CHAPTER 14

Dynamical Systems

The objective of this chapter is to discuss the importance of learning tools like MATLAB.

In this chapter we examine MATLAB's application to four relatively simple problems in engineering: the deflection of a cantilever beam subject to a uniform load; a single-loop closed electrical circuit; the free-fall problem; and an extension of the projectile problem discussed in Chapter 3. The first problem involves a structural element you investigate in a first course in engineering mechanics—the cantilever beam, which is one of the primary elements of engineered structures such as buildings and bridges. We examine the deflection of this beam with a constant cross-section when subject to a uniform distribution of load (e.g., its own weight).

The second problem involves the equation that describes the "flow" of electrical current in a simple closed-loop electrical circuit. You meet this type of problem in a first course in electrical science.

The third problem involves the free fall of an object in a gravitational field with constant acceleration, g. This is one of the first problems you encounter in a first course in physics. We examine the effect of friction on the free-fall problem and learn that with friction (i.e., air resistance) the object can reach a terminal velocity.

The fourth problem extends the projectile problem that takes into account air resistance (you learn about air resistance in your first course in fluid mechanics). You will learn why golfers don't hit the ball from the tee at 45 degrees from the horizontal (which is the optimum angle for the furthest distance of travel of a projectile launched in frictionless air).

DOI: 10.1016/B978-0-12-374883-6.00014-X

14.1 CANTILEVER BEAM

In this section we want to examine the problem of the cantilever beam. The beam and its deflection under load are illustrated in Figure 14.1, which is generated by the script M-file created to solve the problem posed next. Many structural mechanics formulae are available in *Formulas for Stress and Strain*, fifth edition, by Raymond J. Roark and Warren C. Young (McGraw-Hill 1982).

For a uniformly loaded span of a cantilever beam attached to a wall at $x = 0$ with the free end at $x = L$, the formula for the vertical displacement from $y = 0$ under the loaded condition, with y the coordinate in the direction opposite that of the load, can be written as follows:

$$Y = \frac{y\,24EI}{wL^4} = -\left(X^4 - 4X^3 + 6X^2\right),$$

where $X = x/L$, E is a material property known as the modulus of elasticity; I is a geometric property of the cross-section of the beam known as the moment of inertia; L is the length of the beam extending from the wall on which it is mounted; and w is the load per unit width of the beam (this is a two-dimensional analysis). The formula was put into dimensionless form to answer the following question: What is the shape of the deflection curve when the beam is in its loaded condition, and how does it compare with its unloaded perfectly horizontal orientation? The answer will be provided graphically.

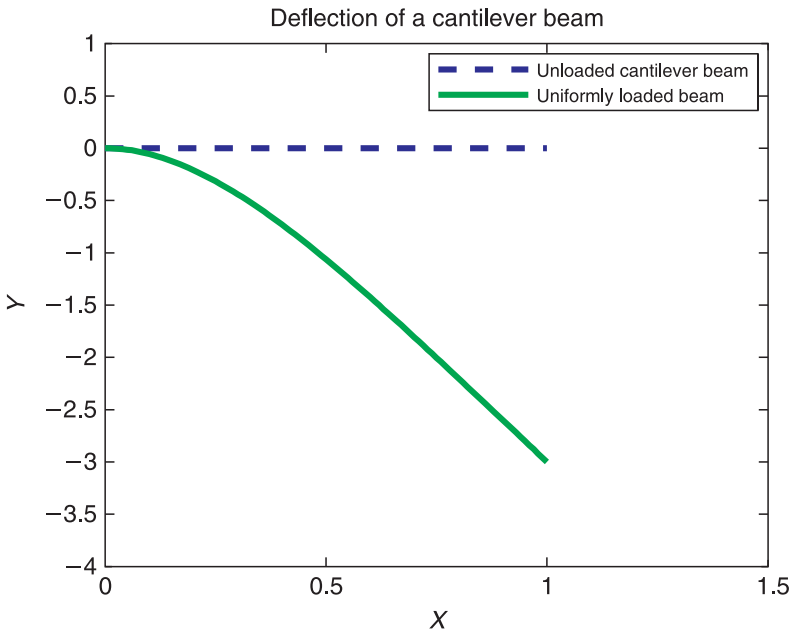

FIGURE 14.1

Vertical deflection of a uniformly loaded cantilever beam.

