MATLAB and Engineering Application

Review

An overview of MATLAB

Table 1.1–1 Scalar arithmetic operations

Symbol	Operation	MATLAB form
^	exponentiation: a^b	`a^b`
*	multiplication: ab	`a*b`
/	right division: $a/b = \frac{a}{b}$	`a/b`
\	left division: $a\backslash b = \frac{b}{a}$	`a\b`
+	addition: $a + b$	`a+b`
−	subtraction: $a - b$	`a-b`

Table 1.1–2 Order of precedence

Precedence	Operation
First	Parentheses, evaluated starting with the innermost pair.
Second	Exponentiation, evaluated from left to right.
Third	Multiplication and division with equal precedence, evaluated from left to right.
Fourth	Addition and subtraction with equal precedence, evaluated from left to right.

Table 1.1–3 Commands for managing the work session

Command	Description
`clc`	Clears the Command window.
`clear`	Removes all variables from memory.
`clear var1 var2`	Removes the variables `var1` and `var2` from memory.
`exist('name')`	Determines if a le or variable exists having the name 'name'.
`quit`	Stops MATLAB.
`who`	Lists the variables currently in memory.
`whos`	Lists the current variables and sizes and indicate if they have imaginary parts.
`:`	Colon; generates an array having regularly spaced elements.
`,`	Comma; separates elements of an array.
`;`	Semicolon; suppresses screen printing; also denotes a new row in an array.
`...`	Ellipsis; continues a line.

Table 1.1–4 Special variables and constants

Command	Description
`ans`	Temporary variable containing the most recent answer.
`eps`	Speci es the accuracy of oating point precision.
`i,j`	The imaginary unit $\sqrt{-1}$.
`Inf`	In nity .
`NaN`	Indicates an unde ned numerical result.
`pi`	The number π.

Table 1.1–5 Numeric display formats

Command	Description and example
`format short`	Four decimal digits (the default); 13.6745.
`format long`	16 digits; 17.27484029463547.
`format short e`	Five digits (four decimals) plus exponent; 6.3792e+03.
`format long e`	16 digits (15 decimals) plus exponent; 6.379243784781294e−04.
`format bank`	Two decimal digits; 126.73.
`format +`	Positive, negative, or zero; +.
`format rat`	Rational approximation; 43/7.
`format compact`	Suppresses some blank lines.
`format loose`	Resets to less compact display mode.

Use MATLAB to compute the following expressions.

a. $6\left(\frac{10}{13}\right) + \frac{18}{5(7)} + 5(9^2)$

b. $6(35^{1/4}) + 14^{0.35}$

(Answers: a. 410.1297 b. 17.1123.)

Volume of a Circular Cylinder

The volume of a circular cylinder of height h and radius r is given by $V = \pi r^2 h$. A particular cylindrical tank is 15 m tall and has a radius of 8 m. We want to construct another cylindrical tank with a volume 20 percent greater but having the same height. How large must its radius be?

Arrays, Files, and Plots

T1.3–1 Use MATLAB to determine how many elements are in the array `cos(0):0.02:log10(100)`. Use MATLAB to determine the 25th element. (Answer: 51 elements and 1.48.)

T1.3–2 Use MATLAB to nd the roots of the polynomial $290 - 11x + 6x^2 + x^3$. (Answer: $x = -10, 2 \pm 5i$.)

Plotting with MATLAB

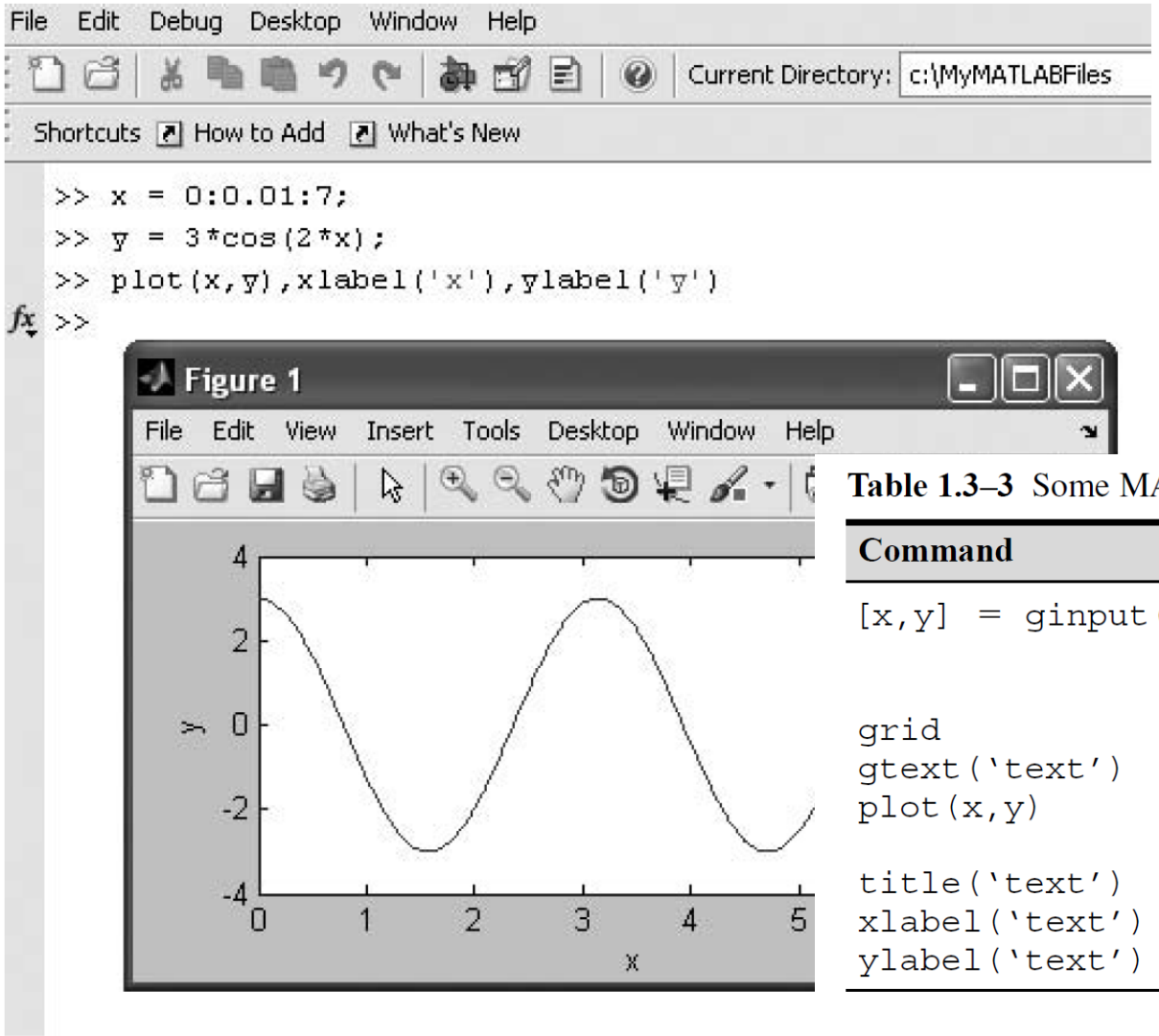


Table 1.3–3 Some MATLAB plotting commands

Command	Description
`[x,y] = ginput(n)`	Enables the mouse to get *n* points from a plot, and returns the *x* and *y* coordinates in the vectors `x` and `y`, which have a length *n*.
`grid`	Puts grid lines on the plot.
`gtext('text')`	Enables placement of text with the mouse.
`plot(x,y)`	Generates a plot of the array `y` versus the array `x` on rectilinear axes.
`title('text')`	Puts text in a title at the top of the plot.
`xlabel('text')`	Adds a text label to the horizontal axis (the abscissa).
`ylabel('text')`	Adds a text label to the vertical axis (the ordinate).

Test Your Understanding

T1.3–3 Use MATLAB to plot the function $s = 2\sin(3t + 2) + \sqrt{5t + 1}$ over the interval $0 \leq t \leq 5$. Put a title on the plot, and properly label the axes. The variable s represents speed in feet per second; the variable t represents time in seconds.

T1.3–4 Use MATLAB to plot the functions $y = 4\sqrt{6x + 1}$ and $z = 5e^{0.3x} - 2x$ over the interval $0 \leq x \leq 1.5$. Properly label the plot and each curve. The variables y and z represent force in newtons; the variable x represents distance in meters.

Linear Algebraic Equations

```
>>A = [6,12,4;7,-2,3;2,8,-9];
>>B = [70;5;64];
>>Solution = A\B
Solution =
     3
     5
    -2
```

Script Files and the Editor/Debugger

Table 1.4–1 Input/output commands

Command	Description
`disp(A)`	Displays the contents, but not the name, of the array `A`.
`disp('text')`	Displays the text string enclosed within single quotes.
`format`	Controls the screen's output display format (see Table 1.1–5).
`x = input('text')`	Displays the text in quotes, waits for user input from the keyboard, and stores the value in `x`.
`x = input('text','s')`	Displays the text in quotes, waits for user input from the keyboard, and stores the input as a string in `x`.
`k=menu('title','option1', 'option2',...`	Displays a menu whose title is in the string variable `'title'` and whose choices are `'option1'`,`'option2'`, and so on.

Problem-Solving Methodologies

Table 1.6–1 Steps in engineering problem solving

1. Understand the purpose of the problem.
2. Collect the known information. Realize that some of it might later be found unnecessary.
3. Determine what information you must nd.
4. Simplify the problem only enough to obtain the required information. State any assumptions you make.
5. Draw a sketch and label any necessary variables.
6. Determine which fundamental principles are applicable.
7. Think generally about your proposed solution approach and consider other approaches before proceeding with the details.
8. Label each step in the solution process.
9. If you solve the problem with a program, hand check the results using a simple version of the problem. Checking the dimensions and units and printing the results of intermediate steps in the calculation sequence can uncover mistakes.
10. Perform a "reality check" on your answer. Does it make sense? Estimate the range of the expected result and compare it with your answer. Do not state the answer with greater precision than is justi ed by any of the following:
    (a) The precision of the given information.
    (b) The simplifying assumptions.
    (c) The requirements of the problem.

    Interpret the mathematics. If the mathematics produces multiple answers, do not discard some of them without considering what they mean. The mathematics might be trying to tell you something, and you might miss an opportunity to discover more about the problem.

Piston Motion

Figure 1.6–2a shows a piston, connecting rod, and crank for an internal combustion engine. When combustion occurs, it pushes the piston down. This motion causes the connecting rod to turn the crank, which causes the crankshaft to rotate. We want to develop a MATLAB program to compute and plot the distance d traveled by the piston as a function of the angle A, for given values of lengths L_1 and L_2. Such a plot would help the engineers designing the engine to select appropriate values for lengths L_1 and L_2.

