,23,33,54,83,118,156,210,282,...
   350,440,557,685,815,990,1170,1350,1575,1830];
% Linear  t
p1 = poly t(x,y,1);
% Quadratic  +
p2 = poly t(x
% Cubic  t
p3 = poly t(x
% Exponential
p4 = poly t(x
% Residuals
res1 = polyval(p1,x)-y;
res2 = polyval(p2,x)-y;
res3 = polyval(p3,x)-y;
res4 = 10.^polyval(p4,x)-y;
```

You can then plot the residuals as shown in Figure 6.2–3. Note that there is a de nite pattern in the residuals of the linear t. This indicates that the linear function cannot match the curvature of the data. The residuals of the quadratic t are much smaller , but there is still a pattern, with a random component. This indicates that the quadratic

You can then plot the residuals as shown in Figure 6.2–3. Note that there is a de nite pattern in the residuals of the linear t. This indicates that the linear function cannot match the curvature of the data. The residuals of the quadratic t are much smaller , but there is still a pattern, with a random component. This indicates that the quadratic



Residual plots for the four models

- There is a definite pattern in the residuals of linear fit. This indicates that the linear function cannot match the curvature of the data.
- The residuals of the quadratic fit are much smaller, but there is still a pattern, with a random component. This indicates that quadratic function also cannot match the curvature of the data.
- The residuals of the cubic fit are even smaller, with no strong pattern and a large random component. This indicates that a polynomial degree higher than 3 will not be able to match the data curvature any better than the cubic.
- The residuals for the exponential are the largest of all, and indicate a poor fit.

The cubic is the best fit of the four models considered. Its coefficient of determination is $r^2 = 0.9999$. The model is

$$y = 0.1916t^3 + 1.2082t^2 + 3.607t + 7.7307$$

where y is the bacteria population in ppm and t is time in minutes