The problem is easily solved using the following script (the results are plotted in Figure 14.1):

```
%
% The deflection of a cantilever beam under a uniform load
% Script by D.T.V. ........ September 2006
%
% Step 1: Select the distribution of X's from 0 to 1
% where the deflections to be plotted are to be determined.
%
X = 0:.01:1;
%
% Step 2: Compute the deflections Y at each X. Note that YE
% is the unloaded position of all points on the beam.
%
Y = - ( X.^4 - 4 * X.^3 + 6 * X.^2 );
YE = 0;
%
% Step 3: Plot the results to illustrate the shape of the
% deflected beam.
%
plot([0 1],[0 0],'--',X,Y,'LineWidth',2)
axis([0,1.5,-4, 1]),title('Deflection of a cantilever beam')
xlabel('X'),ylabel('Y')
legend('Unloaded cantilever beam','Uniformly loaded beam')
%
% Step 4: Stop
```

It looks like the beam is deflected tremendously, but the actual deflection is not so dramatic once the actual material properties are substituted to determine y and x in, say, meters. The scaling (i.e., rewriting the equation in dimensionless form in terms of the unitless quantities Y and X) provides insight into the shape of the curve. In addition, the shape is independent of the material as long as the material has uniform properties and the geometry has a uniform cross-sectional area (e.g., a rectangular area of height h and width b that is independent of the distance along the span (or length) L).

14.2 ELECTRIC CURRENT

In electrical science you investigate circuits with a variety of components. In this section we will solve the governing equation to examine the dynamics of a single, closed-loop electrical circuit. The loop contains a voltage source, V (e.g., a battery), a resistor, R (i.e., an energy dissipation device), an inductor, L (i.e., an energy storage device), and a switch that is instantaneously closed at time $t = 0$.

From Kirchoff's law, as described in *Circuits, Devices, and Systems* by Ralph J. Smith (John Wiley & Sons, 1967), the equation describing the response of this system from an initial state of zero current is

$$L\frac{di}{dt} + Ri = V$$

where i is the current. At $t = 0$ the switch is engaged to close the circuit and initiate the current. At this instant of time the voltage is applied to the resistor and inductor (which are connected in series) instantaneously. The equation describes the value of i as a function of time after the switch is engaged. For the present purpose, then, we want to solve it to determine i versus t graphically. Rearranging the equation, we get

$$\frac{di}{dt} + \frac{R}{L}i = \frac{V}{L}$$

The solution, by inspection (another method that you learn when you study differential equations), is

$$i = \frac{V}{R}\left(1 - e^{-\frac{R}{L}t}\right)$$

which is checked with the following script:

```
%
% Script to check the solution to the soverning
% equation for a simple circuit, i.e., to check
% that
%       i = (V/R) * (1 -  exp(-R*t/L))
%
% is the solution to the following ODE
%
%       di/dt + (R/L) * i - V/L = 0
%
% Step 1: We will use the Symbolics tools; hence,
% define the symbols as follows
%
       syms i V R L t
%
% Step 2: Construct the solution for i
%
       i = (V/R) * ( 1 - exp(-R*t/L) );
%
% Step 3: Find the derivative of i
%
      didt = diff(i,t);
%
```

```
% Step 4: Sum the terms in ODE
%
       didt + (R/L) * i - V/L;
%
% Step 5: Is the answer ZERO?
%
      simple(ans)
%
% Step 6: What is i at t = 0?
%
subs(i,t,0)
%
% REMARK: Both answers are zero; hence,
%         the solution is correct and the
%         initial condition is correct.
%
% Step 7: To illustrate the behavior of the
%         current, plot i vs. t for V/R = 1
%         and R/L = 1. The curve illustrates
%         the fact that the current approaches
%         i = V/R exponentially.
%
       V = 1; R = 1; L = 1;
       t = 0 : 0.01 : 6;
       i = (V/R) * ( 1 - exp(-R.*t/L) );
       plot(t,i,'ro'), title(Circuit problem example)
xlabel('time, t'),ylabel('current, i')
%
```

Running this script will prove that the solution is correct. Figure 14.2 illustrates the solution.

14.3 FREE FALL

In this section we will use MATLAB to investigate the problem of free fall with friction (or air resistance) and to check the theoretical results found in the literature. We want to determine the effect of air resistance on the distance of free fall in 5 seconds from a location $y = 0$, where the object is initially at rest (y is in the direction of gravity). Hence, we want to determine the distance, $y = L$ that an object falls from a state of rest with and without air resistance.