We are told that typical values for these lengths are $L_1 = 1$ ft and $L_2 = 0.5$ ft. Because the mechanism's motion is symmetrical about $A = 0$, we need consider only angles in the range $0 \le A \le 180°$. Figure 1.6–2b shows the geometry of the motion. From this gure we can use trigonometry to write the following expression for d:

$$d = L_1 \cos B + L_2 \cos A \qquad (1.6\text{–}2)$$

Thus to compute d given the lengths L_1 and L_2 and the angle A, we must rst determine the angle B. We can do so using the law of sines, as follows:

$$\frac{\sin A}{L_1} = \frac{\sin B}{L_2}$$

Solve this for B:

$$\sin B = \frac{L_2 \sin A}{L_1}$$

$$B = \sin^{-1}\left(\frac{L_2 \sin A}{L_1}\right) \qquad (1.6\text{–}3)$$

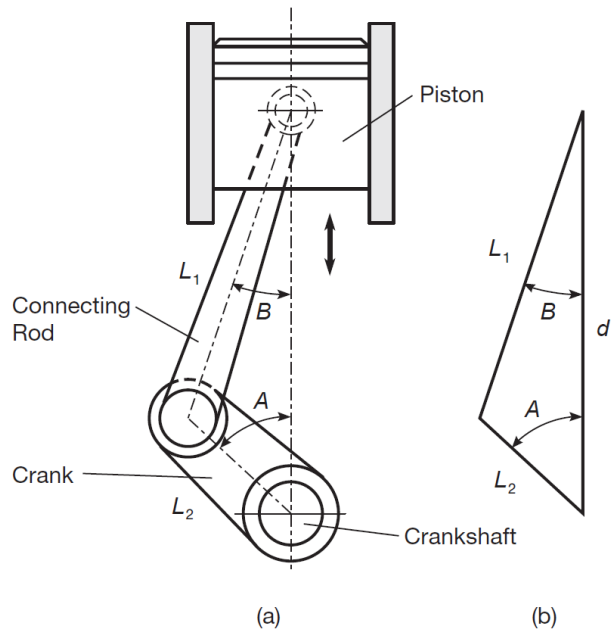


Figure 1.6–2 A piston, connecting rod, and crank for an internal combustion engine.

■ Solution

Here are the steps in the solution, following those listed in Table 1.6–2.

1. *State the problem concisely.* Use Equations (1.6–2) and (1.6–3) to compute d; use enough values of A in the range $0 \le A \le 180°$ to generate an adequate (smooth) plot.
2. *Specify the input data to be used by the program.* The lengths L_1 and L_2 and the angle A are given.
3. *Specify the output to be generated by the program.* A plot of d versus A is the required output.
4. *Work through the solution steps by hand or with a calculator.* You could have made an error in deriving the trigonometric formulas, so you should check them for several cases. You can check for these errors by using a ruler and protractor to make a scale drawing of the triangle for several values of the angle A; measure the length d; and compare it to the calculated values. Then you can use these results to check the output of the program.

   Which values of A should you use for the checks? Because the triangle "collapses" when $A = 0°$ and $A = 180°$, you should check these cases. The results are $d = L_1 - L_2$ for $A = 0°$ and $d = L_1 + L_2$ for $A = 180°$. The case $A = 90°$ is also easily checked by hand, using the Pythagorean theorem; for this case $d = \sqrt{L_1^2 - L_2^2}$. You should also check one angle in the quadrant $0° < A < 90°$ and one in the quadrant $90° < A < 180°$. The following table shows the results of these calculations using the given typical values: $L_1 = 1$, $L_2 = 0.5$ ft.

A (degrees)	d (ft)
0	1.5
60	1.15
90	0.87
120	0.65
180	0.5

5. *Write and run the program.* The following MATLAB session uses the values $L_1 = 1$, $L_2 = 0.5$ ft.

```
>>L_1 = 1;
>>L_2 = 0.5;
>>R = L_2/L_1;
>>A_d = 0:0.5:180;
>>A_r = A_d*(pi/180);
>>B = asin(R*sin(A_r));
>>d = L_1*cos(B)+L_2*cos(A_r);
>>plot(A_d,d),xlabel('A (degrees)'), ...
ylabel('d (feet)'),grid
```

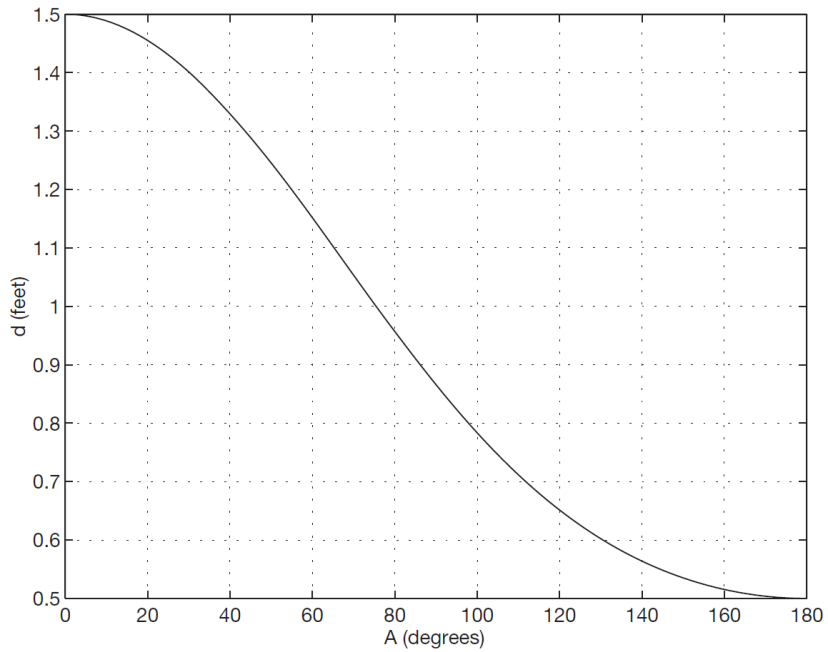


Figure 1.6–3 Plot of the piston motion versus crank angle.

The Fourier series is a series representation of a periodic function in terms of sines and cosines. The Fourier series representation of the function

$$f(x) = \begin{cases} 1 & 0 < x < \pi \\ -1 & -\pi < x < 0 \end{cases}$$

is

$$\frac{4}{\pi}\left(\frac{\sin x}{1} + \frac{\sin 3x}{3} + \frac{\sin 5x}{5} + \frac{\sin 7x}{7} + \cdots\right)$$

Plot on the same graph the function $f(x)$ and its series representation, using the four terms shown.

The four-sided figure shown in Figure P26 consists of two triangles having a common side a. The law of cosines for the top triangle states that

$$a^2 = b_1^2 + c_1^2 - 2b_1c_1 \cos A_1$$

and a similar equation can be written for the bottom triangle. Develop a procedure for computing the length of side c_2 if you are given the lengths of sides b_1, b_2, and c_1 and the angles A_1 and A_2 in degrees. Write a script file to implement this procedure. Test your script, using the following values: $b_1 = 180$ m, $b_2 = 165$ m, $c_1 = 115$ m, $A_1 = 120°$, and $A_2 = 100°$.

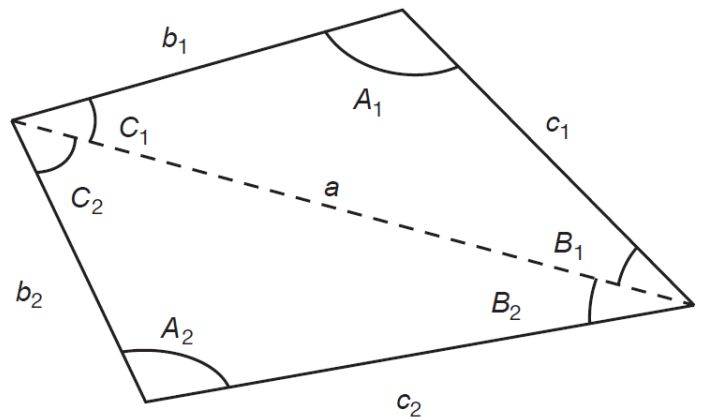


Figure P26

Numeric, Cell, and Structure Arrays

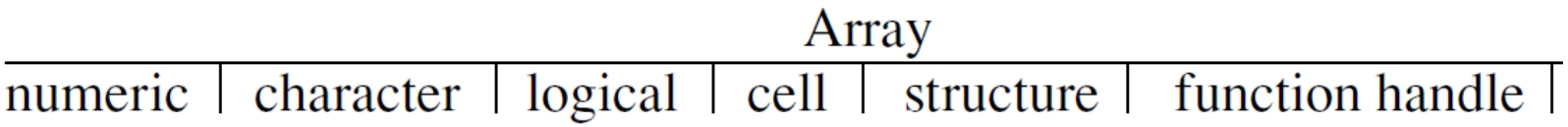


Table 2.1–1 Basic syntax of array functions*

Command	Description
`nd(x)`	Computes an array containing the indices of the nonzero elements of the array **x**.
`[u,v,w] = nd(A)`	Computes the arrays **u** and **v**, containing the row and column indices of the nonzero elements of the matrix **A**, and the array **w**, containing the values of the nonzero elements. The array **w** may be omitted.
`length(A)`	Computes either the number of elements of **A** if **A** is a vector or the largest value of m or n if **A** is an $m \times n$ matrix.
`linspace(a,b,n)`	Creates a row vector of n regularly spaced values between a and b.
`logspace(a,b,n)`	Creates a row vector of n logarithmically spaced values between a and b.
`max(A)`	Returns the algebraically largest element in **A** if **A** is a vector. Returns a row vector containing the largest elements in each column if **A** is a matrix. If any of the elements are complex, `max(A)` returns the elements that have the largest magnitudes.
`[x,k] = max(A)`	Similar to `max(A)` but stores the maximum values in the row vector **x** and their indices in the row vector **k**.
`min(A)`	Same as `max(A)` but returns minimum values.
`[x,k] = min(A)`	Same as `[x,k] = max(A)` but returns minimum values.
`norm(x)`	Computes a vector's geometric length $\sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$.
`size(A)`	Returns a row vector `[m n]` containing the sizes of the $m \times n$ array **A**.
`sort(A)`	Sorts each column of the array **A** in ascending order and returns an array the same size as **A**.
`sum(A)`	Sums the elements in each column of the array **A** and returns a row vector containing the sums.

