```
      return dt;
   }

   double step() {
      for(int i = 1, n = imagPsi.length-1;i<n;i++) {
         double imH = potential[i]*imagPsi[i]-0.5*(imagPsi[i+1]-
              2*imagPsi[i]+imagPsi[i-1])/dx2;
         realPsi[i] += imH*dt;
      }
      for(int i = 1, n = realPsi.length-1;i<n;i++) {
         double reH = potential[i]*realPsi[i]-0.5*(realPsi[i+1]-
              2*realPsi[i]+realPsi[i-1])/dx2;
         imagPsi[i] -= reH*dt;
      }
      return dt;
   }

   public double getV(double x) {
      return 0; // change this statement to model other potentials
   }
}
```

Before we can use the `TDHalfStep` class, we need to choose an initial wave function. A convenient form is the Gaussian wave packet with a width w centered about x_0 given by

$$\Psi(x,0) = \left(\frac{1}{2\pi w^2}\right)^{1/4} e^{ik_0(x-x_0)}\, e^{-(x-x_0)^2/4w^2}. \tag{16.36}$$

The expectation value of the initial velocity of the wave packet is $\langle v\rangle = p_0/m = \hbar k_0/m$. Note that the wave function has a nonzero momentum expectation value, which is known as a *momentum boost*. An implementation of (16.36) is shown in the `GaussianPacket` class. The constructor is passed the width, center, and momentum of the packet. Real and imaginary values can then be calculated at any x to fill the wave function arrays.

Listing 16.8 The `GaussianPacket` class creates a wave function with a Gaussian probability distribution and a momentum boost.

```
package org.opensourcephysics.sip.ch16;
public class GaussianPacket {
   double w, x0, p0;
   double w42;
   double norm;

   public GaussianPacket(double width, double center,
                         double momentum) {
      w = width;
      w42 = 4*w*w;
      x0 = center;
      p0 = momentum;
      norm = Math.pow(2*Math.PI*w*w, -0.25);
   }

   public double getReal(double x) {
      return norm*Math.exp(-(x-x0)*(x-x0)/w42)*Math.cos(p0*(x-x0));
   }

   public double getImaginary(double x) {
      return norm*Math.exp(-(x-x0)*(x-x0)/w42)*Math.sin(p0*(x-x0));
   }
}
```


To start the half-step algorithm, we need the value of $\mathrm{I}(x, t = \frac{1}{2}\Delta t)$ and $\mathrm{R}(x, t = 0)$. To obtain $\mathrm{I}(x, t = \frac{1}{2}\Delta t)$, we use the real component of the wave function to perform a half step:

$$\mathrm{I}(x, t + \Delta t/2) = \mathrm{I}(x, t) - \hat{H}\,\mathrm{R}(x, t)\,\frac{\Delta t}{2}. \tag{16.37}$$

The normalization factor must be computed after we correct the initial wave function using (16.37). For completeness, we list the `TDHalfStepApp` target class.

Listing 16.9 The `TDHalfStepApp` class solves the time-independent Schrödinger equation and displays the wave function.

```
package org.opensourcephysics.sip.ch16;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.frames.ComplexPlotFrame;

public class TDHalfStepApp extends AbstractSimulation {
   ComplexPlotFrame psiFrame = new ComplexPlotFrame("x", "|Psi|",
        "Wave function");
   TDHalfStep wavefunction;
   double time;

   public TDHalfStepApp() {
      // do not autoscale within this y-range
      psiFrame.limitAutoscaleY(-1, 1);
   }

   public void initialize() {
      time = 0;
      psiFrame.setMessage("t="+0);
      double xmin = control.getDouble("xmin");
      double xmax = control.getDouble("xmax");
      int numberOfPoints = control.getInt("number of points");
      double width = control.getDouble("packet width");
      double x0 = control.getDouble("packet offset");
      double momentum = control.getDouble("packet momentum");
      GaussianPacket packet = new GaussianPacket(width, x0, momentum);
      wavefunction =
           new TDHalfStep(packet, numberOfPoints, xmin, xmax);
      psiFrame.clearData(); // removes old data
      psiFrame.append(wavefunction.x, wavefunction.realPsi,
           wavefunction.imagPsi);
   }

   public void doStep() {
      time += wavefunction.step();
      psiFrame.clearData();
      psiFrame.append(wavefunction.x, wavefunction.realPsi,
           wavefunction.imagPsi);
      psiFrame.setMessage("t="+decimalFormat.format(time));
   }
```