In *Introduction to Theoretical Mechanics* by R. A. Becker (McGraw-Hill 1954), the equations for free fall are given. The three cases of interest are as follows:

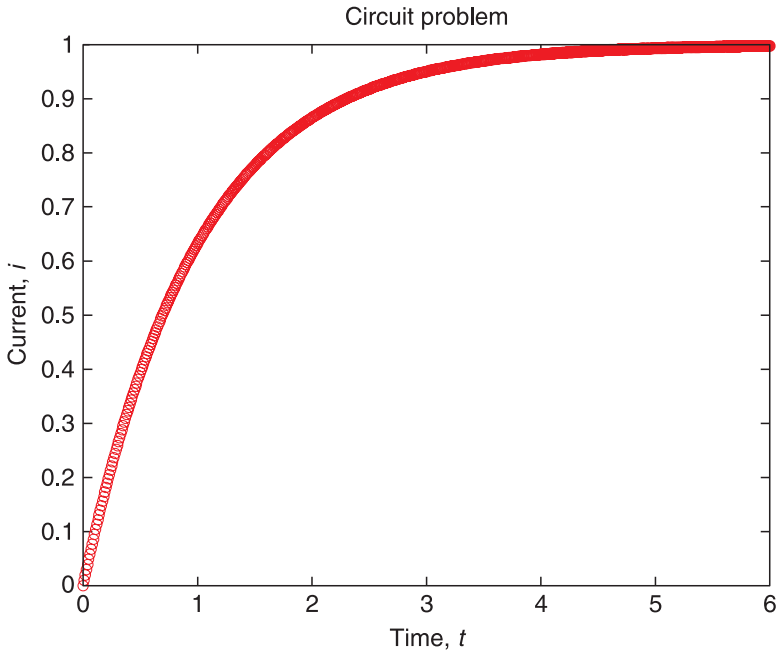

FIGURE 14.2

Exponential approach to steady current condition of a simple RL circuit with an instantaneously applied constant voltage.

Case 1. Without air resistance:

$$a = \frac{d^2y}{dt^2} = g, \quad v = \frac{dy}{dt} = gt, \quad y = \frac{1}{2}gt^2$$

where a is the acceleration of the object, v is its speed, and y is the distance of its free fall from the start of motion at $t = 0$.

Case 2. With resistance proportional to the linear power of velocity:

$$a = \frac{d^2y}{dt^2} = g - k\frac{dy}{dt}, \quad v = \frac{dy}{dt} = \frac{g}{k}\left(1 - e^{-kt}\right), \quad y = \frac{g}{k}t - \frac{g}{k^2}\left(1 - e^{-kt}\right)$$

Case 3. With resistance proportional to the second power of velocity:

$$a = \frac{d^2y}{dt^2} = g - k\left(\frac{dy}{dt}\right)^2, \quad v = \frac{dy}{dt} = \frac{1}{2}\frac{g}{k}\tanh\left(\frac{gk}{2}t\right), \quad y = \frac{1}{k}\log_e\left[\cosh(gkt/2)\right]$$

For all three cases, the initial condition is $y = v = 0$ at $t = 0$. (You will learn more about ordinary differential equations in your third semester of mathematics. In addition, in a first course in fluid mechanics you will learn some of the details

about air resistance. In particular, for air resistance associated with "laminar flow" problems, case 2 applies. In many practical situations, case 3 is usually the one of interest; this is where "turbulent flow" is important.)

Let us consider the following problem: Assume the above equations are correct. Note that they are written for a unit of mass. Also assume $g = 9.81$ m/s^2 and $k = 0.2$ for the air-drag parameter. Finally, answer the following:

- What is y in meters for $t = 5$ seconds of free fall for the three cases?
- What are the terminal speeds in meters per second for cases 2 and 3 for the specified air-drag parameter?
- Is the terminal speed reached at $t = 5$ seconds?

Note that

- The terminal speeds are g/k and $(g/k)/2$ for cases 2 and 3, respectively, where the terminal speed is the time-independent, or steady, speed reached after a sufficient distance of free fall. This is the speed at which the gravitational force balances the air resistance force. From Part 1, the Essentials, we learned that MATLAB is quite useful in this type of problem because it has the capability of computing the elementary functions in the above formulae.
- As part of providing an answer to the questions raised, we want to examine the distance of free fall within the 5 seconds of flight time using the equations for y given above. If we examine the influence of air resistance on free fall graphically, we need to plot the distance y versus t from $t = 0$ to $t = 5$ seconds. We plot all three curves on one figure for direct comparison, and show that for a short period of time (significantly less than 5 seconds) after the onset of motion, all three curves are on top of each other. We also show that at $t = 5$ seconds the distances of free fall are quite different.