Multidimensional Numeric Arrays

A three-dimensional array has the dimension $m \times n \times q$. A four-dimensional array has the dimension $m \times n \times q \times r$, and so forth. The first two dimensions are the row and column, as with a matrix. The higher dimensions are called *pages*. You can think of a three-dimensional array as layers of matrices. The first layer is page 1; the second layer is page 2, and so on. If `A` is a $3 \times 3 \times 2$ array, you can access the element in row 3, column 2 of page 2 by typing `A(3,2,2)`. To access all of page 1, type `A(:,:,1)`. To access all of page 2, type `A(:,:,2)`. The `ndims` command returns the number of dimensions. For example, for the array `A` just described, `ndims(A)` returns the value 3.

```
>>A = [4,6,1;5,8,0;3,9,2];
>>A(:,:,2) = [6,2,9;0,3,1;4,7,5];
```

Table 2.3–1 Element-by-element operations

Symbol	Operation	Form	Example
+	Scalar-array addition	`A + b`	`[6,3]+2=[8,5]`
−	Scalar-array subtraction	`A − b`	`[8,3]−5=[3,−2]`
+	Array addition	`A + B`	`[6,5]+[4,8]=[10,13]`
−	Array subtraction	`A − B`	`[6,5]−[4,8]=[2,−3]`
.*	Array multiplication	`A.*B`	`[3,5].*[4,8]=[12,40]`
./	Array right division	`A./B`	`[2,5]./[4,8]=[2/4,5/8]`
.\	Array left division	`A.\B`	`[2,5].\[4,8]=[2\4,5\8]`

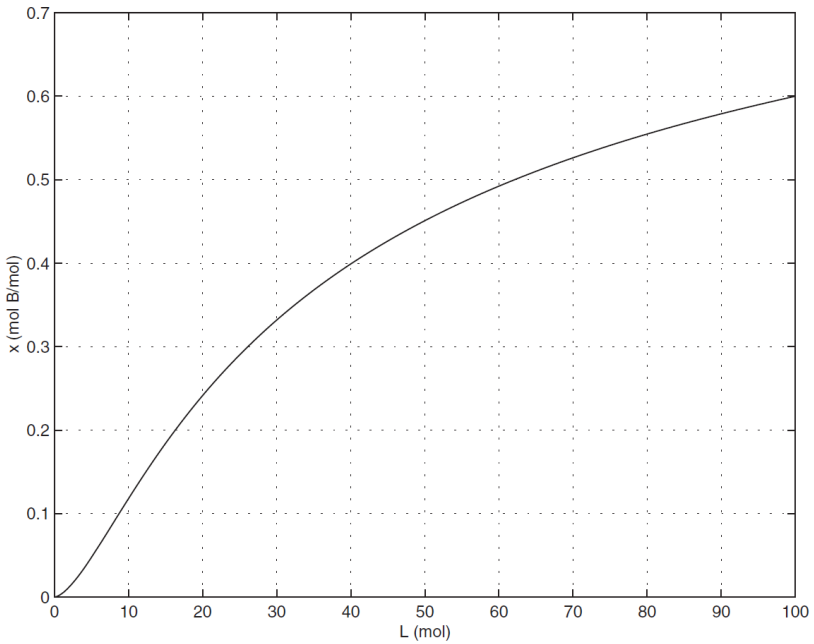
.^	Array exponentiation	`A.^B`	`[3,5].^2=[3^2,5^2]`
			`2.^[3,5]=[2^3,2^5]`
			`[3,5].^[2,4]=[3^2,5^4]`

Consider a system for heating a liquid benzene/toluene solution to distill a pure benzene vapor. A particular batch distillation unit is charged initially with 100 mol of a 60 percent mol benzene/40 percent mol toluene mixture. Let L (mol) be the amount of liquid remaining in the still, and let x (mol B/mol) be the benzene mole fraction in the remaining liquid. Conservation of mass for benzene and toluene can be applied to derive the following relation [Felder, 1986].

$$L = 100\left(\frac{x}{0.6}\right)^{0.625}\left(\frac{1-x}{0.4}\right)^{-1.625}$$

Determine what mole fraction of benzene remains when $L = 70$. Note that it is dif cult to solve this equation directly for x. Use a plot of x versus L to solve the problem.

```
>>x = 0:0.001:0.6;
>>L = 100*(x/0.6).^(0.625).*((1-x)/0.4).^(-1.625);
>>plot(L,x),grid,xlabel('L(mol)'),ylabel('x (mol B/mol)'), ...
  [L,x] = ginput(1)
```



Height versus Velocity

The maximum height h achieved by an object thrown with a speed v at an angle θ to the horizontal, neglecting drag, is

$$h = \frac{v^2 \sin^2\theta}{2g}$$

Create a table showing the maximum height for the following values of v and θ:

$$v = 10, 12, 14, 16, 18, 20 \text{ m/s} \qquad \theta = 50°, 60°, 70°, 80°$$

The rows in the table should correspond to the speed values, and the columns should correspond to the angles.

■ Solution

The program is shown below.

```
g = 9.8; v = 10:2:20;
theta = 50:10:80;
h = (v'.^2)*(sind(theta).^2)/(2*g);
table = [0, theta; v', h]
```

The arrays `v` and `theta` contain the given velocities and angles. The array `v` is 1×6 and the array `theta` is 1×4. Thus the term `v'.^2` is a 6×1 array, and the term `sind(theta).^2` is a 1×4 array. The product of these two arrays, `h`, is a matrix product and is a $(6 \times 1)(1 \times 4) = (6 \times 4)$ matrix.

The array `[0, theta]` is 1×5 and the array `[v', h]` is 6×5, so the matrix `table` is 7×5. The following table shows the matrix `table` rounded to one decimal place. From this table we can see that the maximum height is 8.8 m if $v = 14$ m/s and $\theta = 70°$.

0	50	60	70	80
10	3.0	3.8	4.5	4.9
12	4.3	5.5	6.5	7.1
14	5.9	7.5	8.8	9.7
16	7.7	9.8	11.5	12.7
18	9.7	12.4	14.6	16.0
20	12.0	15.3	18.0	19.8

Table 2.4–2 shows the hourly cost of four types of manufacturing processes. It also shows the number of hours required of each process to produce three different products. Use matrices and MATLAB to solve the following. (a) Determine the cost of each process to produce 1 unit of product 1. (b) Determine the cost to make 1 unit of each product. (c) Suppose we produce 10 units of product 1, 5 units of product 2, and 7 units of product 3. Compute the total cost.

Table 2.4–2 Cost and time data for manufacturing processes

		Hours required to produce one unit		
Process	**Hourly cost ($)**	**Product 1**	**Product 2**	**Product 3**
Lathe	10	6	5	4
Grinding	12	2	3	1
Milling	14	3	2	5
Welding	9	4	0	3

Table 2.4–5 Special matrices

Command	Description
`eye(n)`	Creates an $n \times n$ identity matrix.
`eye(size(A))`	Creates an identity matrix the same size as the matrix **A**.
`ones(n)`	Creates an $n \times n$ matrix of 1s.
`ones(m,n)`	Creates an $m \times n$ array of 1s.
`ones(size(A))`	Creates an array of 1s the same size as the array **A**.
`zeros(n)`	Creates an $n \times n$ matrix of 0s.
`zeros(m,n)`	Creates an $m \times n$ array of 0s.
`zeros(size(A))`	Creates an array of 0s the same size as the array **A**.

Table 2.5–1 Polynomial functions

Command	Description
`conv(a,b)`	Computes the product of the two polynomials described by the coef cient arrays `a` and `b`. The two polynomials need not be of the same degree. The result is the coef cient array of the product polynomial.
`[q,r] = deconv(num,den)`	Computes the result of dividing a numerator polynomial, whose coef cient array is `num`, by a denominator polynomial represented by the coef cient array `den`. The quotient polynomial is given by the coef cient array `q`, and the remainder polynomial is given by the coef cient array `r`.
`poly(r)`	Computes the coef cients of the polynomial whose roots are speci ed by the vector `r`. The result is a *row* vector that contains the polynomial's coef cients arranged in descending order of power.
`polyval(a,x)`	Evaluates a polynomial at speci ed values of its independent variable `x`, which can be a matrix or a vector. The polynomial's coef cients of descending powers are stored in the array `a`. The result is the same size as `x`.
`roots(a)`	Computes the roots of a polynomial speci ed by the coef cient array `a`. The result is a *column* vector that contains the polynomial's roots.

The product of the polynomials $f(x)$ and $g(x)$ is

$$\begin{aligned} f(x)g(x) &= (9x^3 - 5x^2 + 3x + 7)(6x^2 - x + 2) \\ &= 54x^5 - 39x^4 + 41x^3 + 29x^2 - x + 14 \end{aligned}$$

Dividing $f(x)$ by $g(x)$ using synthetic division gives a quotient of

$$\frac{f(x)}{g(x)} = \frac{9x^3 - 5x^2 + 3x + 7}{6x^2 - x + 2} = 1.5x - 0.5833$$

with a remainder of $-0.5833x + 8.1667$. Here is the MATLAB session form these operations.

```
>>f = [9,-5,3,7];
>>g = [6,-1,2];
>>product = conv(f,g)
product =
   54    -39    41    29    -1    14
>>[quotient, remainder] = deconv(f,g)
quotient =
   1.5    -0.5833
remainder =
   0    0    -0.5833    8.1667
```

Cell Arrays

The *cell array* is an array in which each element is a *bin,* or *cell,* which can contain an array. You can store different classes of arrays in a cell array, and you can group data sets that are related but have different dimensions. You access cell arrays using the same indexing operations used with ordinary arrays.

Suppose you want to create a 2×2 cell array `A`, whose cells contain the location, the date, the air temperature (measured at 8 A.M., 12 noon, and 5 P.M.), and the water temperatures measured at the same time in three different points in a pond. The cell array looks like the following.

Walden Pond	June 13, 1997
[60 72 65]	$\begin{bmatrix} 55 & 57 & 56 \\ 54 & 56 & 55 \\ 52 & 55 & 53 \end{bmatrix}$

■ Solution

You can create this array by typing the following either in interactive mode or in a script file and running it.

```
A(1,1) = {'Walden Pond'};
A(1,2) = {'June 13, 1997'};
A(2,1) = {[60,72,65]};
A(2,2) = {[55,57,56;54,56,55;52,55,53]};
```

```
A{1,1} = 'Walden Pond';
A{1,2} = 'June 13, 1997';
A{2,1} = [60,72,65];
A{2,2} = [55,57,56;54,56,55;52,55,53];
```

Type `A` at the command line. You will see

```
A =
    'Walden Pond'    'June 13, 1997'
     [1x3 double]     [3x3 double]
```

You can use the `celldisp` function to display the full contents. For example, typing `celldisp(A)` displays

```
A{1,1} =
   Walden Pond
A{2,1} =
   60  72  65
   .
   .
   .
   etc.
```

Structure Arrays

Structure arrays are composed of *structures.* This class of arrays enables you to store dissimilar arrays together. The elements in structures are accessed using *named elds.* This feature distinguishes them from cell arrays, which are accessed using the standard array indexing operations.

