```
      odeSolver.step(); // do one time step using selected algorithm
   }

   public void setState(double x, double vx, double y, double vy) {
      state[0] = x;
      state[1] = vx;
      state[2] = y;
      state[3] = vy;
      state[4] = 0;
   }

   public double[] getState() {
      return state;
   }

   public void getRate(double[] state, double[] rate) {
      rate[0] = state[1]; // rate of change of x
      rate[1] = 0;        // rate of change of vx
      rate[2] = state[3]; // rate of change of y
      rate[3] = -g;       // rate of change of vy
      rate[4] = 1;        // dt/dt = 1
   }

   public void draw(DrawingPanel drawingPanel, Graphics g) {
      int xpix = drawingPanel.xToPix(state[0]);
      int ypix = drawingPanel.yToPix(state[2]);
      g.setColor(Color.red);
      g.fillOval(xpix-pixRadius, ypix-pixRadius, 2*pixRadius,
           2*pixRadius);
      g.setColor(Color.green);
      int xmin = drawingPanel.xToPix(-100);
      int xmax = drawingPanel.xToPix(100);
      int y0 = drawingPanel.yToPix(0);
      // draw a line to represent the ground
      g.drawLine(xmin, y0, xmax, y0);
   }
}
```

Listing 3.9 A target class for projectile motion simulation.

```
package org.opensourcephysics.sip.ch03;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.frames.*;

public class ProjectileApp extends AbstractSimulation {
   PlotFrame plotFrame = new PlotFrame("Time", "x,y",
                         "Position versus time");
   Projectile projectile = new Projectile();
   PlotFrame animationFrame = new PlotFrame("x", "y", "Trajectory");

   public ProjectileApp() {
      animationFrame.addDrawable(projectile);
      plotFrame.setXYColumnNames(0, "t", "x");
      plotFrame.setXYColumnNames(1, "t", "y");
   }

   public void initialize() {
```

```
      double dt = control.getDouble("dt");
      double x = control.getDouble("initial x");
      double vx = control.getDouble("initial vx");
      double y = control.getDouble("initial y");
      double vy = control.getDouble("initial vy");
      projectile.setState(x, vx, y, vy);
      projectile.setStepSize(dt);
      // estimate of size needed for display
      double size = (vx*vx+vy*vy)/10;
      animationFrame.setPreferredMinMax(-1, size, -1, size);
   }

   public void doStep() {
      // x vs time data added
      plotFrame.append(0, projectile.state[4], projectile.state[0]);
      // y vs time data added
      plotFrame.append(1, projectile.state[4], projectile.state[2]);
      animationFrame.append(0, projectile.state[0],
           projectile.state[2]); // trajectory data added
      projectile.step(); // advance the state by one time step
   }

   public void reset() {
      control.setValue("initial x", 0);
      control.setValue("initial vx", 10);
      control.setValue("initial y", 0);
      control.setValue("initial vy", 10);
      control.setValue("dt", 0.01);
      enableStepsPerDisplay(true);
   }

   public static void main(String[] args) {
      SimulationControl.createApp(new ProjectileApp());
   }
}
```

The analytical solution for free fall near the Earth's surface, (2.4), is well known, and thus finding a numerical solution is useful only as an introduction to numerical methods. It is not difficult to think of more realistic models of motion near the Earth's surface for which the equations of motion do not have simple analytical solutions. For example, if we take into account the variation of the Earth's gravitational field with the distance from the center of the Earth, then the force on a particle is not constant. According to Newton's law of gravitation, the force due to the Earth on a particle of mass m is given by

$$F = \frac{GMm}{(R+y)^2} = \frac{GMm}{R^2(1+y/R)^2} = mg\left(1 - 2\frac{y}{R} + \cdots\right), \tag{3.8}$$

where y is measured from the Earth's surface, R is the radius of the Earth, M is the mass of the Earth, G is the gravitational constant, and $g = GM/R^2$.

Problem 3.7 Position-dependent force

Extend `FallingParticleODE` to simulate the fall of a particle with the position-dependent force law (3.8). Assume that a particle is dropped from a height h with zero initial velocity and compute its impact velocity (speed) when it hits the ground at $y = 0$. Determine the value of h for which the impact velocity differs by one percent from its value