The steps in the structure plan and the MATLAB code are as follows:

```
%
%     Free fall analysis (saved as FFall.m):
%      Comparison of exact solutions of free
%      fall with zero, linear and quadratic
%      friction for t = 0 to 5 seconds.
%
% Script by D. T. V. ......... September 2006.
%     Revised by D.T.V. ..........  November 2008.
%
% Step 1: Specify constants
%
```

```
% Friction coefficient provided in the problem statement.
k = 0.2;
% Acceleration of gravity in m/s/s.
g = 9.81;
%
% Step 2: Selection of time steps for computing solutions
%
dt = .01;
%
% Step 3: Set initial condition (the same for all cases)
%
t(1) = 0.; v(1) = 0.; y(1) = 0.;
%
t = 0:dt:5;
%
% Step 4: Compute exact solutions at each time step
%         from t = 0 to 5.
%
%    (a) Without friction:
%
         v = g * t;
         y = g * t.^2 * 0.5;
%
%    (b) Linear friction
%
         velf = (g/k) * (1. - exp(-k*t));
      yelf = (g/k) * t - (g/(k^2)) * (1.-exp(-k*t));
%
%    (c) Quadratic friction
%
      veqf = sqrt(g/k) * tanh( sqrt(g*k) * t);
      yeqf = (1/k) * log(cosh( sqrt(g*k) * t) );
%
% Step 5: Computation of the terminal speeds
%         (cases with friction)
%
         velfT = g/k;
         veqfT = sqrt(g/k);
%
% Step 6: Graphical comparison
%
 plot(t,y,t,yelf,t,yeqf)
 title('Fig 1. Comparison of results')
 xlabel(' Time, t')
 ylabel(' Distance, y ')
```

```
 figure
 plot(t,v,t,velf,t,veqf)
 title('Fig. 2. Comparison of results')
 xlabel(' Time, t')
 ylabel(' Speed, v ')
%
% Step 7: Comparison of distance and speed at t = 5
%
disp(' ');
fprintf(' y(t) = %f, yelf(t) = %f, yeqf(t) = %f at t = %f\n',...
          y(501),yelf(501),yeqf(501),t(501))
disp(' ');
fprintf(' v(t) = %f, velf(t) = %f, veqf(t) = %f at t = %f\n',...
          y(501),yelf(501),yeqf(501),t(501))
%
% Step 8: Comparison of terminal velocity
%
disp(' ');
fprintf('  velfT = %f, veqfT = %f\n',...
   velfT,veqfT)
%
% Step 9: Stop
%
```

The Command Window execution of this file (named `FFall.m`) gives the comparisons in Figures 14.3 and 14.4 and the printed results as follows:

```
≫ FFall

  y(t) = 122.625000, yelf(t) = 90.222433, yeqf(t) = 31.552121 at t = 5.000000
  v(t) = 49.050000, velf(t) = 31.005513, veqf(t) = 7.003559 at t = 5.000000
  velfT = 49.050000, veqfT = 7.003571
```

The figures illustrate, as we may have expected, that for no friction the object falls the furthest distance. The case with quadratic friction reaches terminal velocity well within the 5 seconds examined. The linear friction case does not reach terminal speed, yet it moves at a slower velocity as compared with the no-friction case.

Keep in mind that a unit mass object falls with a friction coefficient $k = 0.2$. The same k is used for both the second and third cases. Within the first half-second from the time of release the three curves are not distinguishable, illustrating the fact that it takes a little time before the friction effects are felt. At 5 seconds after release, the speed and the distance fallen are quite different. It is not surprising that quadratic friction slows the object quicker, because the air resistance (or

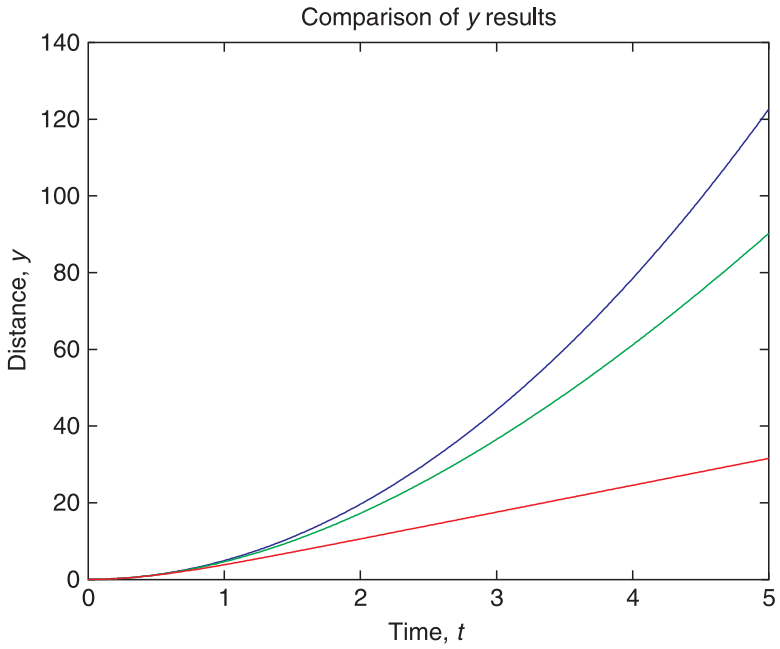

FIGURE 14.3

Comparison of free-fall distance: top curve, no friction; middle curve, linear friction; bottom curve, quadratic friction.