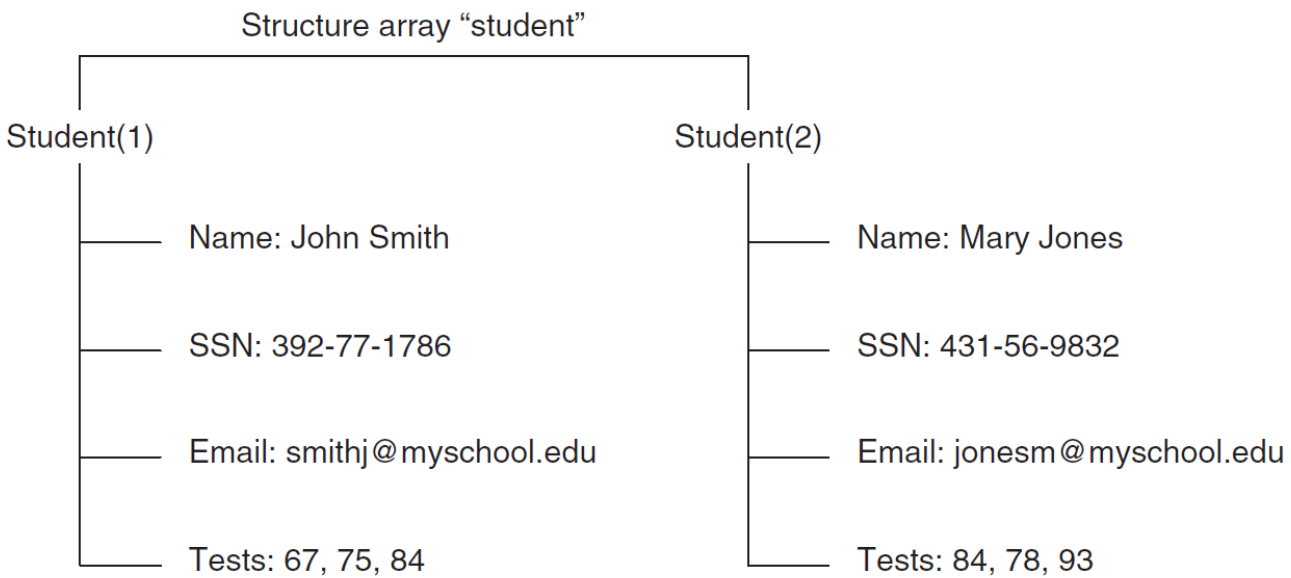


Figure 2.7–1 Arrangement of data in the structure array `student`.

Creating Structures

You can create a structure array by using assignment statements or by using the `struct` function. The following example uses assignment statements to build a structure. Structure arrays use the dot notation (.) to specify and to access the elds. You can type the commands either in the interactive mode or in a script le.

A Student Database

Create a structure array to contain the following types of student data:

- Student name.
- Social Security number.
- Email address.
- Test scores.

Enter the data shown in Figure 2.7–1 into the database.

■ Solution

You can create the structure array by typing the following either in the interactive mode or in a script le. Start with the data for the rst student.

```
student.name = 'John Smith';
student.SSN = '392-77-1786';
student.email = 'smithj@myschool.edu';
student.tests = [67,75,84];
```

If you then type

```
>>student
```

at the command line, you will see the following response:

```
name: 'John Smith'
SSN: = '392-77-1786'
email: = 'smithj@myschool.edu'
tests: = [67 75 84]
```

To determine the size of the array, type `size(student)`. The result is `ans = 1 1`, which indicates that it is a 1×1 structure array.

To add a second student to the database, use a subscript 2 enclosed in parentheses after the structure array's name and enter the new information. For example, type

```
student(2).name = 'Mary Jones';
student(2).SSN = '431-56-9832';
student(2).email = 'jonesm@myschool.edu';
student(2).tests = [84,78,93];
```

A fenced enclosure consists of a rectangle of length L and width $2R$, and a semicircle of radius R, as shown in Figure P20. The enclosure is to be built to have an area A of 1600 ft^2. The cost of the fence is \$40/ft for the curved portion and \$30/ft for the straight sides. Use the `min` function to determine with a resolution of 0.01 ft the values of R and L required to minimize the total cost of the fence. Also compute the minimum cost.

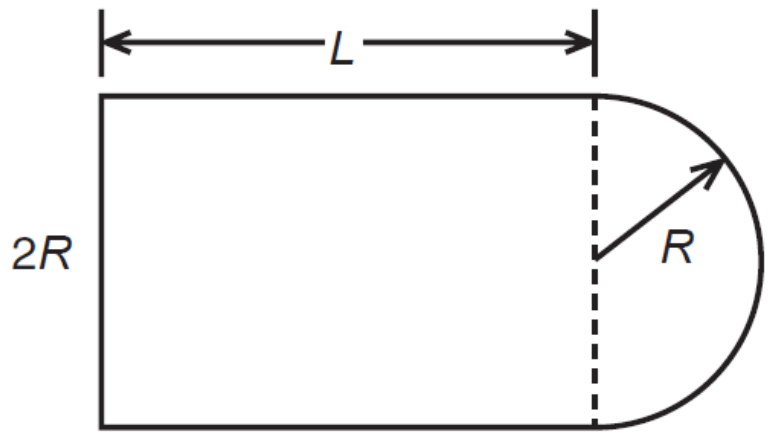


Figure P20

A cable of length L_c supports a beam of length L_b, so that it is horizontal when the weight W is attached at the beam end. The principles of statics can be used to show that the tension force T in the cable is given by

$$T = \frac{L_b L_c W}{D\sqrt{L_b^2 - D^2}}$$

where D is the distance of the cable attachment point to the beam pivot. See Figure P24.

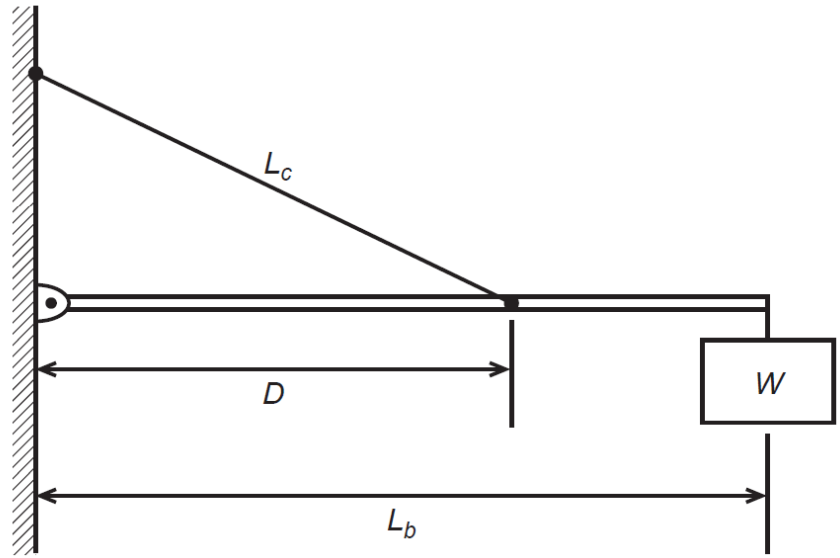


Figure P24

a. For the case where $W = 400$ N, $L_b = 3$ m, and $L_c = 5$ m, use element-by-element operations and the `min` function to compute the value of D that minimizes the tension T. Compute the minimum tension value.

b. Check the sensitivity of the solution by plotting T versus D. How much can D vary from its optimal value before the tension T increases 10 percent above its minimum value?

The following formulas are commonly used by engineers to predict the lift and drag of an airfoil:

$$L = \frac{1}{2}\rho C_L S V^2$$

$$D = \frac{1}{2}\rho C_D S V^2$$

where L and D are the lift and drag forces, V is the airspeed, S is the wing span, ρ is the air density, and C_L and C_D are the *lift* and *drag* coef cients. Both C_L and C_D depend on α, the angle of attack, the angle between the relative air velocity and the airfoil's chord line.

Wind tunnel experiments for a particular airfoil have resulted in the following formulas.

$$C_L = 4.47 \times 10^{-5}\alpha^3 + 1.15 \times 10^{-3}\alpha^2 + 6.66 \times 10^{-2}\alpha + 1.02 \times 10^{-1}$$

$$C_D = 5.75 \times 10^{-6}\alpha^3 + 5.09 \times 10^{-4}\alpha^2 + 1.8 \times 10^{-4}\alpha + 1.25 \times 10^{-2}$$

where α is in degrees.

Plot the lift and drag of this airfoil versus V for $0 \leq V \leq 150$ mi/hr (you must convert V to ft /sec; there is 5280 ft/mi). Use the values $\rho = 0.002378$ slug/ft^3 (air density at sea level), $\alpha = 10°$, and $S = 36$ ft. The resulting values of L and D will be in pounds.

The lift-to-drag ratio is an indication of the effectiveness of an airfoil. Referring to Problem 43, the equations for lift and drag are

$$L = \frac{1}{2}\rho C_L S V^2$$

$$D = \frac{1}{2}\rho C_D S V^2$$

where, for a particular airfoil, the lift and drag coef cients versus angle of attack α are given by

$$C_L = 4.47 \times 10^{-5}\alpha^3 + 1.15 \times 10^{-3}\alpha^2 + 6.66 \times 10^{-2}\alpha + 1.02 \times 10^{-1}$$

$$C_D = 5.75 \times 10^{-6}\alpha^3 + 5.09 \times 10^{-4}\alpha^2 + 1.81 \times 10^{-4}\alpha + 1.25 \times 10^{-2}$$

Using the rst two equations, we see that the lift-to-drag ratio is given simply by the ratio C_L/C_D.

$$\frac{L}{D} = \frac{\frac{1}{2}\rho C_L S V^2}{\frac{1}{2}\rho C_D S V^2} = \frac{C_L}{C_D}$$

Plot L/D versus α for $-2° \leq \alpha \leq 22°$. Determine the angle of attack that maximizes L/D.

Functions and Files

Table 3.1–1 Some common mathematical functions

Exponential	
`exp(x)`	Exponential; e^x.
`sqrt(x)`	Square root; $\sqrt{x}$.
Logarithmic	
`log(x)`	Natural logarithm; ln x.
`log10(x)`	Common (base-10) logarithm; log $x = \log_{10} x$.
Complex	
`abs(x)`	Absolute value; x.
`angle(x)`	Angle of a complex number x.
`conj(x)`	Complex conjugate.
`imag(x)`	Imaginary part of a complex number x.
`real(x)`	Real part of a complex number x.
Numeric	
`ceil(x)`	Round to the nearest integer toward ∞.
`fix(x)`	Round to the nearest integer toward zero.
`floor(x)`	Round to the nearest integer toward $-\infty$.
`round(x)`	Round toward the nearest integer.
`sign(x)`	Signum function: $+1$ if $x > 0$; 0 if $x = 0$; -1 if $x < 0$.

Table 3.1–2 Trigonometric functions

Trigonometric*	
`cos(x)`	Cosine; cos x.
`cot(x)`	Cotangent; cot x.
`csc(x)`	Cosecant; csc x.
`sec(x)`	Secant; sec x.
`sin(x)`	Sine; sin x.
`tan(x)`	Tangent; tan x.
Inverse trigonometric†	
`acos(x)`	Inverse cosine; arccos $x = \cos^{-1} x$.
`acot(x)`	Inverse cotangent; arccot $x = \cot^{-1} x$.
`acsc(x)`	Inverse cosecant; arccsc $x = \csc^{-1} x$.
`asec(x)`	Inverse secant; arcsec $x = \sec^{-1} x$.
`asin(x)`	Inverse sine; arcsin $x = \sin^{-1} x$.
`atan(x)`	Inverse tangent; arctan $x = \tan^{-1} x$.
`atan2(y,x)`	Four-quadrant inverse tangent.

*These functions accept x in radians.
†These functions return a value in radians.