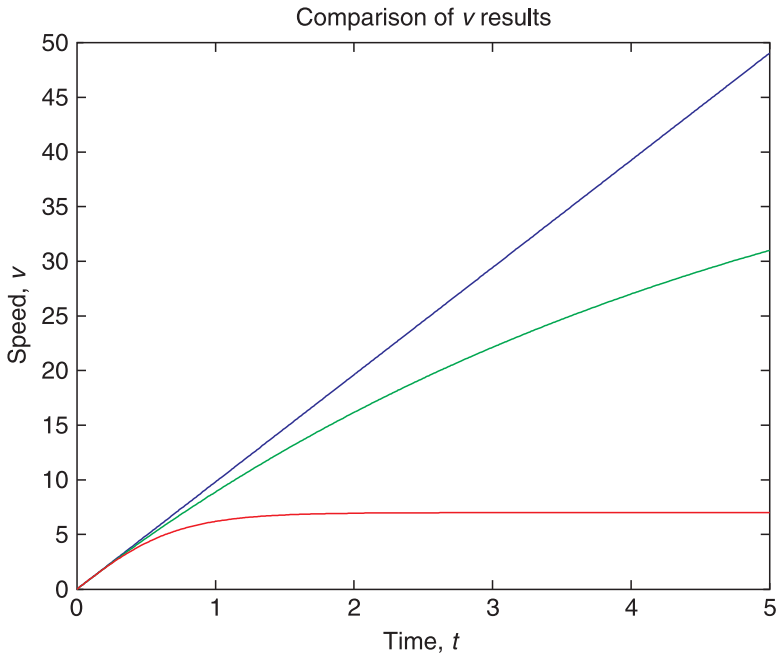

FIGURE 14.4

Comparison of free-fall speed: top curve, no friction; middle curve, linear friction; bottom curve, quadratic friction.

friction) is proportional to the speed squared, which is significantly larger than speed to the first power (as it is in the linear-friction case).

The above analysis is based on the exact solutions to the free-fall problem. Let us now use the symbolic tools to check the theoretical results found in the literature. We will examine case 2, linear friction. The following script was implemented in the Command Window. The responses of MATLAB are also reproduced.

```
%
% The formula for the distance of free fall
% of an object from rest with linear friction
% is as follows:
%
% y = (g / k) * t - (g/k^2) * [ 1 - exp^(- k t)].
%
% To check the theory, we want to differentiate
% this twice to determine the formulas for velocity
% v and acceleration a, respectively. The results
% should match the published results.
%
% Step 1: Define the symbolic variables
%
syms g k t y
%
% Step 2: Write the formula for y = f(t)
%
y = (g/k) * t - (g/k^2) * ( 1 - exp(-k * t));
%
% Step 3: Determine the velocity
%
v = diff(y,t);
%
% Step 4: Determine the acceleration
%
a = diff(v,t);
%
% Step 5: Print the v and a formulas and compare with
% the published results
%
v, a
%
% Step 6: Determine a from published formula and v.
%
```

```
a2 = g - k * v;
% Step 7: Simplify to simplest form
a2 = simple(a2)
%
% Step 6: Stop. REMARK: Results compare exactly. The
% results printed in the command window after executing
% this script are as follows:
%   v = g/k-g/k*exp(-k*t)
%   a  = g*exp(-k*t)
%   a2 = g/exp(k*t)
% These results verify the conclusion that the published
% formulas are correct.
%
```

We next consider an approximate method for solving the linear-friction case. This procedure is something you could implement if you didn't have the exact solution. More on numerical methods is provided in Chapter 17.

The equations for free-fall acceleration and velocity are differentials. The following approximate analysis of a differential equation is called finite-difference. Let's consider free fall with linear friction (i.e., air drag that is linearly proportional to the speed of free fall). The formula (or equation) for this and its exact solution were given above. In the following analysis a script to solve this problem by the approximate method is written and executed for the same interval of time. The approximate solution is compared with the exact solution graphically.

For a unit mass, the formula that describes the velocity of free fall from rest with air resistance proportional to the linear power of velocity can be written as follows:

$$\frac{dv}{dt} = g - kv,$$

We approximate this equation by inverting the fundamental theorem of differential calculus. That is, we rewrite it in terms of the definition of a derivative before the appropriate limit is taken. What is meant is that we write this equation for a small interval of time $\Delta t = t(n+1) - t(n)$ as follows:

$$\frac{dv}{dt} \approx \frac{v(n+1) - v(n)}{\Delta t} = g - k\left(\frac{v(n+1) + v(n)}{2}\right)$$

where the value identified by the integer n is the value of v at the beginning of the time interval (i.e., $v(n)$ at $t(n)$). The value identified by $n+1$ is the value of v at the end of the time interval (i.e., $v(n+1)$ at $t(n+1)$). This is an initial-value problem, so the value of v is known at $t(n)$. Knowing $v(n)$ at

$t(n)$, and specifying $t(n+1)$, the value $v(n+1)$ can be calculated by solving the finite-difference equation for $v(n+1)$ given above. The solution will depend on the size of Δt. This formula gives us $v(n+1)$.

We next need to solve for $y(n+1)$ from the definition of v (an approximate form of dy/dt). The average value of v over the interval of time, Δt, can be written in terms of y as follows:

$$\frac{dy}{dt} \approx \frac{y(n+1) - y(n)}{\Delta t} = \frac{v(n+1) + v(n)}{2}$$

Rearranging this equation, we get a formula for $y(n+1)$ in terms of $v(n+1)$, $v(n)$, $y(n)$, and Δt. Given the initial condition—that is, $y(n)$ and $v(n)$ at $t(n)$—we compute $v(n+1)$ with the first difference equation followed by $y(n+1)$ with the last equation. *Note that the new values become the initial condition for the next time step.* This procedure is repeated until the overall time duration of interest is reached (in this case $t = 5$). The script file to implement this procedure is given next. The M-file is executed and the results are plotted on the same graph with the exact solution. As shown, the comparison is remarkable. This is not always the case when approximate methods are used. The comparison is, of course, encouraging because we usually do not have exact results and need to resort to numerical approximations in our work.

The script file applied to examine the difference between the approximate method and the exact result is as follows:

```
%
% Approximate and exact solution comparison of
% free fall with linear friction for t = 0 to 5.
%
% Script by D. T. V. .......... September 2006.
% Revised by D.T.V. ............ November 2008.
%
% Step 1: Specified constants
%
k = 0.2;
g = 9.81;
%
% Step 2: Selection of time step for approximate solution
%           dt = t(n+1) - t(n)
dt = 0.01;
%
% Step 3: Initial condition
%
```

```
t(1) = 0.;
v(1) = 0.;
y(1) = 0.;
%
% Step 3: Sequential implementation of the approximate
% solution method by repeating the procedure 500 times
% (using a for loop).
%
for n = 1:1:500
    t(n+1) = t(n) + dt;
    v(n+1) = (v(n) + dt * (g-0.5*k*v(n)) )/(1.+dt*0.5*k);
    y(n+1) = y(n) + dt * 0.5 * (v(n+1) + v(n));
end
%
% Step 4: Exact solution over the same interval of time:
%
    ye = (g/k) * t - (g/(k^2)) * (1.-exp(-k*t));
%
% Step 5: Graphical comparison:
%
plot(t,y,'o',t,ye)
title('Comparison of numerics w/ exact results')
xlabel(' Time, t')
ylabel(' Distance of free fall from rest, y')
%
% Step 6: Comparison of distance at t=5
%
disp(' ');
fprintf(' y(t) = %f, ye(t) = %f at t = %f \n',...
          y(501),ye(501),t(501))
%
% Step 7: End of script by DTV.
%
```

The plot command in this script produced Figure 14.5. The exact solution is a green line through the center of the circle.

In summary, examination of the free-fall problem illustrates the application and the checking of formulae reported in the literature. An approximate method solves the same problem, illustrating a method that would be necessary if exact formulas were not found. Finally, the approximate method can certainly be improved by utilizing the ordinary differential equation solvers available in MATLAB. Examples of this type of procedure are given in Chapter 17. Other examples are in MATLAB; Help found via the question mark in the toolbar on the desktop (as already mentioned).

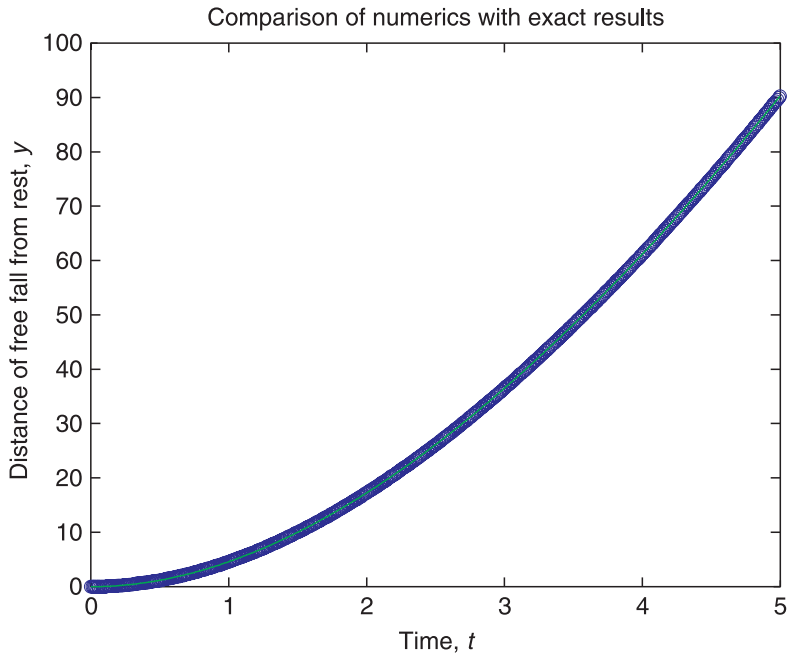

FIGURE 14.5

Comparison of free-fall distance: line, exact distance; circles, finite-difference approximate solution.