Function de nition line	File name
1. `function [area_square] = square(side);`	`square.m`
2. `function area_square = square(side);`	`square.m`
3. `function volume_box = box(height,width,length);`	`box.m`
4. `function [area_circle,circumf] = circle(radius);`	`circle.m`
5. `function sqplot(side);`	`sqplot.m`

Programming with MATLAB

Example 1. Sequential Operations Compute the perimeter p and the area A of a triangle whose sides are a, b, and c. The formulas are

$$p = a + b + c \qquad s = \frac{p}{2} \qquad A = \sqrt{s(s-a)(s-b)(s-c)}$$

1. Enter the side lengths a, b, and c.
2. Compute the perimeter p.
$$p = a + b + c$$
3. Compute the semiperimeter s.
$$s = \frac{p}{2}$$
4. Compute the area A.
$$A = \sqrt{s(s-a)(s-b)(s-c)}$$
5. Display the results p and A.
6. Stop.

The program is

```
a = input('Enter the value of side a: ');
b = input('Enter the value of side b: ');
c = input('Enter the value of side c: ');
p = a + b + c;
s = p/2;
A = sqrt(s*(s-a)*(s-b)*(s-c));
disp('The perimeter is:')
p
```

Example 2. Conditional Operations Given the (x, y) coordinates of a point, compute its polar coordinates (r, θ), where

$$r = \sqrt{x^2 + y^2} \qquad \theta = \tan^{-1}\left(\frac{y}{x}\right)$$

1. Enter the coordinates x and y.
2. Compute the hypoteneuse r.
```
r = sqrt(x^2+y^2)
```
3. Compute the angle θ.
   3.1 If $x \geq 0$
   ```
   theta = atan(y/x)
   ```
   3.2 Else
   ```
   theta = atan(y/x) + pi
   ```
4. Convert the angle to degrees.
```
theta = theta*(180/pi)
```
5. Display the results `r` and `theta`.
6. Stop.

Example 3. Iterative Operations Determine how many terms are required for the sum of the series $10k^2 - 4k + 2, k = 1, 2, 3, \ldots$ to exceed 20 000. What is the sum for this many terms?

Because we do not know how many times we must evaluate the expression $10k^2 - 4k + 2$, we use a `while` loop, which is covered in Section 4.6.

1. Initialize the total to zero.
2. Initialize the counter to zero.
3. While the total is less than 20 000 compute the total.
   3.1 Increment the counter by 1.
   ```
   k = k + 1
   ```
   3.2 Update the total.
   ```
   total = 10*k^2 - 4*k + 2 + total
   ```
4. Display the current value of the counter.
5. Display the value of the total.
6. Stop.

The following program implements the pseudocode. The statements in the `while` loop are executed until the variable `total` equals or exceeds 2×10^4.

```
total = 0;
k = 0;
while total < 2e+4
   k = k+1;
   total = 10*k^2 - 4*k + 2 + total;
end
disp('The number of terms is:')
disp(k)
disp('The sum is:')
disp(total)
```

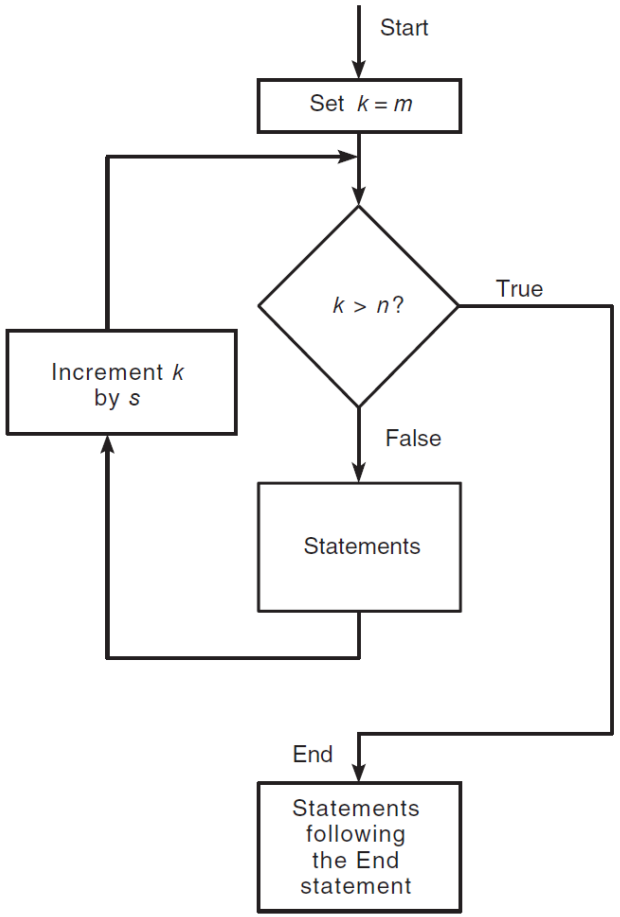


Figure 4.5–1 Flowchart of a `for` loop.

Figure P20 shows a mass-spring model of the type used to design packaging systems and vehicle suspensions, for example. The springs exert a force that is proportional to their compression, and the proportionality constant is the spring constant k. The two side springs provide additional resistance if the weight W is too heavy for the center spring. When the weight W is gently placed, it moves through a distance x before coming to rest. From statics, the weight force must balance the spring forces at this new position. Thus

$$W = k_1 x \qquad \text{if } x < d$$
$$W = k_1 x + 2k_2(x - d) \qquad \text{if } x \geq d$$

These relations can be used to generate the plot of x versus W.

a. Create a function file that computes the distance x, using the input parameters W, k_1, k_2, and d. Test your function for the following two cases, using the values $k_1 = 10^4$ N/m; $k_2 = 1.5 \times 10^4$ N/m; $d = 0.1$ m.

$$W = 500 \text{ N}$$
$$W = 2000 \text{ N}$$

b. Use your function to plot x versus W for $0 \leq W \leq 3000$ N for the values of k_1, k_2, and d given in part *a*.

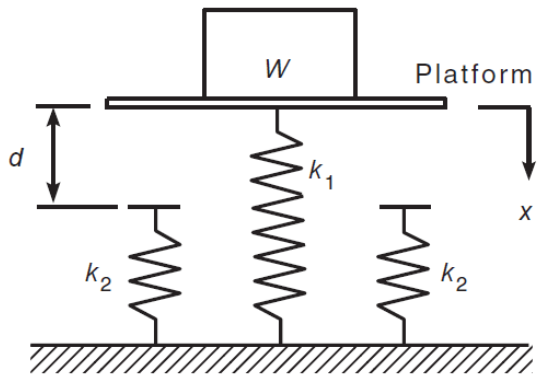


Figure P20

A certain company makes televisions, stereo units, and speakers. Its parts inventory includes chassis, picture tubes, speaker cones, power supplies, and electronics. The inventory, required components, and pro t for each product appear in the following table. Determine how many of each product to make in order to maximize the pro t.

	Product			
	Television	**Stereo unit**	**Speaker unit**	**Inventory**
Requirements				
Chassis	1	1	0	450
Picture tube	1	0	0	250
Speaker cone	2	2	1	800
Power supply	1	1	0	450
Electronics	2	2	1	600
Unit pro t ($)	80	50	40	

A weight W is supported by two cables anchored a distance D apart (see Figure P35). The cable length L_{AB} is given, but the length L_{AC} is to be selected. Each cable can support a maximum tension force equal to W. For the weight to remain stationary, the total horizontal force and total vertical force must each be zero. This principle gives the equations

$$-T_{AB}\cos\theta + T_{AC}\cos\phi = 0$$

$$T_{AB}\sin\theta + T_{AC}\sin\phi = W$$

We can solve these equations for the tension forces T_{AB} and T_{AC} if we know the angles θ and ϕ. From the law of cosines

$$\theta = \cos^{-1}\left(\frac{D^2 + L_{AB}^2 - L_{AC}^2}{2DL_{AB}}\right)$$

From the law of sines

$$\phi = \sin^{-1}\left(\frac{L_{AB}\sin\theta}{L_{AC}}\right)$$

For the given values $D = 6$ ft, $L_{AB} = 3$ ft, and $W = 2000$ lb, use a loop in MATLAB to nd $L_{AC\min}$, the shortest length L_{AC} we can use without T_{AB} or T_{AC} exceeding 2000 lb. Note that the largest L_{AC} can be is 6.7 ft (which corresponds to $\theta = 90°$). Plot the tension forces T_{AB} and T_{AC} on the same graph versus L_{AC} for $L_{AC\min} \le L_{AC} \le 6.7$.