14.4 PROJECTILE WITH FRICTION

Let's examine the projectile problem again—in this case the effect of air resistance on the flight of a projectile such as a golf ball. For a unit mass, the formulae that describe the trajectory of a projectile in a gravitational field, g, with air resistance in the opposite direction of motion (proportional to the speed of the projectile in the direction of motion squared) are as follows:

$$u = dx/dt,$$

$$v = dy/dt,$$

$$du/dt = -ku * \sqrt{u^2 + v^2},$$

$$dv/dt = -kv * \sqrt{u^2 + v^2} - g.$$

The location of the projectile (the golf ball) is initially at $x = 0$, $y = 0$. It is launched at a speed V_s in the direction of angle θ as measured from the horizontal. The coordinate x is in the horizontal direction parallel to the ground. The coordinate y is in the direction perpendicular to x pointing toward the sky. Hence, the gravitational acceleration is in the negative y direction. For a given launch speed and direction, we wish to estimate the range (or distance) the ball travels in the x direction when it first hits the ground. To do this we approximate

the four equations similarly to how we treated the free-fall problem in the last section.

The solution method is given in detail in the script following (`golf.m`):

```
%
% "The Golf ball problem"
% Numerical computation of the trajectory of a
% projectile launched at an angle theta with
% a specified launch speed. They are:
%  theta = launch angle in degrees.
%  Vs = launch speed.
%
%  Script by D. T. V. ..... September 2006.
%  Revised by D.T.V.  .....  November 2008.
%
% Equations of motion:
%  u = dx/dt.  v = dy/dt. g is in the opposite
%  direction of y. x = y = 0 is the location of
%  the tee. k is the coefficient of air drag. It
%  is assumed to be constant. Friction is assumed
%  to be proportional to the speed of the ball
%  squared and it acts in the opposite direction
%  of the direction of motion of the ball. The
%  components of acceleration are thus:
%  du/dt = - [k (u^2 + v^2) * u/sqrt(u^2+v^2)].
%  dv/dt = - [k (u^2 + v^2) * v/sqrt(u^2+v^2)] - g.
%
% INPUT DATA
% Specified constants:
k = 0.02;
g = 9.81;
dt = 0.01;
%
% Input the initial condition:
%
theta = input(' Initial angle of launch: ')
the = theta * pi/180.;
Vs = input('  Initial speed of launch:  ')
u(1) = Vs * cos(the);
v(1) = Vs * sin(the);
% Launch pad location:
x(1) = 0.;
y(1) = 0.;
%
```

```
% Compute approximate solution of the trajectory
% of flight.
% Repeat up to 6000 time, i.e., until ball hits
% the ground.
%
for n=1:1:6000;
   u(n+1) = u(n) ...
    - dt * (k * sqrt(u(n)^2+v(n)^2) * u(n));
   v(n+1) = v(n) ...
    - dt * (k * sqrt(u(n)^2+v(n)^2) * v(n) + g);
   x(n+1) = x(n) + u(n) * dt;
   y(n+1) = y(n) + v(n) * dt;
% Determination of when the object hits ground:
  if y(n+1) < 0
      slope = (y(n+1) - y(n))/(x(n+1) - x(n));
      b = y(n) - slope * x(n);
      xhit = - b/slope;
   plot(x,y)
   fprintf(' The length of the shot =  %5.2f \n', xhit)
  end
% Once object hits terminate the computations with a break:
    if y(n+1) < 0; break; end
end
%
% Graphical presentation of the results:
%
if y(n+1) > 0
    plot(x,y)
end
% ------ End of golf-ball script by DTV
```