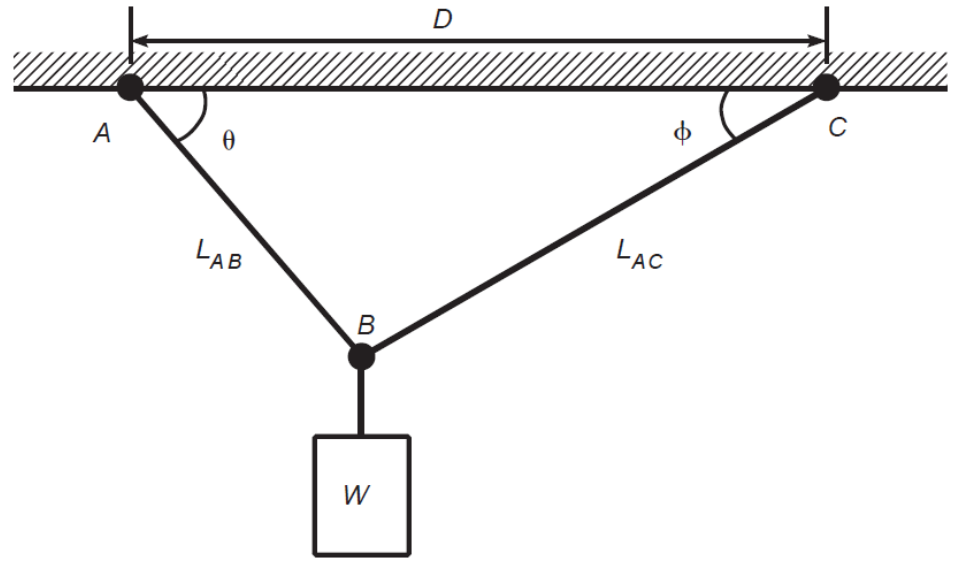


Figure P35

Advanced Plotting

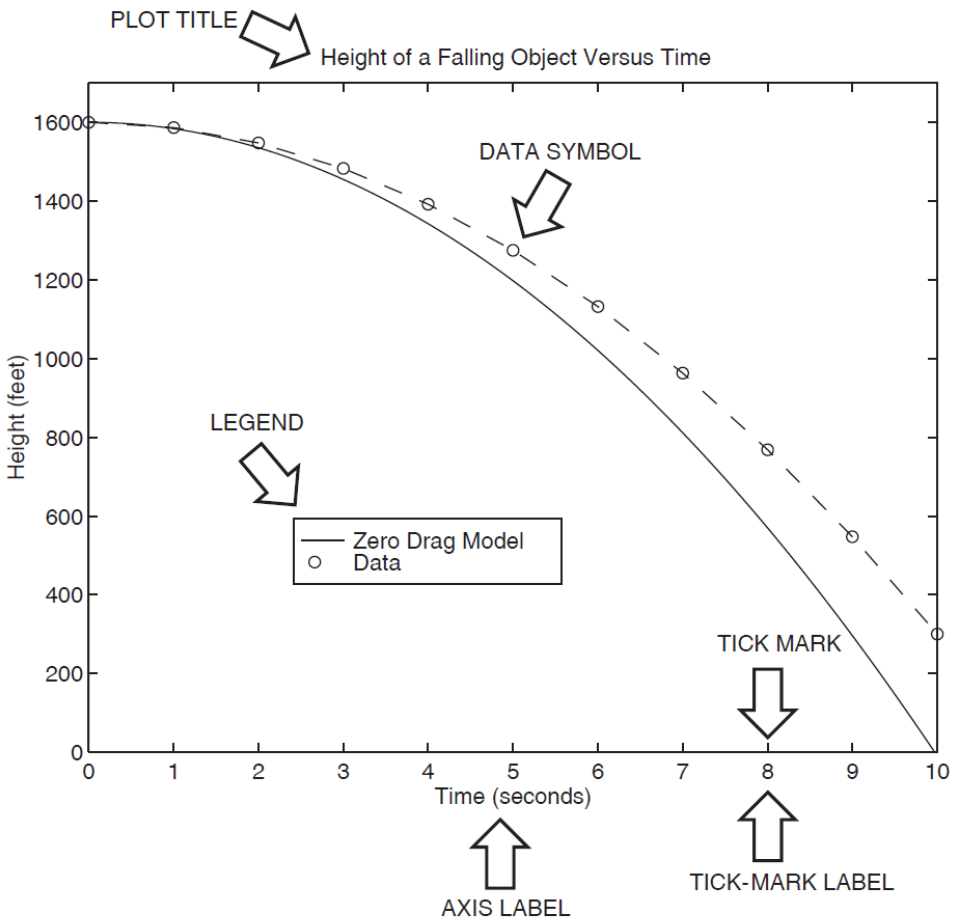


Figure 5.1–1 Nomenclature for a typical *xy* plot.

Table 5.1–2 Basic *xy* plotting commands

Command	Description
`axis([xmin xmax ymin ymax])`	Sets the minimum and maximum limits of the *x* and *y* axes.
`fplot(function,[xmin xmax])`	Performs intelligent plotting of functions, where `function` is a function handle that describes the function to be plotted and `[xmin xmax]` speci es the minimum and maximum values of the independent variable. The range of the dependent variable can also be speci ed. In this case the syntax is `fplot(function, [xmin xmax ymin ymax])`.
`grid`	Displays gridlines at the tick marks corresponding to the tick labels.
`plot(x,y)`	Generates a plot of the array `y` versus the array `x` on rectilinear axes.
`plot(y)`	Plots the values of `y` versus their indices if `y` is a vector. Plots the imaginary parts of `y` versus the real parts if `y` is a vector having complex values.
`polyval(p,x)`	Evaluates the polynomial `p` at speci ed values of its independent variable `x`.
`print`	Prints the plot in the Figure window.
`title('text')`	Puts text in a title at the top of a plot.
`xlabel('text')`	Adds a text label to the *x* axis (the abscissa).
`ylabel('text')`	Adds a text label to the *y* axis (the ordinate).

Table 5.2–2 Plot enhancement commands

Command	Description
`gtext('text')`	Places the string `text` in the Figure window at a point speci ed by the mouse.
`hold`	Freezes the current plot for subsequent graphics commands.
`legend('leg1','leg2',...)`	Creates a legend using the strings `leg1, leg2`, and so on and enables its placement with the mouse.
`plot(x,y,u,v)`	Plots, on rectilinear axes, four arrays: `y` versus `x` and `v` versus `u`.
`plot(x,y,'type')`	Plots the array `y` versus the array `x` on rectilinear axes, using the line type, data marker, and colors speci ed in the string `type`. See Table 5.2–1.
`plot(A)`	Plots the columns of the $m \times n$ array `A` versus their indices and generates *n* curves.
`plot(P,Q)`	Plots array `Q` versus array `P`. See the text for a description of the possible variants involving vectors and/or matrices: `plot(x,A)`, `plot(A,x)`, and `plot(A,B)`.
`subplot(m,n,p)`	Splits the Figure window into an array of subwindows with *m* rows and *n* columns and directs the subsequent plotting commands to the *p*th subwindow.
`text(x,y,'text')`	Places the string `text` in the Figure window at a point speci ed by coordinates `x, y`.

Table 5.2–3 Specialized plot commands

Command	Description
`bar(x,y)`	Creates a bar chart of `y` versus `x`.
`loglog(x,y)`	Produces a log-log plot of `y` versus `x`.
`plotyy(x1,y1,x2,y2)`	Produces a plot with two *y* axes, `y1` on the left and `y2` on the right.
`polar(theta,r,'type')`	Produces a polar plot from the polar coordinates `theta` and `r`, using the line type, data marker, and colors speci ed in the string `type`.
`semilogx(x,y)`	Produces a semilog plot of `y` versus `x` with logarithmic abscissa scale.
`semilogy(x,y)`	Produces a semilog plot of `y` versus `x` with logarithmic ordinate scale.
`stairs(x,y)`	Produces a stairs plot of `y` versus `x`.
`stem(x,y)`	Produces a stem plot of `y` versus `x`.

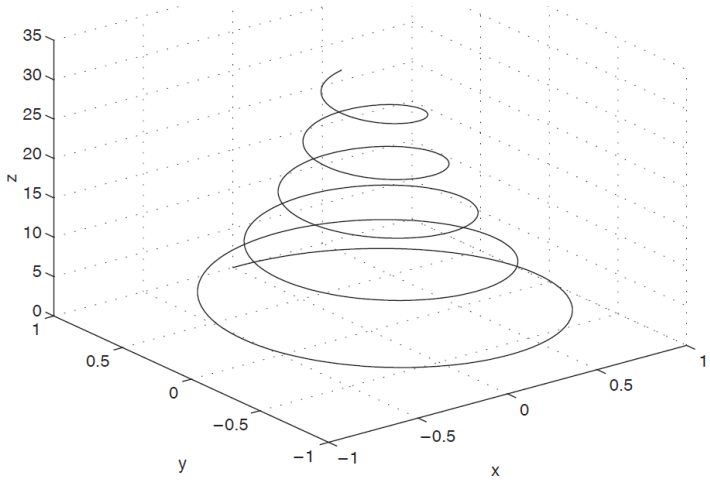


Figure 5.4–1 The curve $x = e^{-0.05t} \sin t$, $y = e^{-0.05t} \cos t$, $z = t$ plotted with the `plot3` function.

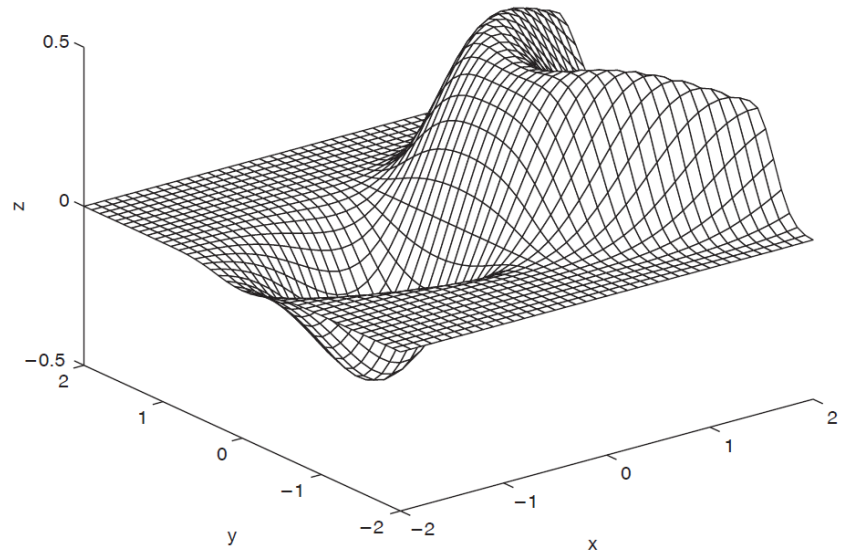


Figure 5.4–2 A plot of the surface $z = xe^{-[(x-y^2)^2+y^2]}$ created with the `mesh` function.

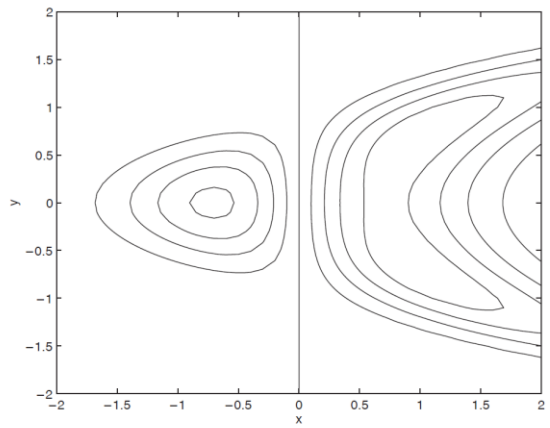


Figure 5.4–3 A contour plot of the surface $z = xe^{-[(x-y^2)^2+y^2]}$ created with the `contour` function.

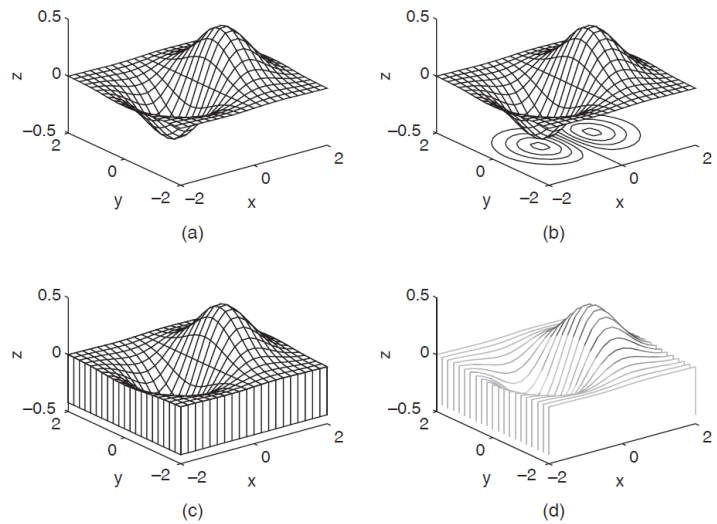


Figure 5.4–4 Plots of the surface $z = xe^{-(x^2+y^2)}$ created with the `mesh` function and its variant forms: `meshc`, `meshz`, and `waterfall`. (a) `mesh`, (b) `meshc`, (c) `meshz`, (d) `waterfall`.

Table 5.4–1 Three-dimensional plotting functions

Function	Description
`contour(x,y,z)`	Creates a contour plot.
`mesh(x,y,z)`	Creates a three-dimensional mesh surface plot.
`meshc(x,y,z)`	Same as `mesh` but draws a contour plot under the surface.
`meshz(x,y,z)`	Same as `mesh` but draws a series of vertical reference lines under the surface.
`surf(x,y,z)`	Creates a shaded three-dimensional mesh surface plot.
`surfc(x,y,z)`	Same as `surf` but draws a contour plot under the surface.
`[X,Y] = meshgrid(x,y)`	Creates the matrices `X` and `Y` from the vectors `x` and `y` to de ne a rectangular grid.
`[X,Y] = meshgrid(x)`	Same as `[X,Y] = meshgrid(x,x)`.
`waterfall(x,y,z)`	Same as `mesh` but draws mesh lines in one direction only.