What is the optimum launch angle for $k = 0.02$? The answer is described next for a launch speed of 100. Note that the optimum angle is the one that gives the longest distance of travel (or greatest range). To compute the optimum angle, the following script was executed after the two input statements in the `golf.m` script were commented out by typing `%` at the beginning of the two lines containing them. The lines in question were commented out as follows:

```
%
% In golf.m the following alterations were made
% prior to running this script!!!!!!
%
% % theta = input(' Initial angle of launch: ')
% % Vs = input('  Initial speed of launch:  ')
%
```

```
%  This script then finds the optimum angle for k = 0.2
%  to compare with the zero friction case, which we know
%  has the optimum launch angle of 45 degrees.
%
% Script by D. T. V. ..... September 2006.
% Revised by D.T.V.  .....  November 2008.
%
% Consider launch angles from 1 to 45 degrees
%
th = 1:1:45;
vs = 100;          % Specified launch speed.
%
% Execute the modified golf.m file 45 times and save
% results for each execution of golf.m
%
for i=1:45
    theta = th(i)
    golf % Execution of modified golf.m script.
    xh(i) = xhit
    thxh(i) = theta
end
% Find the maximum distance and the correponding index
[xmh,n] = max(xh)
% Determine the angle that the maximum distance occurred.
opt_angle = thxh(n)
$ Display the results
disp(' optimum angle ')
disp( opt_angle )
% REMARK: For this case the result is 30 degrees.
% End of script
```

The optimum angle of launch for the case of nonlinear friction was computed and found to be equal to 30 degrees. Without friction it is 45 degrees. Hence, it is not surprising that golfers launch their best drives at angles significantly less than 45 degrees.

SUMMARY

In this chapter we examined three problems using some of the utilities in MATLAB:

- We determined the shape of the deflection of a cantilever beam based on a published formula in the engineering literature. We found we could use the capability of MATLAB to do arithmetic with polynomials.

- We examined the effect of friction on the free-fall problem. In addition to computing the distance and speed of free fall based on published formulae, we checked the formulae with the Symbolics tools, and we solved the problem by an approximate method to illustrate MATLAB's range of possibilities in solving technical problems.
- We examined the projectile problem subjected to air resistance by applying the same type of approximate method as applied in the free fall problem. We found that the optimum angle of launch with friction taken into account is less than 45 degrees (the frictionless value).

CHAPTER EXERCISES

14.1. Reproduce the script for the exact solutions for the free fall problem (**FFall.m**) and execute it for a range of friction coefficients (e.g., $k = 0.1$ and 0.3).

14.2. Use the Symbolics tools, in the same way they were used to check the formulae for the linear-friction free fall case, to check the other two cases.

14.3. Reproduce the **golf.m** (projectile script) and look at the effect of varying the friction coefficient k. Look at $k = 0.01$ and 0.03. What influence does friction have on the optimum angle for a launch speed of 100?

14.4. In designing objects, what shapes do we wish to consider? What equations do we need to solve in making design decisions? We need to learn how to model mathematically both physical and graphical problems. To build intuition about geometric figures you probably learned about the Mobius strip with only one side. The Klein bottle is related to this object. Find the Klein bottle example in MATLAB's. Help documentation by looking in **Demos** under **3-D visualization**. The script to draw it can be copied and pasted into the Editor. The code provided below with slight modifications of the parameters:

```
 n = 24;
 a = .5;                    % the diameter of the small tube
 c = .6;                    % the diameter of the bulb
 t1 = pi/4 : pi/n : 5*pi/4;      % parameter along the tube
 t2 = 5*pi/4 : pi/n : 9*pi/4;    % angle around the tube
 u  = pi/2 : pi/n : 5*pi/2;
[X, Z1] = meshgrid(t1,u);
[Y, Z2] = meshgrid(t2,u);
% The handle
  len = sqrt(sin(X).^2 + cos(2*X).^2);
  x1 = c*ones(size(X)).*(cos(X).*sin(X) ...
    - 0.5*ones(size(X))+a*sin(Z1).*sin(X)./len);
  y1 = a*c*cos(Z1).*ones(size(X));
  z1 = ones(size(X)).*cos(X) + a*c*sin(Z1).*cos(2*X)./len;
  handleHndl=surf(x1,y1,z1,X);
```

(*Continued*)

```
    set(handleHndl,'EdgeColor',[.5 .5 .5]);
    hold on;
  % The bulb
    r = sin(Y) .* cos(Y) - (a + 1/2) * ones(size(Y));
    x2 = c * sin(Z2) .* r;
    y2 = - c * cos(Z2) .* r;
    z2 = ones(size(Y)) .* cos(Y);
    bulbHndl=surf(x2,y2,z2,Y);
  % Modification of the graphics:
    set(bulbHndl,'EdgeColor',[.5 .5 .5])
         colormap(hsv);
         axis vis3d
         view(-37,30);
         axis off
         light('Position',[2 -4 5])
         light
    hold off
  % The following makes the surface transparent:
    shading faceted;
    set(handleHndl,'CData',X);
    set(bulbHndl,'CData',Y);
    set([handleHndl bulbHndl], ...
       'EdgeColor',[.5 .5 .5], ...
       'FaceAlpha',.5);
```