The height $h(t)$ and horizontal distance $x(t)$ traveled by a ball thrown at an angle A with a speed v are given by

$$h(t) = vt \sin A - \frac{1}{2}gt^2$$

$$x(t) = vt \cos A$$

At Earth's surface the acceleration due to gravity is $g = 9.81$ m/s^2.

a. Suppose the ball is thrown with a velocity $v = 10$ m/s at an angle of 35°. Use MATLAB to compute how high the ball will go, how far it will go, and how long it will take to hit the ground.

b. Use the values of v and A given in part *a* to plot the ball's *trajectory;* that is, plot h versus x for positive values of h.

c. Plot the trajectories for $v = 10$ m/s corresponding to ve values of the angle A: 20°, 30°, 45°, 60°, and 70°.

d. Plot the trajectories for $A = 45°$ corresponding to ve values of the initial velocity v: 10, 12, 14, 16, and 18 m/s.

Model Building and Regression

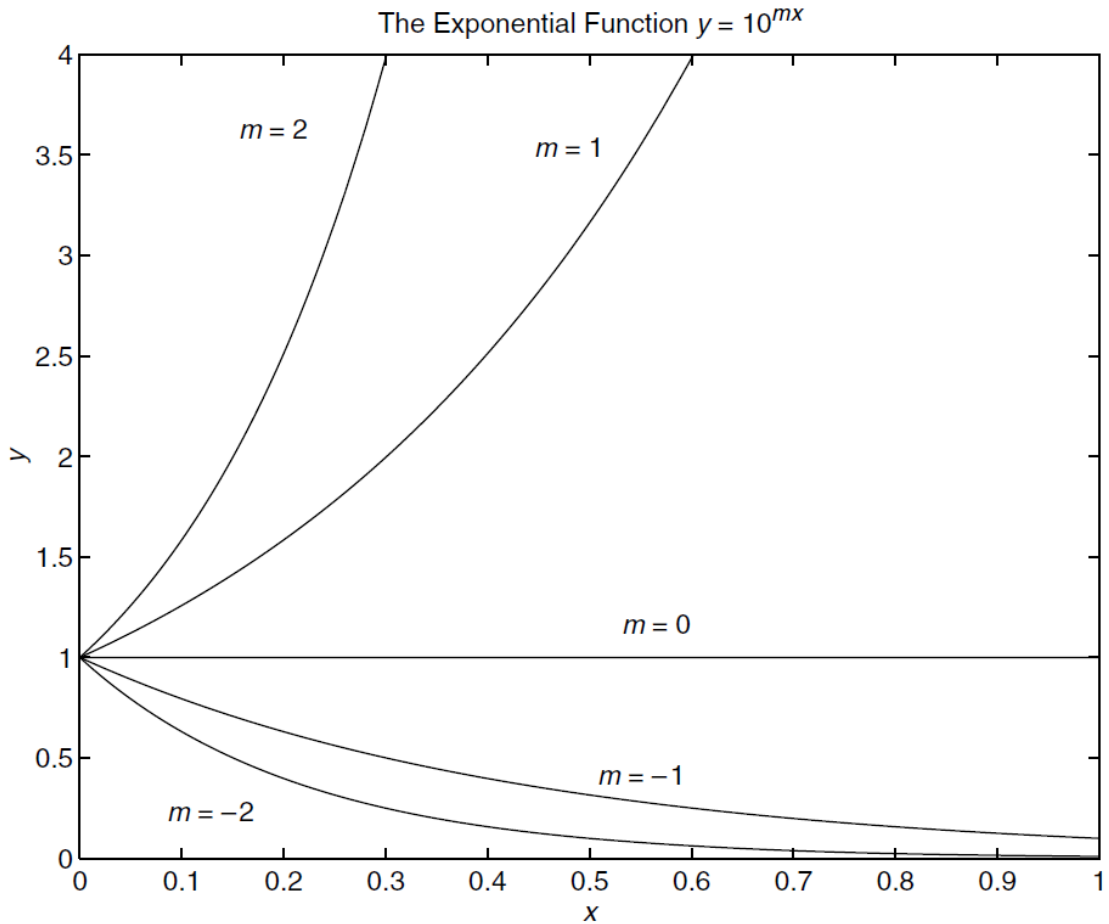


Figure 6.1–1 Examples of exponential functions.

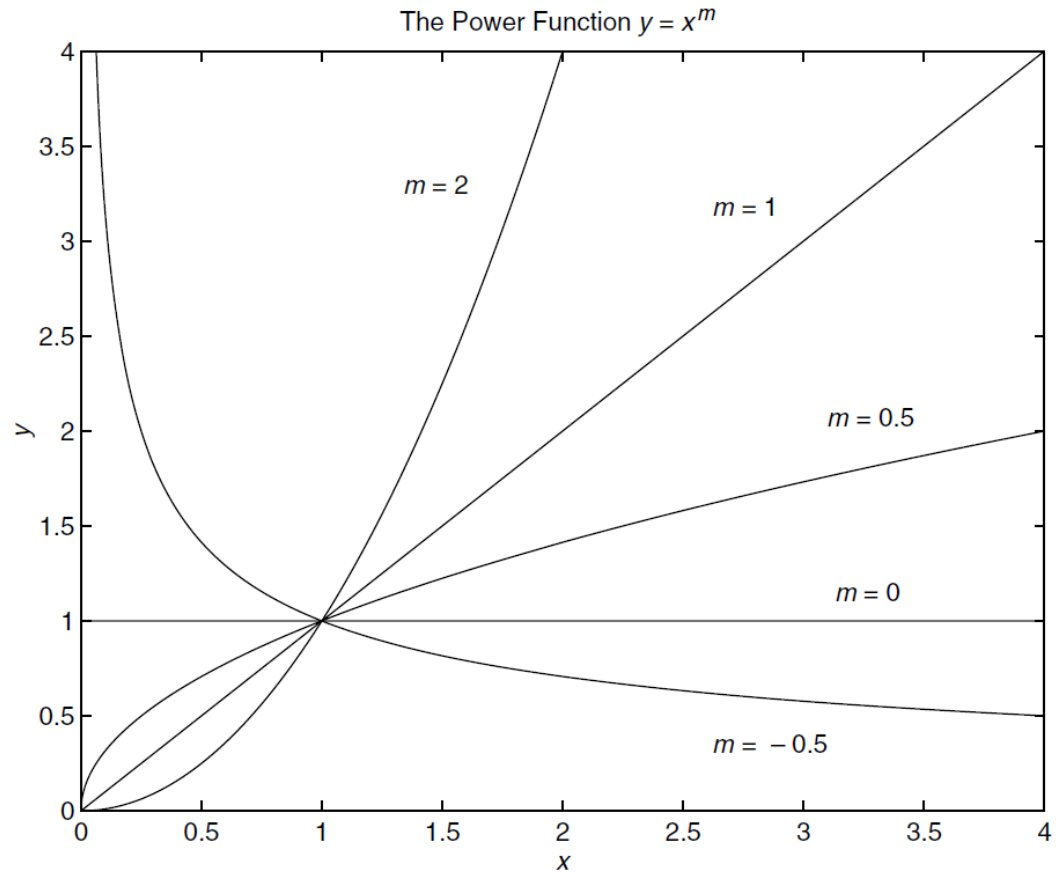


Figure 6.1–2 Examples of power functions.

Table 6.1–1 The `poly t` function

Command	Description
`p = poly t(x,y,n)`	Fits a polynomial of degree n to data described by the vectors x and y, where x is the independent variable. Returns a row vector `p` of length $n + 1$ that contains the polynomial coef cients in order of descending powers.

- **The linear function:** $y = mx + b$. In this case the variables w and z in the polynomial $w = p_1z + p_2$ are the original data variables x and y, and we can nd the linear function that ts the data by typing `p = poly t(x,y,1)`. The rst 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 $\log_{10} y = m \log_{10} x + \log_{10} b$, which has the form $w = p_1z + p_2$, 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 nd the power function that ts the data by typing `p = poly t(log10(x), log10(y),1)`. The rst element p_1 of the vector `p` will be m, and the second element p_2 will be $\log_{10} b$. We can nd 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 $w = p_1z + p_2$, 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 nd the exponential function that ts the data by typing `p = poly t(x, log10(y),1)`. The rst element p_1 of the vector `p` will be m, and the second element p_2 will be $\log_{10} b$. We can nd b from $b = 10^{p_2}$.

The Quality of a Curve Fit

The least-squares criterion used to t a function $f(x)$ is the sum of the squares of the residuals J. It is de ned as

$$J = \sum_{i=1}^{m} [f(x_i) - y_i]^2 \tag{6.2–1}$$

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

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

$$S = \sum_{i=1}^{m} (y_i - \bar{y})^2 \tag{6.2–2}$$

This formula can be used to compute another measure of the quality of the curve t, the *coef cient of determination,* also known as the *r-squared value.* It is de ned as

$$r^2 = 1 - \frac{J}{S} \tag{6.2–3}$$

For a perfect t, $J = 0$ and thus $r^2 = 1$. Thus the closer r^2 is to 1, the better the t. The largest r^2 can be is 1. 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. Thus the ratio J/S indicates the fractional variation 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 t accounts for at least 99 percent of the data variation. This value corresponds to $r^2 \geq 0.99$.

The Basic Fitting Interface

The population data for a certain country are as follows:

Year	2004	2005	2006	2007	2008	2009
Population (millions)	10	10.9	11.7	12.6	13.8	14.9

Obtain a function that describes these data. Plot the function and the data on the same plot. Estimate when the population will be double its 2004 size.

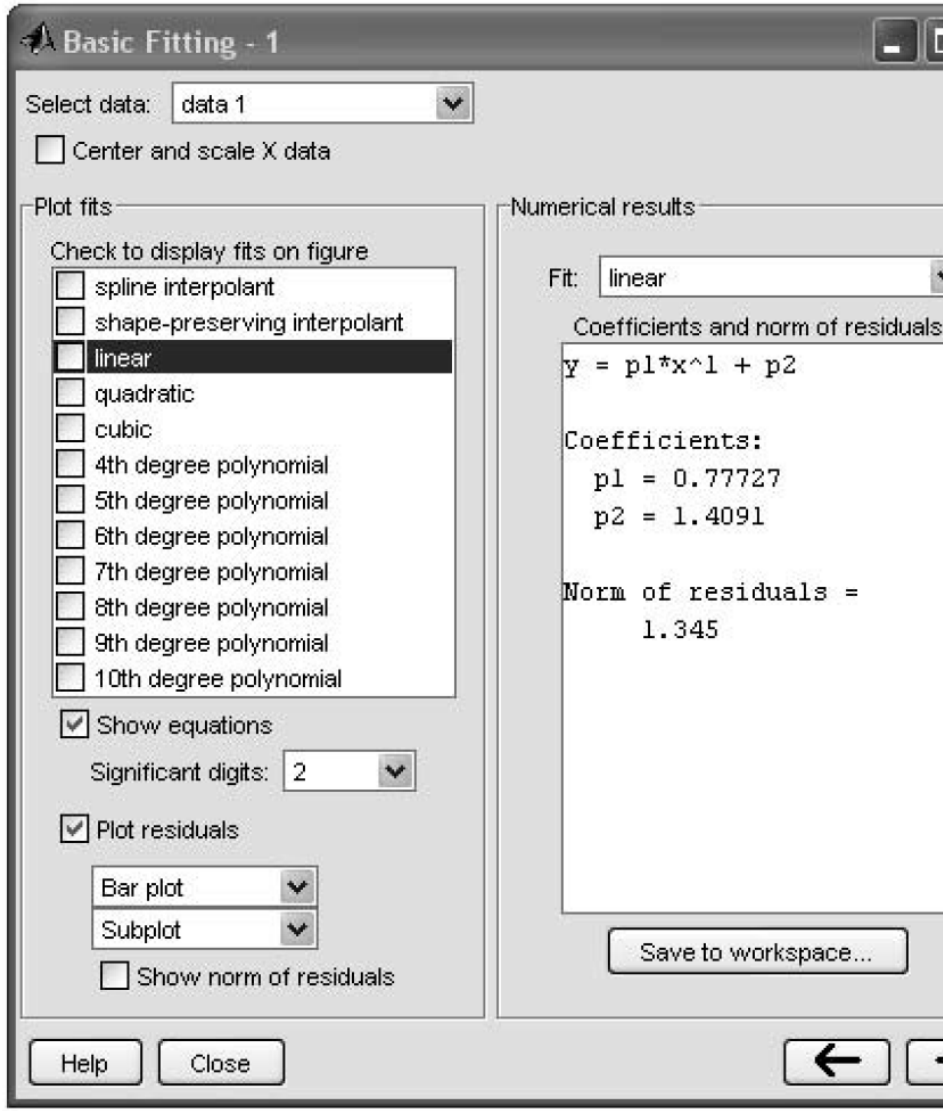


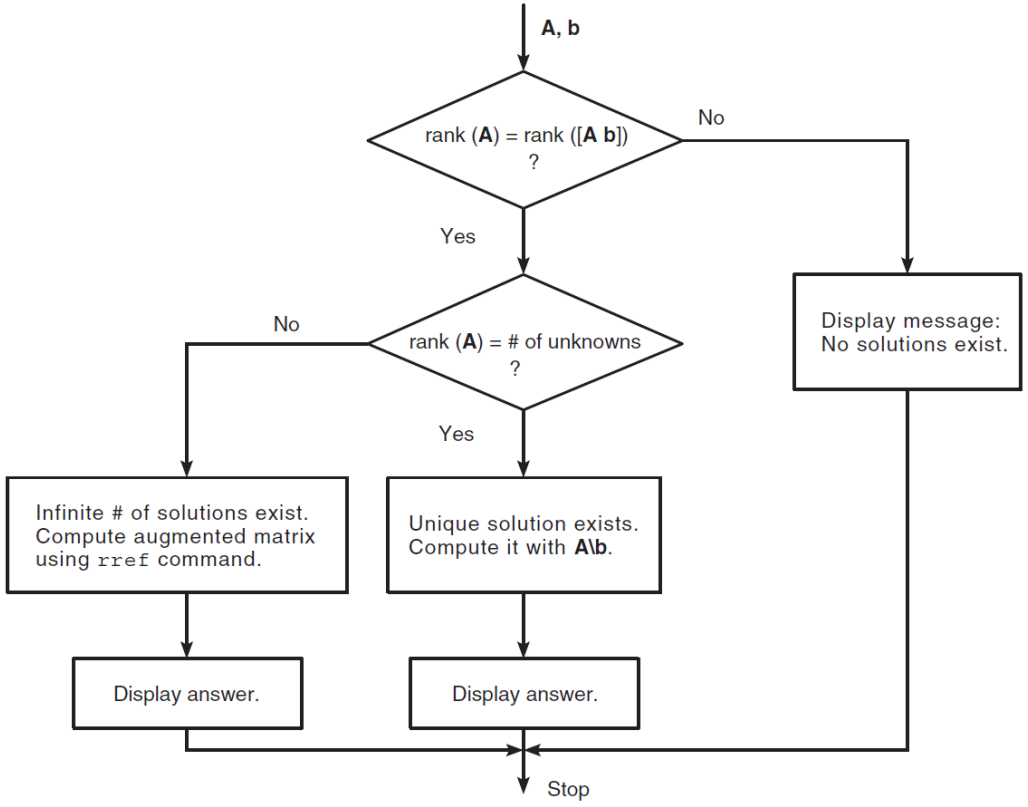
Figure 6.3–1 The Basic Fitting interface.

The solubility of salt in water is a function of the water temperature. Let S represent the solubility of NaCl (sodium chloride) as grams of salt in 100 g of water. Let T be temperature in °C. Use the following data to obtain a curve t for S as a function of T. Use the t to estimate S when $T = 25$°C.

T (°C)	S (g NaCl/100 g H_2O)
10	35
20	35.6
30	36.25
40	36.9
50	37.5
60	38.1
70	38.8
80	39.4
90	40

Linear Algebraic Equations

- Matrix Methods for Linear Equations
- The Left Division Method
- Underdetermined Systems
- Overdetermined Systems
- A General Solution Program



Symbolic Expressions and Algebra

Useful Symbolic Functions

Function	Description
`solve( )`	solve equation(s)
`subs( )`	replace a symbolic variable with a numeric value or another symbolic variable
`double( )`	convert symbolic number to an actual number
`char( )`	convert symbolic expression/equation to a string
`vpa( )`	reformat symbolic fraction numbers (common and often unwieldy) to symbolic decimal numbers having a specified significant figures
`poly2sym( )`	create symbolic polynomial from array of coefficients
`pretty( )`	makes equation pretty (ASCII art)
`ezplot( )`	plots symbolic equation
`ezsurf( )`	surface plot of a symbolic equation
`symsum( )`	evaluates the sum of a series
`diff( )`	differentiates an equation*
`limit( )`	finds the limit of an equation*
`int( )`	integrates an equation*

We want to nd the intersection points of two circles. The rst circle has a radius of 2 and is centered at $x = 3$, $y = 5$. The second circle has a radius b and is centered at $x = 5$, $y = 3$. See Figure 11.3–1.

(a) Find the (x, y) coordinates of the intersection points in terms of the parameter b.

(b) Evaluate the solution for the case where $b = \sqrt{3}$.

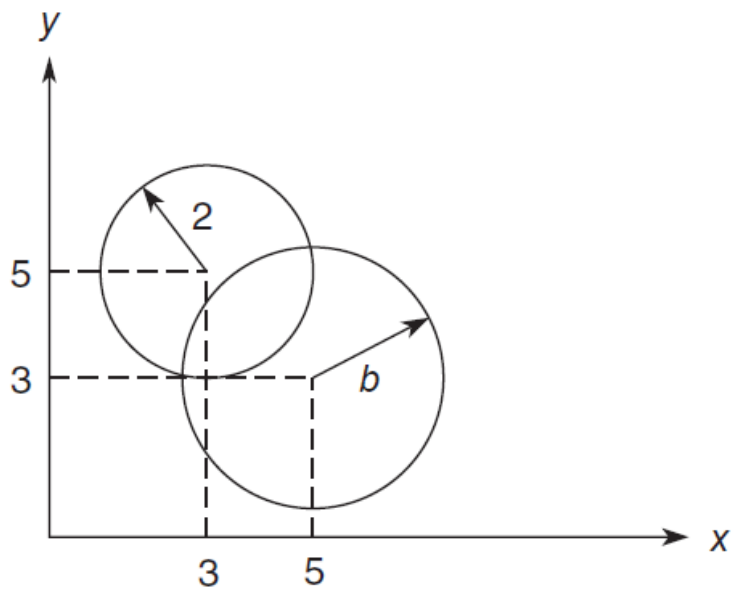


Figure 11.3–1 Intersection points of two circles.

The equation for the voltage y across the capacitor of an RC circuit is

$$RC\frac{dy}{dt} + y = \upsilon(t)$$

where $\upsilon(t)$ is the applied voltage. Suppose that $RC = 0.2$ s and that the capacitor voltage is initially 2 V. If the applied voltage goes from 0 to 10 V at $t = 0$, use MuPAD to determine the voltage $y(t)$.

A rocket's mass decreases as it burns fuel. The equation of motion for a rocket in vertical flight can be obtained from Newton's law and is

$$m(t)\frac{d\upsilon}{dt} = T - m(t)g$$

where T is the rocket's thrust and its mass as a function of time is given by $m(t) = m_0(1 - rt/b)$. The rocket's initial mass is m_0, the burn time is b, and r is the fraction of the total mass accounted for by the fuel. Use the values $T = 48\,000$ N, $m_0 = 2200$ kg, $r = 0.8$, $g = 9.81$ m/s^2, and $b = 40$ s.

a. Use MuPAD to compute the rocket's velocity as a function of time for $t \le b$.

b. Use MuPAD to compute the rocket's velocity at burnout.

This equation describes the motion of a mass connected to a spring with viscous friction on the surface

$$m\ddot{y} + c\dot{y} + ky = f(t)$$

where $f(t)$ is an applied force. The position and velocity of the mass at $t = 0$ are denoted by x_0 and v_0. Use MuPAD to answer the following questions.

a. What is the free response in terms of x_0 and v_0 if $m = 3$, $c = 18$, and $k = 102$?

b. What is the free response in terms of x_0 and v_0 if $m = 3$, $c = 39$, and $k = 120$?

This equation describes the motion of a certain mass connected to a spring with viscous friction on the surface

$$3\ddot{y} + 18\dot{y} + 102y = f(t)$$

where $f(t)$ is an applied force. Suppose that $f(t) = 0$ for $t < 0$ and $f(t) = 10$ for $t \geq 0$.

a. Use MuPAD to obtain $y(t)$ if $y(0) = \dot{y}(0) = 0$.

b. Use MuPAD to obtain $y(t)$ if $y(0) = 0$ and $\dot{y}(0) = 10$.

The equations for an armature-controlled dc motor follow. The motor's current is i and its rotational velocity is ω.

$$L\frac{di}{dt} = -Ri - K_e\omega + v(t)$$

$$I\frac{d\omega}{dt} = K_T i - c\omega$$

where L, R, and I are the motor's inductance, resistance, and inertia; K_T and K_e are the torque constant and back-emf constant; c is a viscous damping constant; and $v(t)$ is the applied voltage.

Use the values $R = 0.8\ \Omega$, $L = 0.003$ H, $K_T = 0.05$ N·m/A, $K_e = 0.05$ V · s/rad, $c = 0$, and $I = 8 \times 10^{-5}$ kg · m^2.

Suppose the applied voltage is 20 V. Use MuPAD to obtain the motor's speed and current versus time for zero initial conditions. Choose a final time large enough to show the motor's speed becoming constant.